

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. Images
16. Tables
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive
22. Forms
23. Iframes
24. Color Names
25. Color Values
26. Color Shades
27. JavaScript
28. Head
29. Entities
30. Symbols
31. Charset
32. URL Encode
33. XHTML

1. Introduction

- **What is HTML?**

HTML is a **markup** language for **describing** web documents (web pages).

HTML stands for **H**yper **T**ext **M**arkup **L**anguage.

A markup language is a set of **markup tags**.

HTML documents are described by **HTML tags**.

Each HTML tag **describes** different document content.

1. Introduction

- HTML Example

```
<!DOCTYPE html>  
<html>  
<body>  
<h1>My First Heading</h1>  
<p>My first paragraph</p>  
</body>  
</html>
```

1. Introduction

- **HTML Tags**

HTML tags are **keywords** (tag names) surrounded by **angle brackets**:

`<tagname>content</tagname>`

HTML tags normally come **in pairs** like `<p>` and `</p>`.

The first tag in a pair is the **start tag**, the second tag is the **end tag**.

The end tag is written like the start tag, but with a **slash** before the tag name.

The start tag is often called the **opening tag**. The end tag is often called the **closing tag**.

1. Introduction

- **HTML Page Structure**

```
<html>  
  
  <body>  
  
    <h1>This is a heading</h1>  
  
    <p>This is a paragraph.</p>  
  
    <p>This is another paragraph.</p>  
  
  </body>  
  
</html>
```

1. Introduction

- HTML Versions

Version	Year
HTML	1991
HTML+	1993
HTML 2.0	1995
HTML 3.2	1997
HTML 4.01	1999
XHTML	2000
HTML5	2012

1. Introduction

- **The <!DOCTYPE> Declaration**

The <!DOCTYPE> declaration helps the browser to display a web page correctly.

There are many different documents on the web, and a browser can only display an HTML page correctly if it knows the HTML version and type.

- **Common <!DOCTYPE> Declarations**

HTML5

```
<!DOCTYPE html>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

XHTML 1.0

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```


2. HTML Editors

- Notepad
- Notepad++
- TextEdit (Mac)

- Adobe Dreamweaver
- MS Expression Web
- CoffeeCup

- ... and much more

3. Basic Example

```
<!DOCTYPE html>
<html>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>

<p>This is a first paragraph.</p>
<p>This is the second paragraph.</p>
<p>This is 3<sup>rd</sup> paragraph.</p>

<a href="http://www.w3schools.com">This is a link</a>


</body>
</html>
```

4. HTML Elements

- **HTML Elements**

HTML elements are written with a **start** tag, with an **end** tag, with the **content** in between:

```
<tagname>content</tagname>
```

- **Nested HTML Elements**

HTML elements can be nested (elements can contain elements).
All HTML documents consist of nested HTML elements.

- **Tip: Use Lowercase Tags**

HTML tags are not case sensitive: <P> means the same as <p>.

The HTML5 standard does not require lowercase tags, but W3C **recommends** lowercase in HTML4, and **demands** lowercase for stricter document types like XHTML.

4. HTML Elements

- **Empty HTML Elements**

HTML elements with no content are called empty elements.

`
` is an empty element without a closing tag (the `
` tag defines a line break).

Empty element can be "closed" in the opening tag like this: `
`.

Important: HTML5 does not require empty elements to be closed. But if you need stricter validation, and make your document readable by XML parsers, please close all HTML elements.

5. HTML Attributes

- HTML elements can have **attributes**.
- Attributes provide **additional information** about an element.
- Attributes are always specified in **the start tag**.
- Attributes come in name/value pairs like: **name="value"**.
- HTML tags and their attributes: <http://www.w3schools.com/tags/>

Attribute	Description
alt	Specifies an alternative text for an image
disabled	Specifies that an input element should be disabled
href	Specifies the URL (web address) for a link
id	Specifies a unique id for an element
src	Specifies the URL (web address) for an image
style	Specifies an inline CSS style for an element
title	Specifies extra information about an element (displayed as a tool tip)
value	Specifies the value (text content) for an input element.

5. HTML Attributes

- **The lang Attribute**

The document language can be declared in the **<html>** tag.

The language is declared in the **lang** attribute.

Declaring a language is important for accessibility applications (screen readers) and search engines:

```
<!DOCTYPE html>  
<html lang="en-US">  
  <body>  
    <h1>My First Heading</h1>  
    <p>My first paragraph.</p>  
  </body>  
</html>
```

5. HTML Attributes

- **The title Attribute**

HTML paragraphs are defined with the `<p>` tag.

In this example, the `<p>` element has a **title** attribute.
The value of the attribute is "**About W3Schools**":

```
<!DOCTYPE html>
<html lang="en-US">
  <body>
    <p title="About W3Schools">
      W3Schools is a web developer's site.
      It provides tutorials and references covering
      many aspects of web programming,
      including HTML, CSS, JavaScript, XML, SQL, PHP, ASP, etc.
    </p>
  </body>
</html>
```

5. HTML Attributes

- **The href Attribute**

HTML links are defined with the `<a>` tag. The link address is specified in the **href** attribute:

```
<!DOCTYPE html>
<html lang="en-US">
  <body>
    <a href="http://www.w3schools.com">This is a link</a>
  </body>
</html>
```

- **Size Attributes**

HTML images are defined with the `` tag.

The filename of the source (**src**), and the size of the image (**width** and **height**) are all provided as **attributes**:

```

```


5. HTML Attributes

- **The alt Attribute**

The **alt** attribute specifies an alternative text to be used, when an HTML element cannot be displayed.

The value of the attribute can be read by "screen readers". This way, someone "listening" to the webpage, i.e. a blind person, can "hear" the element.

```

```

- **Tip: Use lowercase Attributes**

The HTML5 standard does not require lower case attribute names.

The title attribute can be written with upper or lower case like **Title** and/or **TITLE**.

W3C **recommends** lowercase in HTML4, and **demands** lowercase for stricter document types like XHTML.

5. HTML Attributes

- **Suggestion: Always Quote Attribute Values**

The HTML5 standard does not require quotes around attribute values.

W3C **recommends** quotes in HTML4, and **demands** quotes for stricter document types like XHTML.

Double style quotes are the most common in HTML, but single style can also be used.

Sometimes it is **necessary** to use quotes:

```
<!DOCTYPE html>
<html>
  <body>
    <h1>About W3Schools</h1>
    <p title="About W3Schools">
      You cannot ommit qoutes around an attribute value
      if the value contains spaces.
    </p>
  </body>
</html>
```

6. HTML Head

- **The HTML <head> Element**

The HTML <head> element has nothing to do with HTML headings.

The HTML <head> element only contains meta data.

The HTML <head> element is placed between the <html> tag and the <body> tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>My First HTML</title>
    <meta charset="UTF-8">
  </head>
  <body>
    <h1>About W3Schools</h1>
    ...
  </body>
</html>
```

Exercise: Use W3Schools' tag reference for additional information about these tags and their attributes.

7. HTML Paragraphs

■ The HTML `<p>` Element defines a paragraph

- HTML documents are divided into paragraphs.
- You cannot be sure how HTML will be displayed.
- Large or small screens, and resized windows will create different results.
- With HTML, you cannot change the output by adding extra spaces or extra lines in your HTML code.
- The browser will remove extra spaces and extra lines when the page is displayed.
- Any number of spaces, and any number of new lines, count as **only one space**.
- Don't Forget the End Tag: Most browsers will display HTML correctly even if you forget the end tag.

■ HTML Line Breaks

- The HTML `
` element defines a line break.
- Use `
` if you want a line break (a new line) without starting a new paragraph:

```
<p>This is<br>a para<br>graph with line breaks</p>
```

7. HTML Paragraphs

■ The HTML `<pre>` Element

- The HTML `<pre>` element defines a block of pre-formatted text, with structured spaces and lines.
- To display anything, with right spacing and line-breaks, you must wrap the text in a `<pre>` element:

```
<p>This will display as a poem:</p>
```

```
<pre>
```

```
My Bonnie lies over the ocean.
```

```
My Bonnie lies over the sea.
```

```
My Bonnie lies over the ocean.
```

```
Oh, bring back my Bonnie to me.
```

```
</pre>
```

Tag	Description
<code><p></code>	Defines a paragraph
<code>
</code>	Inserts a single line break
<code><pre></code>	Defines pre-formatted text

9. HTML Styles

■ The HTML STYLE-Attribute

- HTML styling has nothing to do with formatting elements.
- Styling is about changing or adding the style of existing HTML elements.
- Every HTML element has a default style: background color is white, text color is black ...
- Changing the default style of an HTML element, can be done with the style attribute.
- The HTML style attribute has the following syntax:

style="property:value"

- Example

```
<body style="background-color:lightgrey">  
  <h1 style="color:blue">This is a heading</h1>  
  <p style="color:red">This is a paragraph.</p>  
</body>
```

9. HTML Styles

■ HTML Text Fonts, Size and Alignment

- The font-family property defines the font to be used for an HTML element.
- The font-size property defines the text size to be used for an HTML element.
- The text-align property defines the horizontal text alignment for an HTML element.
- Any combinations are possible.
- **Important:** The <center> tag, supported in older versions of HTML, is not valid in HTML5.

```
<body>  
  <h1 style="font-family:verdana">This is a heading</h1>  
  <p style="font-size:300%; text-align:center">This is a para...</p>  
</body>
```

10. Quotations & Citations

■ HTML Short Quotations

- The HTML `<q>` element defines a short quotation.
- Browsers usually insert quotation marks around the `<q>` element.

```
<body>  
  <p>WWF's goal is to: <q>Build a future where people ...</q></p>  
</body>
```

■ HTML Long Quotations

- The HTML `<blockquote>` element defines a quoted section.
- Browsers usually indent `<blockquote>` elements.

```
<blockquote cite="http://www.worldwildlife.org/who/index.html">  
  ...  
  For 50 years, WWF has been protecting the future of nature.  
  ...  
</blockquote>
```


10. Quotations & Citations

Tag	Description
<abbr>	Defines an abbreviation or acronym
<address>	Defines contact information for the author/owner of a document
<bdo>	Defines the text direction
<blockquote>	Defines a section that is quoted from another source
<q>	Defines an inline (short) quotation
<cite>	Defines the title of a work
<dfn>	Defines a definition term

Literature

- W3 Schools
<http://www.w3schools.com/>
- W3 Schools --- HTML Tag Reference
<http://www.w3schools.com/tags/>
- Notepad++
<http://notepad-plus-plus.org/>

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. Images
16. Tables
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive
22. Forms
23. Iframes
24. Color Names
25. Color Values
26. Color Shades
27. JavaScript
28. Head
29. Entities
30. Symbols
31. Charset
32. URL Encode
33. XHTML

11. Computercode

■ Computer Code Formatting

- Normally, HTML uses variable letter size, and variable letter spacing.
- This is not wanted when displaying examples of computer code.
- The `<kbd>`, `<samp>`, and `<code>` elements all support fixed letter size and spacing.

Example

```
<!DOCTYPE html>
<html>
<body style="font-size:16px">

<p>Keyboard input: <kbd>File | Open...</kbd></p>
<p>Sample input: <samp>demo.example.com ... Linux 2.6.10</samp></p>
<p>Code input:
    <code>var pers = { firstName:"John", lastName:"Doe" }</code>
</p>

</body>
</html>
```

11. Computercode

■ Math Formula == Variable Formatting

- The HTML `<var>` element defines a mathematical variable.
- The `<sup>` element is used for formatting mathematical exponents.
- The `<sub>` element is used for formatting mathematical indices.

Example

```
<!DOCTYPE html>
<html>
<body style="font-size:16px">

<p>Einstein wrote:</p>
<p><var>E = m c<sup>2</sup></var></p>
<p><var>E<sub>1</sub> = m<sub>1</sub> c<sup>2</sup></var></p>

</body>
</html>
```

12. Comments

■ HTML Comments

- You can add comments to your HTML source by using the following syntax:

```
<!-- Write your comments here -->
```

- There is an exclamation point (!) in the opening tag, but not in the closing tag.
- Comments are not displayed by the browser, but they can help document your HTML.
- With comments you can place notifications and reminders in your HTML.
- Comments are also great for debugging HTML.

Example

```
<!DOCTYPE html>
<html>
<body style="font-size:16px">

<p><var>E = m c<sup>2</sup></var></p>
<!-- Do not display this at the moment

-->
</body>
</html>
```

12. Comments

■ Conditional Comments

- Conditional comments defines HTML tags to be executed by IE only:

Example

```
<!DOCTYPE html>
<html>
<body style="font-size:16px">

<!--[if IE 8]>
    .... some HTML here ...
<![endif]-->

</body>
</html>
```


12. Comments

- **Conditional Comments**

Value	Function	Sample
! IE	if not	<!--[if !IE]>
IE	if Internet Explorer	<!--[if IE]>
IE 5.5	if Internet Explorer Version 5.5	<!--[if IE 5.5]>
IE 8	if Internet Explorer Version 8	<!--[if IE 8]>
mso	if Microsoft Office	<!--[if mso]>
mso 15	if Microsoft Office 2013	<!--[if mso 15]>
vml	if VML supported	<!--[if vml]>

12. Comments

- **Conditional Comments**

Operator	Function	Sample
!	Not-Operator	<!--[if!(IE 6)]> (if not IE 6)
lt	Lower-than-Operator	<!--[if lt IE 6]> (if lower than IE 6)
lte	Lower-or-equal-than	<!--[if lte IE 6]> (if lower than or equal IE 6)
gt	Greater-than	<!--[if gt IE 6]> (if greater than IE 6)
gte	Greater-or-equal-than	<!--[if gte IE 6]>
&	And-Operator	<!--[if mso &!vml]> (if Office without VML)
	Or-Operator	<!--[if mso ie]> (if Office or IE)

13. Styles --- CSS

■ Styling HTML with CSS

- CSS stands for Cascading Style Sheets.
- Styling can be added to HTML elements in 3 ways:
 - Inline - using a style attribute in HTML elements
 - Internal - using a <style> element in the HTML <head> section
 - External - using one or more external CSS files

■ CSS Syntax

- CSS styling has the following syntax:

```
element { property:value ; property:value }
```

- The element is an HTML element name.
- The property is a CSS property.
- The value is a CSS value.
- Multiple styles are separated with semicolon.

13. Styles --- CSS

■ Inline Styling (Inline CSS)

- Inline styling is useful for applying a unique style to a single HTML element.
- Inline styling uses the style attribute.
- Example: This inline styling changes the text color of a single paragraph.

```
<h1 style="color:blue">This is a Blue Heading</h1>
```

13. Styles --- CSS

■ Internal Styling (Internal CSS)

- An internal style sheet can be used to define a common style for all HTML elements on a page.
- Internal styling is defined in the <head> section of an HTML page, using a <style> element.

- Example:

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {background-color:lightgrey}
    h1   {color:blue}
    p    {color:green}
  </style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

13. Styles --- CSS

■ External Styling (External CSS)

- External style sheets are ideal when the style is applied to many pages.
- With external style sheets, you can change the look of an entire site by changing one file.
- External styles are defined in the <head> section of an HTML page, in the <link> element.
- Example:

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="myStyles.css">
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

13. Styles --- CSS

■ CSS Fonts

- The CSS property color defines the text color to be used for an HTML element.
- The CSS property font-family defines the font to be used for an HTML element.
- The CSS property font-size defines the text size to be used for an HTML element.

- Example:

```
<!DOCTYPE html>
<html>
<head>
  <style> h1 {
    color:blue;
    font-family:verdana;
    font-size:300%;
  }
</style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```

13. Styles --- CSS

■ CSS Box Model

- Every visible HTML element has a box around it, even if you cannot see it.
- The CSS border property defines a visible border around an HTML element.
- The CSS padding property defines a padding (space) inside the border.
- The CSS margin property defines a margin (space) outside the border.

Example:

```
<!DOCTYPE html>
<html>
<head>
  <style> p {
                                border:1px solid black;
                                padding:10px;
                                margin:30px;
                              } </style>
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph.</p>
</body>
</html>
```


13. Styles --- CSS

■ CSS id Attribute

- All the examples above use CSS to style HTML elements in a general way.
- The CSS styles define an equal style for all equal elements.
- To define a special style for a special element, first add an id attribute to the element.
- In a second step define a different style for the (identified) element.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <style> p#p01 { color:blue; } </style>
</head>
<body>
<p>This is a paragraph.</p>
<p id="p01">I am different.</p>
</body>
</html>
```

13. Styles --- CSS

■ CSS class Attribute

- To define a style for a special type (class) of elements, add a class attribute to the element.
- Then you can define a different style for this type (class) of element.
- Use id to address single elements. Use class to address groups of elements.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <style> p.error { color:red; } </style>
</head>
<body>
<p>This is a paragraph.</p>
<p class="error">I am an error.</p>
</body>
</html>
```

14. Links

■ HTML Links - Hyperlinks

- HTML links are hyperlinks.
- A hyperlink is an element, a text, or an image that you can click on, and jump to another document.
- HTML Links – Syntax

In HTML, links are defined with the `<a>` tag:

```
<a href="url">link text</a>
```

Example:

```
<!DOCTYPE html>
<html>
<body>
<a href="http://www.w3schools.com/html/">Visit our HTML tutorial</a>
</body>
</html>
```

- The href attribute specifies the destination address.
- The link text is the visible part.
- Clicking on the link text, will send you to the specified address.

14. Links

■ HTML Links – Colors

- When you move the mouse cursor over a link, two things will normally happen:
 - The mouse arrow will turn into a little hand
 - The color of the link element will change
- By default, links will appear as this in all browsers:
 - An unvisited link is underlined and blue
 - A visited link is underlined and purple
 - An active link is underlined and red
- You can change the default colors, using styles:

```
<style>  
  a:link      {color:#000000; background-color:transparent}  
  a:visited   {color:#000000; background-color:transparent}  
  a:hover     {color:#ff0000; background-color:transparent}  
  a:active    {color:#ff0000; background-color:transparent}  
</style>
```

14. Links

■ HTML Links – The target Attribute

- The target attribute specifies where to open the linked document.
- This example will open the linked document in a new browser window or in a new tab.

```
<a href "http://www.w3schools.com/" target="_blank">Visit W3Schools!</a>
```

Target Value	Description
<code>_blank</code>	Opens the linked document in a new window or tab
<code>_self</code>	Opens the linked document in the same frame as it was clicked (default)
<code>_parent</code>	Opens the linked document in the parent frame
<code>_top</code>	Opens the linked document in the full body of the window
<code>framename</code>	Opens the linked document in a named frame

14. Links

■ HTML Links – Image as a Link

- It is common to use images as links.
- `border:0` is added to prevent IE9 (and earlier) from displaying a border around the image.
- Use the `alt` attribute for describing your pictures!

```
<a href="default.asp">  
      
</a>
```

14. Links

■ HTML Links – The id Attribute

- The id attribute can be used to create bookmarks inside HTML documents.
- Bookmarks are not displayed in any special way.
- Bookmarks are invisible to the reader.
- Example:
 - Add an id attribute to any <a> element.

```
<a id="tips">Useful Tips Section</a>
```

- Create a link to the <a> element (Useful Tips Section).

```
<a href="#tips">Visit the Useful Tips Section</a>
```

- Or, create a link to the <a> element (Useful Tips Section) from another page.

```
<a href="http://www.w3schools.com/html_links.htm#tips">  
    Visit the Useful Tips Section  
</a>
```

Literature

- W3 Schools
<http://www.w3schools.com/>
- W3 Schools --- HTML Tag Reference
<http://www.w3schools.com/tags/>
- Notepad++
<http://notepad-plus-plus.org/>

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. **Images**
16. **Tables**
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive
22. Forms
23. Iframes
24. Color Names
25. Color Values
26. Color Shades
27. JavaScript
28. Head
29. Entities
30. Symbols
31. Charset
32. URL Encode
33. XHTML

15. Images

■ HTML Images Syntax

- The `` tag is empty, it contains attributes only, and does not have a closing tag.
- The **src** attribute defines the url (web address) of the image.
- The **alt** attribute specifies an alternate text for the image, if it cannot be displayed.
- The **alt** attribute is required!

```

```

- Example

```

```

■ HTML Screen Reader

- Screen readers are software programs that can read what is displayed on a screen.
- Used on the web, screen readers can "reproduce" HTML as text-to-speech, sound icons, or braille output.
- Screen readers are used by people who are blind, visually impaired, or learning disabled.
- Screen readers can read the **alt** attribute.

15. Images

■ Image Size - Width and Height

- You can use the **style** attribute to specify the **width** and **height** of an image.
- The values are specified in pixels (use px after the value).
- Examples

```

```

```

```

- Both the width, the height, and the style attributes, are valid in the latest HTML5 standard.
- If you use the style attribute. It prevents styles sheets to change the default size of images.

15. Images

■ Using an Image as a Link

- It is common to use images as links.
- Example

```
<a href="default.asp">  
    
</a>
```

- We have added border:0 to prevent IE9 (and earlier) from displaying a border around the image.

15. Images

■ Image Maps

- For an image, you can create an image map, with clickable areas.
- Example

```

```

```
<map name="planetmap">  
  <area shape="rect"    coords="0,0,82,126" alt="Sun" href="sun.htm">  
  <area shape="circle" coords="90,58,3"    alt="Mercury" href="mercur.htm">  
  <area shape="circle" coords="124,58,8"   alt="Venus" href="venus.htm">  
</map>
```

15. Images

■ Image Floating

- You can let an image float to the left or right of a paragraph.
- Examples

```
<p>  
    
</p>
```

```
<p>  
    
</p>
```

16. Tables

■ Defining HTML Tables

- Tables are defined with the `<table>` tag.
- Tables are divided into **table rows** with the `<tr>` tag.
- Table rows are divided into **table data** with the `<td>` tag.
- A table row can also be divided into **table headings** with the `<th>` tag.
- Table data `<td>` are the data containers of the table. They can contain all sorts of HTML elements like text, images, lists, other tables, etc.
- Example

```
<table style="width:100%">
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
</table>
```


16. Tables

Tag	Description
<code><table></code>	Defines a table
<code><th></code>	Defines a header cell in a table
<code><tr></code>	Defines a row in a table
<code><td></code>	Defines a cell in a table
<code><caption></code>	Defines a table caption
<code><colgroup></code>	Specifies a group of one or more columns in a table for formatting
<code><col></code>	Specifies column properties for each column within a <code><colgroup></code> element
<code><thead></code>	Groups the header content in a table
<code><tbody></code>	Groups the body content in a table
<code><tfoot></code>	Groups the footer content in a table

16. Tables

■ HTML Table Headings

- A table row can also be divided into **table headings** with the `<th>` tag.
- Table headings are defined with the `<th>` tag.
- By default, all major browsers display table headings as bold and centered.
- Example

```
<table style="width:100%">  
  <tr>  
    <th>Firstname</th>  
    <th>Lastname</th>  
    <th>Points</th>  
  </tr>  
  <tr>  
    <td>Eve</td>  
    <td>Jackson</td>  
    <td>94</td>  
  </tr>  
</table>
```

16. Tables

■ Table Cells that Span Many Columns

- To make a cell span more than one column, use the **colspan** attribute.
- Example

```
<table style="width:100%">  
  <tr>  
    <th>Name</th>  
    <th colspan="2">Telephone</th>  
  </tr>  
  <tr>  
    <td>Bill Gates</td>  
    <td>555 77 854</td>  
    <td>555 77 855</td>  
  </tr>  
</table>
```

Name	Telephone	
Bill Gates	555 77 854	555 77 855

16. Tables

■ Table Cells that Span Many Rows

- To make a cell span more than one row, use the **rowspan** attribute.
- Example

```
<table style="width:100%">
  <tr>
    <th>Name</th>
    <td>Bill Gates</td>
  </tr>
  <tr>
    <th rowspan="2">Telephone:</th>
    <td>555 77 854</td>
  </tr>
  <tr>
    <td>555 77 855</td>
  </tr>
</table>
```

First Name:	Bill Gates
Telephone:	555 77 854
	555 77 855

16. Tables

■ HTML Table With a Caption

- To add a caption to a table, use the `<caption>` tag.
- Example

```
<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>
```

Monthly savings

Month	Savings
January	\$100
February	\$50

16. Tables

■ HTML <thead>, <tfoot> and <tbody> Tag

- The <thead> tag is used to group header content in an HTML table.
- The <thead> element is used in conjunction with the <tbody> and <tfoot> elements to specify each part of a table (header, body, footer).
- Browsers can use these elements to enable **scrolling of the table body independently of the header and footer**. Also, when printing a large table that spans multiple pages, these elements can enable the table header and footer to be printed at the top and bottom of each page.
- The <thead> tag must be used in the following context:
 - As a child of a <table> element,
 - after any <caption>, and <colgroup> elements, and
 - before any <tbody>, <tfoot>, and <tr> elements.
- The <thead> element must have one or more <tr> tags inside.
- The <thead>, <tbody>, and <tfoot> elements will not affect the layout of the table by default. However, you can use CSS to style these elements.

Month	Savings
January	\$100
February	\$80
Sum	\$180

16. Tables

- HTML `<thead>`, `<tfoot>` and `<tbody>` Tag

Month	Savings
January	\$100
February	\$80
Sum	\$180

```

<!DOCTYPE html>
<html>
<head>
<style>
  head {color:green;}
  tbody {color:blue;}
  tfoot {color:red;}
  table, th, td { border: 1px solid black; }
</style>
</head>
<body>
<table>
  <thead>
    <tr>
      <th>Month</th>
      <th>Savings</th>
    </tr>
  </thead>
  <tfoot>
    <tr>
      <td>Sum</td>
      <td>$180</td>
    </tr>
  </tfoot>
  <tbody>
    <tr>
      <td>January</td>
      <td>$100</td>
    </tr>
    <tr>
      <td>February</td>
      <td>$80</td>
    </tr>
  </tbody>
</table>
</body>
</html>

```

```

<thead>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
</thead>

```

```

<tfoot>
  <tr>
    <td>Sum</td>
    <td>$180</td>
  </tr>
</tfoot>

```

```

<tbody>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  ...
</tbody>

```

16. Tables

■ HTML <colgroup> Tag

- The <colgroup> tag specifies a group of one or more columns in a table for formatting.
- The <colgroup> tag is useful for applying styles to entire columns, instead of repeating the styles for each cell, for each row.
- The <colgroup> tag must be a child of a <table> element, after any <caption> elements and before any <thead>, <tbody>, <tfoot>, and <tr> elements.
- To define different properties to a column within a <colgroup>, use the <col> tag within the <colgroup> tag.
- Example

```
<table>
  <colgroup>
    <col span="2" style="background-color:red">
    <col style="background-color:yellow">
  </colgroup>
  <tr>
    <th>ISBN</th>
    <th>Title</th>
    <th>Price</th>
  </tr>
  <tr>
    <td>3476896</td>
    <td>My first HTML</td>
    <td>$53</td>
  </tr>
</table>
```

ISBN	Title	Price
3476896	My first HTML	\$53
5869207	My first CSS	\$49

Literature

- W3 Schools
<http://www.w3schools.com/>
- W3 Schools --- HTML Tag Reference
<http://www.w3schools.com/tags/>
- Notepad++
<http://notepad-plus-plus.org/>

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. Images
16. Tables
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive Web Design
22. Forms
23. Iframes
24. Color Names
25. Color Values
26. Color Shades
27. JavaScript
28. Head
29. Entities
30. Symbols
31. Charset
32. URL Encode
33. XHTML

17. Lists

■ Unordered HTML List

- The first item
 - The second item
 - The third item
-
- An unordered list starts with the `` tag.
 - Each list item starts with the `` tag.
 - The list items will be marked with bullets (small black circles).
-
- Example

```
<ul>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

17. Lists

■ Unordered HTML List --- Style Attributes

Style	Description
list-style-type:disc	The list items will be marked with bullets (default)
list-style-type:circle	The list items will be marked with circles
list-style-type:square	The list items will be marked with squares
list-style-type:none	The list items will not be marked

- Example: Circle

```
<ul style="list-style-type:circle">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ul>
```

17. Lists

▪ Ordered HTML List

1. The first item
 2. The second item
 3. The third item
- An ordered list starts with the `` tag.
 - Each list item starts with the `` tag.
 - The list items will be marked with numbers.

- Example

```
<ol>  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

17. Lists

- **Ordered HTML List --- Type Attribute**

Type	Description
type="1"	The list items will be numbered with numbers (default)
type="A"	The list items will be numbered with uppercase letters
type="a"	The list items will be numbered with lowercase letters
type="I"	The list items will be numbered with uppercase roman numbers
type="i"	The list items will be numbered with lowercase roman numbers

- Example:

```
<ol type="A">  
  <li>Coffee</li>  
  <li>Tea</li>  
  <li>Milk</li>  
</ol>
```

17. Lists

▪ HTML Description List

The first item

Description of first item

The second item

Description of second item

- A description list, is a list of terms, with a description of each term.
- The `<dl>` tag defines a description list.
- The `<dt>` tag defines the term (name), and
- the `<dd>` tag defines the data (description).

- Example

```
<dl>
  <dt>Coffee</dt>
  <dd>- black hot drink</dd>
  <dt>Milk</dt>
  <dd>- white cold drink</dd>
</dl>
```


17. Lists

▪ Nested Lists

- List can be nested (lists inside lists).
- List items can contain new list, and other HTML elements, like images and links, etc.

- Example

```
<ul>  
  <li>Coffee</li>  
  <li>Tea  
    <ol>  
      <li>Black tea</li>  
      <li>Green tea</li>  
    </ol>  
  </li>  
  <li>Milk</li>  
</ul>
```

Result

- Coffee
- Tea
 - 1. Black tea
 - 2. Green tea
- Milk

17. Lists

▪ Horizontal Lists

- HTML lists can be styled in many different ways with CSS.
- One popular way, is to style a list to display horizontally.
- Example

```
<!DOCTYPE html>
<html>

<head>
<style>
  ul#myMenu li { display:inline; }
</style>
</head>

<body>

<h2>Horizontal List</h2>

<ul id="myMenu">
  <li>Apples</li>
  <li>Bananas</li>
  <li>Lemons</li>
  <li>Oranges</li>
</ul>

</body>
</html>
```

17. Lists

■ Horizontal Lists

- With a little extra style, you can make it look like a real menu.
- Example

Tables

Lists

Blocks

Classes

```
<style>
ul#menu {
    padding: 0;
}

ul#menu li {
    display: inline;
}

ul#menu li a {
    background-color: black;
    color: white;
    padding: 10px 20px;
    text-decoration: none;
    border-radius: 4px 4px 0 0;
}

ul#menu li a:hover {
    background-color: orange;
}
</style>
```

18. Blocks

■ HTML Block Elements and Inline Elements

- Most HTML elements are defined as **block level** elements or **inline** elements.
- Block level elements normally start (and end) with a new line, when displayed in a browser.
- Examples: `<h1>`, `<p>`, ``, `<table>`
- Inline elements are normally displayed without line breaks.
- Examples: ``, `<td>`, `<a>`, ``

18. Blocks

- **The HTML <div> Element**
 - The HTML <div> element is a **block level element** that can be used as a container for other HTML elements.
 - The <div> element has no special meaning. It has no required attributes, but **style** and **class** are common.
 - Because it is a block level element, the browser will display line breaks before and after it.
 - When used together with CSS, the <div> element can be used to style blocks of content.

London

London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.

```
<!DOCTYPE html>
<html>

<body>

<div style="background-color:darkgray;
          color:darkblue;
          margin:20px;
          padding:20px;">

<h2>London</h2>
... London is the capital city of England.
It is the most populous city in the United
Kingdom, with a metropolitan area of over
13 million inhabitants.
...
</div>
</body>
</html>
```

18. Blocks

■ The HTML `` Element

- The HTML `` element is an **inline element** that can be used as a container for text.
- The `` element has no special meaning. It has no required attributes, but **style** and **class** are common.
- Unlike `<div>`, which is formatted with line breaks, the `` element does not have any automatic formatting.
- When used together with CSS, the `` element can be used to style parts of the text.

My Important Heading

```
<!DOCTYPE html>
<html>
<body>
<h1>
  My
  <span style="color:red">Important</span>
  Heading
</h1>
</body>
</html>
```

19. Classes

■ HTML Classes

- Classing HTML elements, makes it possible to define CSS styles for classes of elements.
- Equal styles for equal classes, or different styles for different classes.
- Classing Block Elements
 - The HTML <div> element is a **block level** element. It can be used as a container for other HTML elements.
 - Classing <div> elements, makes it possible to define equal styles for equal <div> elements.

```

<!DOCTYPE html>
<html>

<head>
<style>
.news {
    background-color:lightgray;
    color:darkblue;
    margin:10px;
    padding:5px;
}
</style>
</head>

<body>

<div class="news">
    <h2>London</h2>
    <p>Standing on the River Thames, ...</p>
</div>

<div class="news">
    <h2>Paris</h2>
    <p>La capitale de la France, ...</p>
</div>

</body>
</html>

```

London

Standing on the River Thames, ...

Paris

La capitale de la France, ...

19. Classes

■ HTML Classes

- Classing Inline Elements
 - The HTML `` element is an inline element that can be used as a container for text.
 - Classing `` elements makes it possible to design equal styles for equal `` elements.

My **Important** Heading

```
<!DOCTYPE html>
<html>

<head>
<style>
  span.red {color:red;}
</style>
</head>

<body>

<h1>My <span class="red">Important</span>
Heading</h1>

</body>
</html>
```


20. Layouts

- **Website Layout Using <div> Elements**
 - The <div> element is often used as a layout tool, because it can easily be positioned with CSS.
 - The example uses 4 <div> elements to create a multiple column layout.



The screenshot shows a website layout with a grey header containing the title "City Gallery" in blue. Below the header is a light grey sidebar on the left with a list of city names: "London", "Paris", and "Tokyo". The main content area on the right displays the selected city, "London", in bold. Below the city name is a paragraph of text: "London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants." followed by another paragraph: "Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium." At the bottom of the page, a grey footer contains the text "Copyright © W3Schools.com".

20. Layouts

- Website Layout Using <div> Elements

```

<!DOCTYPE html>
<html>

<head>
<style>
    ...
</style>
</head>

<body>

<div id="header"><h1>City Gallery</h1></div>

<div id="nav">London<br>Paris<br>Tokyo<br></div>

<div id="section"><h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom,with a metropolitan area of over 13 million inhabitants.</p>

<p>Standing on the River Thames, London has been a major settlement for two millennia,its history going back to its founding by the Romans, who named it Londinium.</p>

</div>

<div id="footer">Copyright © W3Schools.com</div>

</body>
</html>

```



20. Layouts

- Website Layout Using <div> Elements

```

<!DOCTYPE html>
<html>

<head>
<style>
  #header {
    background-color:darkgray;
    color:darkblue;
    text-align:center;
    padding: 5px;
  }
  #nav {
    line-height:30px;
    background-color:#eeeeee;
    height:300px;
    width:100px;
    float:left;
    padding: 5px;
  }
  #section {
    width:350px;
    float:left;
    padding:10px;
  }

```

```

  #footer {
    background-color:darkgray;
    color:darkblue;
    clear:both;
    text-align:center padding: 5px;
  }
</style>
</head>

<body>
  ...

</body>
</html>

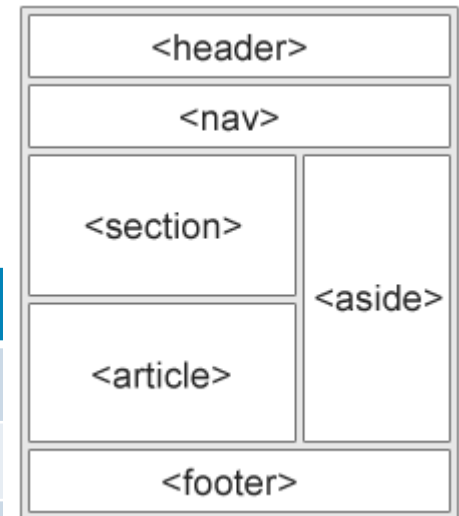
```



20. Layouts

- **Website Layout Using HTML 5**
 - HTML5 offers new semantic elements that define different parts of a web page.

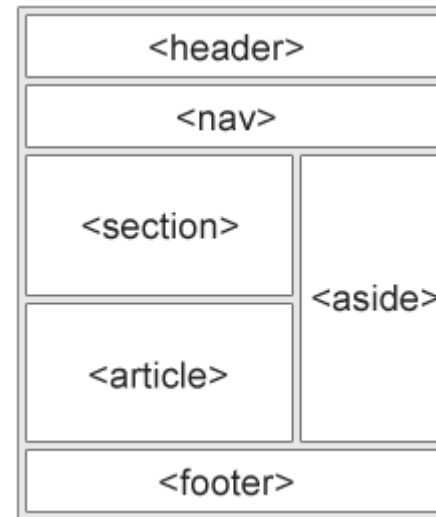
Element	Description
header	Defines a header for a document or a section
nav	Defines a container for navigation links
section	Defines a section in a document
article	Defines an independent self-contained article
aside	Defines content aside from the content (like a sidebar)
footer	Defines a footer for a document or a section
details	Defines additional details
summary	Defines a heading for the details element



20. Layouts

Website Layout Using HTML 5

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ...
    </style>
  </head>
  <body>
    <header><h1>City Gallery</h1></header>
    <nav>London<br>Paris<br>Tokyo<br></nav>
    <section><h2>London</h2> ... </section>
    <footer>Copyright © W3Schools.com</footer>
  </body>
</html>
```



20. Layouts

Website Layout Using HTML 5

```
<!DOCTYPE html>
<html>
```

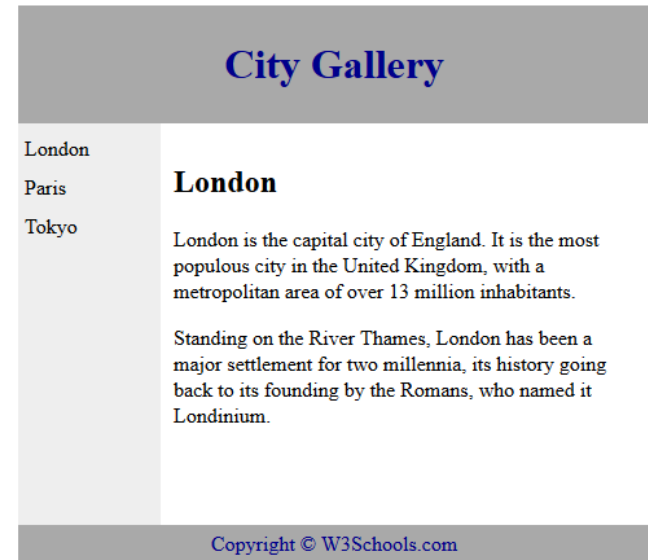
```
<head>
<style>
```

```
header {
  background-color:darkgray;
  color:darkblue;
  text-align:center;
  padding: 5px;
}
```

```
nav {
  line-height:30px;
  background-color:#eeeeee;
  height:300px;
  width:100px;
  float:left;
  padding: 5px;
}
```

```
section {
  width:350px;
  float:left;
  padding:10px;
}
```

```
footer {
  background-color:darkgray;
  color:darkblue;
  clear:both;
  text-align:center padding: 5px;
}
</style>
</head>
<body>
...
</body>
</html>
```



21. Responsive Web Design

■ What is Responsive Web Design?

- RWD stands for Responsive Web Design
- RWD can deliver web pages in variable sizes
- RWD is a must for tablets and mobile devices

- One way to create a responsive design, is to create it yourself.

- Another way to create a responsive design, is to use an already existing CSS framework.
 - Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive webs.
 - <http://getbootstrap.com/>
 - Bootstrap helps you to develop sites that look nice at any size; screen, laptop, tablet, or phone.

Literature

- W3 Schools
<http://www.w3schools.com/>
- W3 Schools --- HTML Tag Reference
<http://www.w3schools.com/tags/>
- Notepad++
<http://notepad-plus-plus.org/>
- Bootstrap --- HTML, CSS, and JS framework
<http://getbootstrap.com/>
- W3 Schools --- Bootstrap --- Tutorial
<http://www.w3schools.com/bootstrap/>

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. Images
16. Tables
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive Web Design
22. **Forms**
23. Iframes
24. Color Names
25. Color Values
26. Color Shades
27. JavaScript
28. Head
29. Entities
30. Symbols
31. Charset
32. URL Encode
33. XHTML

22. HTML Forms and Input

■ HTML Forms

- HTML forms are used to select different kinds of user input.
- HTML forms are used to pass data to a server.
- HTML forms can contain input elements like
 - text fields, checkboxes, radio-buttons, submit buttons, select lists,
 - textarea, fieldset, legend, and label elements.
- The `<form>` tag is used to create an HTML form:

```
<form>
```

```
.
```

```
input elements
```

```
.
```

```
</form>
```

22. HTML Forms and Input

■ The Input Element

- The most important form element is the `<input>` element.
- The `<input>` element is used to select user information.
- An `<input>` element can vary in many ways, depending on the type attribute.
- An `<input>` element can be of type
 - text field, checkbox, radio-button, submit button, ...

■ Text Fields

- `<input type="text">` defines a one-line input field that a user can enter text into.
- The default width of a text field is 20 characters.

```
<form>
```

```
First name: <input type="text" name="firstname"><br>
```

```
Last name: <input type="text" name="lastname">
```

```
</form>
```

22. HTML Forms and Input

■ Password Fields

- `<input type="password">` defines a password field.
- The characters in a password field are masked (shown as asterisks or circles).

```
<form>  
Password: <input type="password" name="pwd">  
</form>
```

■ Radio Buttons

- `<input type="radio">` defines a radio button.
- Radio buttons let a user select ONLY ONE of a limited number of choices.

```
<form>  
<input type="radio" name="sex" value="male">Male<br>  
<input type="radio" name="sex" value="female">Female  
</form>
```

22. HTML Forms and Input

■ Checkboxes

- `<input type="checkbox">` defines a checkbox.
- Checkboxes let a user select ZERO or MORE options of a limited number of choices.

```
<form>
```

```
<input type="checkbox" name="vehicle" value="Bike">I have a bike<br>
```

```
<input type="checkbox" name="vehicle" value="Car">I have a car
```

```
</form>
```

22. HTML Forms and Input

▪ Submit Buttons

- `<input type="submit">` defines a submit button.
- A submit button is used to send form data to a server.
- The data is sent to the page specified in the form's action attribute.
- The file defined in the action attribute usually does something with the received input.
- If you click the "Submit" button, the browser will send your input to a page called "demo.php".
- The page demo.php will show you the received input.

```
<form name="input" action="demo.php" method="get">  
Username: <input type="text" name="user">  
<input type="submit" value="Submit">  
</form>
```

22. HTML Forms and Input

- Drop-Down-List with / without preselected value

```
<form name="input" action="demo-2.php" method="get">  
<select name="cars">  
  <option value="volvo">Volvo</option>  
  <option value="saab">Saab</option>  
  <option value="fiat" selected>Fiat</option>  
  <option value="audi">Audi</option>  
</select>  
</form>
```


22. HTML Forms and Input

■ Textarea

- How to create a multi-line text input control.
- In a text-area the user can write an unlimited number of characters.

```
<form name="input" action="demo-2.php" method="get">  
<textarea rows="10" cols="30">  
    Put your default text here ...  
</textarea>  
</form>
```

22. HTML Forms and Input

▪ Buttons

- Just have a button to click.

```
<form name="input" action="demo-3.php" method="get">  
<input type="button" value="Hello world!">  
</form>
```

▪ Fieldsets

- How to create a border around elements in a form.

```
<form name="input" action="demo-3.php" method="get">  
<fieldset>  
  <legend>Personal information:</legend>  
  Name: <input type="text" size="30"><br>  
  E-mail: <input type="text" size="30"><br>  
  Date of birth: <input type="text" size="10">  
</fieldset>  
</form>
```

22. HTML Forms and Input

- **Send an eMail from a form**
 - Just have a button to click.

```
<form action="MAILTO:you@example.de" method="post" enctype="text/plain">  
  Name: <input type="text" name="name" value="your name"><br>  
  E-mail: <input type="text" name="mail" value="your email"><br>  
  Comment: <input type="text" name="comment" value="" size="50"><br>  
  <input type="submit" value="Send">  
  <input type="reset" value="Reset">  
</form>
```

Literature

- W3 Schools
<http://www.w3schools.com/>
- PHP --- Vordefinierte Variablen (GET & POST)
<http://php.net/manual/de/reserved.variables.php>

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Content

1. Introduction
2. Editors
3. Basic Example
4. Elements
5. Attributes
6. Head
7. Paragraphs
8. Formatting
9. Styles
10. Quotations
11. Computercode
12. Comments
13. CSS / CSS3
14. Links
15. Images
16. Tables
17. Lists
18. Blocks
19. Classes
20. Layout
21. Responsive Web Design
22. Forms
23. **Iframes**
24. **Color Names**
25. **Color Values**
26. **Color Shades**
27. **JavaScript**
28. **Head**
29. **Entities**
30. **Symbols**
31. **Charset**
32. **URL Encode**
33. **XHTML**

Remainder

- **The following chapters are not part of this lecture!**

- 23. Iframes
- 24. Color Names
- 25. Color Values
- 26. Color Shades
- 27. JavaScript

28. HTML Head

- HTML head Elements

Tag	Description
<head>	Defines information about the document
<title>	Defines the title of a document
<base>	Defines a default address or a default target for all links on a page
<link>	Defines the relationship between a document and an external resource
<meta>	Defines metadata about an HTML document
<script>	Defines a client-side script
<style>	Defines style information for a document

28. HTML Head

■ The HTML `<head>` Element


- The `<head>` element is a container for all the head elements.
- Elements inside `<head>` can include scripts, instruct the browser where to find style sheets, provide meta information, and more.
- The following tags can be added to the head section:
 - `<title>`,
 - `<style>`,
 - `<meta>`,
 - `<link>`,
 - `<script>`,
 - `<noscript>`, and
 - `<base>`.

28. HTML Head

▪ The HTML <title> Element

- The <title> tag defines the title of the document.
- The <title> element is required in all HTML/XHTML documents.
- The <title> element:
 - defines a title in the browser toolbar
 - provides a title for the page when it is added to favorites
 - displays a title for the page in search engine results
- Example

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Title of the document</title>  
  </head>  
  ...
```



28. HTML Head

- The HTML `<base>` Element

- The `<base>` tag specifies the base URL/target for all relative URLs in a page.
- Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
    <base href="http://www.hof-university.de/" target="_blank">
  </head>
  ...
```



28. HTML Head

▪ The HTML <link> Element

- The <link> tag defines the relationship between a document and an external resource.
- The <link> tag is most used to link to style sheets:
- Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
    <base href="http://www.hof-university.de/" target="_blank">
    <link rel="stylesheet" type="text/css" href="mystyle-1.css">
    <link rel="stylesheet" type="text/css" href="mystyle-2.css">
  </head>
  ...
```



28. HTML Head

- The HTML `<style>` Element

- The `<style>` tag is used to define style information for an HTML document.
- Inside the `<style>` element you specify how HTML elements should render in a browser.

- Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
    <style>
      body {background-color:yellow;}
      p    {color:blue;}
    </style>
  </head>
  ...
```



28. HTML Head

▪ The HTML <meta> Element

- Metadata is data (information) about data.
- The <meta> tag provides metadata about the HTML document.
- Metadata will not be displayed on the page, but will be machine parsable.
- Meta elements are typically used to specify page description, keywords, author of the document, last modified, and other metadata.
- The metadata can be used by browsers (how to display content or reload page), search engines (keywords), or other web services.
- <meta> tags always go inside the <head> element.
- Example (Refresh page every 30 seconds)

<head>

<meta name="keywords" content="HTML, CSS, XML, XHTML, JavaScript">

<meta name="description" content="Free Web tutorials ...">

<meta name="author" content="Hans Wurscht">

<meta http-equiv="refresh" content="30">

</head>



28. HTML Head

▪ The HTML `<script>` Element

- The `<script>` tag is used to define a client-side JavaScript.
- The script below writes Hello World! into an HTML element with `id="demo"`.



```
<head>  
<script>  
function myFunction {  
    document.getElementById("demo").innerHTML = "Hello World!";  
}  
</script>  
</head>
```

29. HTML Entities

■ HTML Entities

- Some characters are reserved in HTML.
- If you use the less than (<) or greater than (>) signs in your text, the browser might mix them with tags.
- Character entities are used to display reserved characters in HTML.
- To display a less than sign we must write: **<** or **<**;
- A common character entity used in HTML is the non breaking space (** **).
- A general character entity looks like this:

`&entity_name;`

OR

`&#entity_number;`

29. HTML Entities

Result	Description	Entity Name	Entity Number
	non-breaking space	 	
<	less than	<	<
>	greater than	>	>
&	ampersand	&	&
¢	cent	¢	¢
£	pound	£	£
¥	yen	¥	¥
€	euro	€	€
©	copyright	©	©
®	registered trademark	®	®

29. HTML Entities

Mark	Character	Construct	Result
`	a	à	à
´	a	á	á
^	a	â	â
~	a	ã	ã
`	O	Ò	Ò
´	O	Ó	Ó
^	O	Ô	Ô
~	O	Õ	Õ

30. HTML Symbols

■ HTML Symbol Entities

- Many mathematical, technical, and currency symbols, are not present on a normal keyboard.
- To add these symbols to an HTML page, you can use an HTML entity name.
- If no entity name exists, you can use an entity number; a decimal (or hexadecimal) reference.
- If you use an HTML entity name or a hexadecimal number, the character will always display correctly. This is independent of what character set (encoding) your page uses!

```
<p>I will display &euro;</p>  
<p>I will display &#8364;</p>  
<p>I will display &#x20AC;</p>
```

30. HTML Symbols

- Some Mathematical Symbols Supported by HTML

Char	Number	Entity	Description
\forall	∀	∀	FOR ALL
∂	∂	∂	PARTIAL DIFFERENTIAL
\exists	∃	∃	THERE EXISTS
\emptyset	∅	∅	EMPTY SETS
∇	∇	∇	NABLA
\in	∈	∈	ELEMENT OF
\notin	∉	∉	NOT AN ELEMENT OF
\ni	∋	∋	CONTAINS AS MEMBER
\prod	∏	∏	N-ARY PRODUCT
\sum	∑	∑	N-ARY SUMMATION

30. HTML Symbols

- Some Greek Letters Supported by HTML

Char	Number	Entity	Description
A	Α	Α	GREEK CAPITAL LETTER ALPHA
B	Β	Β	GREEK CAPITAL LETTER BETA
Γ	Γ	Γ	GREEK CAPITAL LETTER GAMMA
Δ	Δ	Δ	GREEK CAPITAL LETTER DELTA
E	Ε	Ε	GREEK CAPITAL LETTER EPSILON
Z	Ζ	Ζ	GREEK CAPITAL LETTER ZETA

31. HTML Encoding (Character Sets)

- **What is Character Encoding?**
 - To display an HTML page correctly, a web browser must know the character set (character encoding) to use.
 - ASCII supported numbers (0-9), English letters (A-Z), and some special characters like ! \$ + - () @ < > .
 - ANSI (Windows-1252) was the original Windows character set. It supported 256 different character codes.
 - ISO-8859-1 was the default character set for HTML 4. It also supported 256 different character codes.
 - Because ANSI and ISO was limited, the default character encoding was changed to UTF-8 in HTML5.
 - UTF-8 (Unicode) covers almost all of the characters and symbols in the world.
 - All HTML 4 processors also support UTF-8.
 - If a browser detects ISO-8859-1 in a web page, it defaults to ANSI, because ANSI is identical to ISO-8859-1 except that ANSI has 32 extra characters.

31. HTML Encoding (Character Sets)

■ Examples

- HTML 4

```
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

- HTML 5

```
<meta charset="UTF-8">
```

32. HTML URL Encoding

- **HTML Uniform Resource Locators (URLs)**
 - A URL is another word for a web address.
 - A URL can be composed of words (w3schools.com), or an Internet Protocol (IP) address (192.68.20.50).
 - Web browsers request pages from web servers by using a URL.
 - A Uniform Resource Locator (URL) is used to address a document (or other data) on the web.
 - Syntax: **scheme://host.domain:port/path/filename**
 - **scheme** - defines the **type** of Internet service (most common is **http**)
 - **host** - defines the **domain host** (default host for http is **www**)
 - **domain** - defines the Internet **domain name** (w3schools.com)
 - **port** - defines the **port number** at the host (default for http is **80**)
 - **path** - defines a **path** at the server (If omitted: the root directory of the site)
 - **filename** - defines the name of a document or resource

32. HTML URL Encoding

■ URL Encoding

- URLs can only be sent over the Internet using the ASCII character-set.
- Since URLs often contain characters outside the ASCII set, the URL has to be converted into ASCII.
- URL encoding converts characters into a format that can be transmitted over the Internet.
- URL encoding replaces non ASCII characters with a "%" followed by two hexadecimal digits.
- URLs cannot contain spaces. URL encoding normally replaces a space with a + sign.
- URL Encoding Reference

http://www.w3schools.com/tags/ref_urlencode.asp

33. HTML and XHTML

■ What Is XHTML?

- XHTML stands for **EX**tensible **HyperText Markup Language**
- XHTML is almost identical to HTML
- XHTML is stricter than HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers
- Validate XHTML With The W3C Validator: <http://validator.w3.org/>

■ The Most Important Differences from HTML

- Document Structure
 - XHTML DOCTYPE is mandatory
 - The xmlns attribute in <html> is mandatory
 - <html>, <head>, <title>, and <body> are mandatory

33. HTML and XHTML

■ The Most Important Differences from HTML

- XHTML Elements
 - XHTML elements must be **properly nested**
 - XHTML elements must always be **closed**
 - XHTML empty elements must always be **closed**
 - XHTML elements must be in **lowercase**
 - XHTML documents must have **one root element**
- XHTML Attributes
 - Attribute names must be in **lower case**
 - Attribute values must be **quoted**
 - Attribute minimization is **forbidden**

wrong: `<option selected>`

correct: `<option selected="selected">`

Literature

- W3 Schools
<http://www.w3schools.com/>
- URL Encoding Reference
http://www.w3schools.com/tags/ref_urlencode.asp

HTML / HTML 5

Part 1 --- HTML

Prof. Dr. Jürgen Heym

Hochschule Hof

Was ist neu in HTML 5 ?

- DOCTYPE für HTML 5 ist html

```
<!DOCTYPE html>
```

- Default Character Encoding in HTML 5 ist UTF8

```
<meta charset="UTF-8">
```

- Beispiel

```
<!DOCTYPE html>  
<html>  
<head>  
<meta charset="UTF-8">  
<title>Title of the document</title>  
</head>
```

Was ist neu in HTML 5 ?

- Nachfolgende HTML-Tags gibt es in HTML 5 nicht mehr.

```
<acronym>  
<applet>  
<basefont>  
<big>  
<center>  
<dir>  
<font>  
<frame>  
<frameset>  
<noframes>  
<strike>  
<tt>
```

Was ist neu in HTML 5 ?

- In HTML 5 neue HTML-Tags

- Semantische Elemente

- `<header>`
 - `<footer>`
 - `<article>`
 - `<section>`
 - etc.

- Formulkontrollattribute

- `number`, `date`, `time`, `range`, etc.

- Graphische Elemente

- `<svg>`
 - `<canvas>`

- Multimedia-Elemente

- `<audio>`
 - `<video>`
 - etc.

Was ist neu in HTML 5 ?

- HTML 5 APIs (JavaScript)
 - HTML Geolocation
 - HTML Drag and Drop
 - HTML Local Storage (anstatt Cookies!)
 - HTML Application Cache
 - HTML Web Workers
 - HTML Server Sent Events (HTML SSE)

Neue semantische Elemente in HTML 5

Tag	Beschreibung
<code><article></code>	Artikel
<code><aside></code>	Inhalt neben dem Seiteninhalt
<code><figure></code>	Bild
<code><footer></code>	Fußbereich
<code><header></code>	Kopfbereich
<code><main></code>	Hauptinhalt
<code><nav></code>	Navigation
<code><section></code>	Abschnitt



Neue Strukturelemente in HTML 5

Tag	Beschreibung
<bdi>	Textrichtung
<details>	Details
<dialog>	Dialogbox oder –fenster
<figcaption>	Über-/Unterschrift für ein Bild
<mark>	hervorgehobener Bereich
<menuitem>	Menüpunkt in einem Menü
<meter>	Meßzahl innerhalb eines definierten Bereichs
<progress>	Fortschritt einer Aufgabe
<rp>	Ersatzanzeige, falls Ruby-Annotations nicht möglich.
<ruby>	Ruby Annotation (Ostasiatische Typographie)
<summary>	Sichtbarer Kopfbereich für Details
<time>	Datum / Zeit
<wbr>	Möglicher Zeilenumbruch

Neue Formelemente und -attribute in HTML 5

Tag	Beschreibung
<datalist>	Vordefinierte Optionen für Eingabefelder
<keygen>	Definiert eine Schlüssel-Wert-Paar-Generatorfeld
<output>	Ergebnis einer Berechnung

Attribut	Attribut
color	range
date	search
datetime	tel
datetime-local	time
email	url
month	week
number	

Neue Mediaelemente in HTML 5

Tag	Beschreibung
<audio>	Musik, Sound, etc.
<embed>	Container für externe Anwendungen
<source>	Quelle für <audio> bzw. <video>
<track>	Spur für <audio> bzw. <video>
<video>	Video- oder Filminhalt

Neue Attributsyntax in HTML 5

Typ	Beispiel
Empty	<code><input type="text" value="John" disabled></code>
Unquoted	<code><input type="text" value=John></code>
Double-quoted	<code><input type="text" value="John Doe"></code>
Single-quoted	<code><input type="text" value='John Doe'></code>

HTML 5 Skelett --- Teil 1

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>HTML5 Skeleton</title>

<meta charset="utf-8">

<!--[if lt IE 9]>
    ...
<![endif]-->

<style>
    body {font-family: Verdana, sans-serif; font-size:0.8em;}
    header,nav, section,article,footer
        {border:1px solid grey; margin:5px; padding:8px;}
    nav ul {margin:0; padding:0;}
    nav ul li {display:inline; margin:5px;}
</style>
</head>
```

HTML 5 Skelett --- Teil 2

...

```
<body>
```

```
<header>
```

```
  <h1>HTML5 SKeleton</h1>
```

```
</header>
```

```
<nav>
```

```
<ul>
```

```
  <li><a href="html5_semantic_elements.asp">HTML5 Semantic</a></li>
```

```
  <li><a href="html5_geolocation.asp">HTML5 Geolocation</a></li>
```

```
  <li><a href="html5_canvas.asp">HTML5 Graphics</a></li>
```

```
</ul>
```

```
</nav>
```


HTML 5 Skelett --- Teil 3

...

```
<section>
```

```
<h1>Famous Cities</h1>
```

```
<article>
```

```
<h2>London</h2>
```

```
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>
```

```
</article>
```

```
<article>
```

```
<h2>Paris</h2>
```

```
<p>Paris is the capital and most populous city of France.</p>
```

```
<figure>
```

```
  
```

```
  <figcaption>Fig.1 - The Pulpit Rock, Norway.</figcaption>
```

```
</figure>
```

```
</article>
```

```
</section>
```

HTML 5 Skelett --- Teil 4

...

```
<footer>  
<p>&copy; 2014 W3Schools. All rights reserved.</p>  
</footer>  
  
<aside>  
  <h4>Epcot Center</h4>  
  <p>The Epcot Center is a theme park in Disney World, Florida.</p>  
</aside>  
  
</body>  
</html>
```

HTML 5 CANVAS

- Mit dem HTML 5 Canvas-Element kann man Graphiken auf einer Webseite zeichnen.
 - Das Canvas-Element ist nur ein Kontainer!
 - Sie müssen eine Scriptsprache nutzen, um eine Graphik zu zeichnen.
 - Beispiel (Javascript)

```
<!DOCTYPE html>
<html>
<body>
<canvas id="myCanvas" width="200" height="100" style="border:1px solid #c3c3c3;">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
    var c = document.getElementById("myCanvas");
    var ctx = c.getContext("2d");
    ctx.fillStyle = "#FF0000";
    ctx.fillRect(0,0,150,75);
</script>
</body>
</html>
```

HTML 5 Scalable Vector Graphics (SVG)

- Mit dem HTML 5 SVG-Element kann man skalierbare Vektorgraphiken auf einer Webseite zeichnen.
 - Das SVG-Element ist nur ein Container!
 - Das SVG-Element kennt mehrere Methoden, um verschiedene Elemente zu zeichnen.
 - Beispiel (Javascript)

```
<!DOCTYPE html>
<html>
<body>
<svg width="100" height="100">
  <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" />
  <rect width="400" height="100" style="fill:rgb(0,0,255);stroke-
                                width:10;stroke:rgb(0,0,0)" />
  <polygon points="100,10 40,198 190,78 10,78 160,198"
           style="fill:lime;stroke:purple;stroke-width:5;fill-rule:evenodd;" />
</svg>

</body>
</html>
```

HTML 5 Multimedia

- HTML 5 unterstützt derzeit nur die Video-Formate MP4, WebM und Ogg!
 - MP4 wurde durch die Moving Pictures Expert Group entwickelt, basiert auf QuickTime und wird in neueren Video-Kameras und TV-Hardware genutzt. MP4 ist empfohlenes Youtube-Format.
 - Theora Ogg wurde durch die Xiph.Org Foundation entwickelt.
 - WebM wurde von den Web-Giganten Mozilla, Opera, Adobe und Google entwickelt.

- HTML 5 unterstützt derzeit nur die Audio-Formate MP3, WAV und Ogg!
 - MP3-Dateien sind der Ton in MPEG-Dateien. MP3 ist das derzeit populärste Format für Music Players und kombiniert gute Kompression bei hoher Qualität.
 - Ogg wurde durch die Xiph.Org Foundation entwickelt.
 - WAV wurde von IBM und Microsoft und ist optimiert auf Windows-, Mac- und Linux-Betriebssystemen.

HTML 5 Video-Element

```
<video width="320" height="240" controls autoplay>  
  <source src="movie.mp4" type="video/mp4">  
  <source src="movie.ogv" type="video/ogg">  
  Your browser does not support the video tag.  
</video>
```

- Das controls –Attribut fügt die Steuerelemente für den Videoplayer hinzu.
- Die Attribute für Breite und Höhe sind sinnvoll, damit die Seite beim Laden nicht flackert, weil die Breite und Höhe noch nicht bekannt sind.
- Die erste durch den Browser unterstützte Datenquelle wird genutzt.
- Das autoplay-Attribut funktioniert nicht auf allen Geräten!

- HTML5 definiert DOM-Methoden, -Eigenschaften und –Ereignisse für das Video-Element. Diese erlauben sowohl die Steuerung des Videos, als auch das Setzen der Spielzeit und der Lautstärke.

HTML 5 Audio-Element

```
<audio controls>  
  <source src="horse.ogg" type="audio/ogg">  
  <source src="horse.mp3" type="audio/mpeg">  
  Your browser does not support the audio element.  
</audio>
```

- Das controls –Attribut fügt die Steuerelemente für den Audioplayer hinzu.
- Die erste durch den Browser unterstützte Datenquelle wird genutzt.
- HTML5 definiert DOM-Methoden, -Eigenschaften und –Ereignisse für das Audio-Element. Diese erlauben sowohl die Steuerung des Audios, als auch das Setzen der Spielzeit und der Lautstärke.

Youtube Videos

```
<iframe width="420" height="315"  
src="http://www.youtube.com/embed/XGSy3_Czz8k?autoplay=1">  
</iframe>
```

1. Laden Sie das Video auf Youtube.
2. Notieren Sie die Video-ID.
3. Definieren Sie ein <iframe>-Element auf ihrer Webseite.
4. Das src-Attribute muss auf das Video bei Youtube zeigen.
5. Mit der Breite und Höhe konfigurieren Sie den Player.
6. Weitere Kontrollparameter beeinflussen den Player:

autoplay = 0, 1, 2 (default = versteckte Kontrollelemente automatisch)

controls = 0 (default), 1 (autoplay)

loop = 0, 1 (default), 2

playlist = 0 (default), 1

playlist = komma-separierte Liste zusätzlicher Video-URLs

Literature

- HTML 5 Introduction
http://www.w3schools.com/html/html5_intro.asp
- HTML 5 Graphics
http://www.w3schools.com/html/html5_canvas.asp
- HTML Media
http://www.w3schools.com/html/html_media.asp

Internettechniken

HTML DOM

Prof. Dr. Jürgen Heym

Hochschule Hof

HTML DOM

W3C Document Object Model (DOM)

- Das Document Object Model (DOM) ist ein W3C-Standard.
- Das W3C-DOM definiert einen Standard für den Zugriff auf die Elemente eines Dokuments und deren Manipulation.

"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."

- Das DOM ist in drei Teile/Level aufgeteilt:
 - Core DOM Standardmodell für strukturierte Dokumente
 - XML DOM Standardmodell für XML-Dokumente
 - HTML DOM Standardmodell für HTML-Dokumente
- Das DOM definiert die Objekte und Eigenschaften aller Elemente des Dokuments und Methode, diese zu manipulieren.

HTML DOM

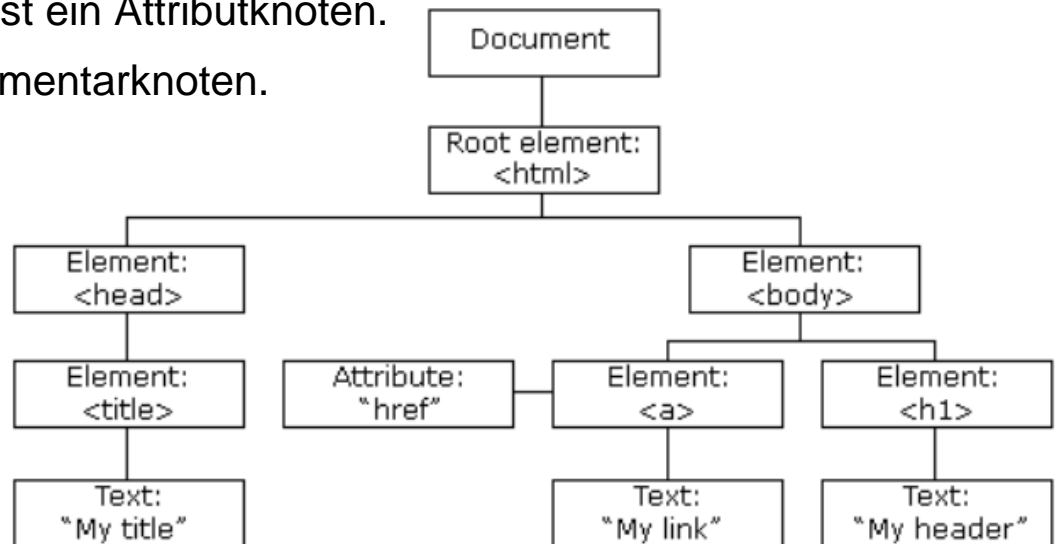
HTML Document Object Model (DOM)

- Das HTML DOM ist
 - ein Standard-Objektmodell für HTML-Dokumente,
 - eine Standard-Programmierschnittstelle für HTML-Dokumente ,
 - Unabhängig von der Plattform und der Programmiersprache und
 - ein W3C-Standard.
- Das HTML DOM definiert die Objekte und Eigenschaften aller HTML-Elemente des Dokuments und Methoden, um auf diese zuzugreifen.
- Das HTML DOM ist ein Standard um HTML-Elemente zu lesen, ändern, hinzuzufügen und zu löschen.

HTML DOM

HTML DOM Knoten

- Im HTML DOM dreht sich alles um Knoten:
 - Das Gesamtdokument ist ein Dokumentenknoten.
 - Jedes HTML-Element ist ein Elementknoten.
 - Die Texte in HTML-Elementen sind Textknoten.
 - Jedes HTML-Attribut ist ein Attributknoten.
 - Kommentare sind Kommentarknoten.



HTML DOM

HTML DOM Beispiel

```
<html>  
  <head>  
    <title>DOM Tutorial</title>  
  </head>  
  <body>  
    <h1>DOM Lesson one</h1>  
    <p>Hello world!</p>  
  </body>  
</html>
```

- Der Wurzelknoten ist durch das html-Tag bestimmt.
- Alle anderen Knoten sind innerhalb des html-Tags.
- Der html-Knoten hat zwei Kinderknoten head und body.
- Der head-Knoten enthält einen title-Knoten.
- Der body-Knoten enthält einen h1- und einen p-Knoten.

HTML DOM

HTML DOM Beispiel

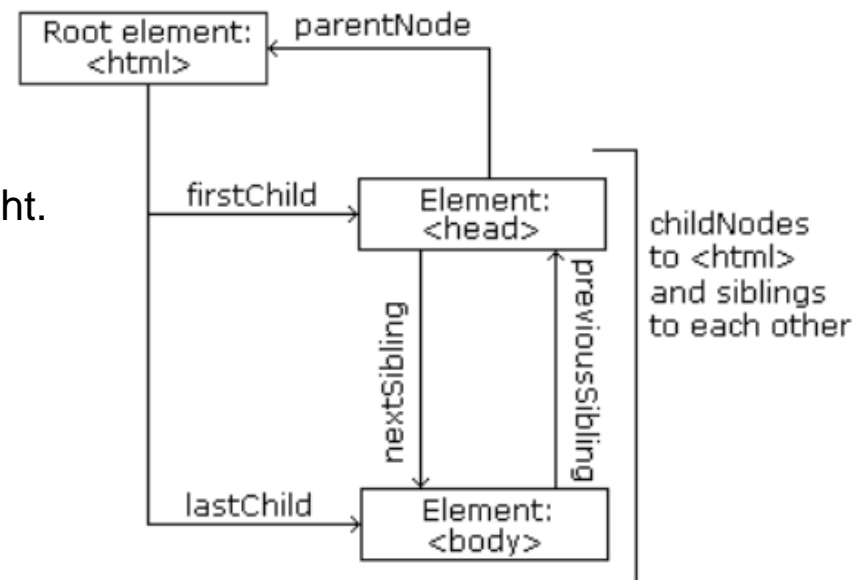
```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

- Ein Elementknoten enthält nie Text! Text wird immer in Textknoten abgelegt!
- In unserem Beispiel `<title>DOM Tutorial</title>` enthält der Elementknoten title einen Textknoten mit dem Wert "DOM Tutorial".
- "DOM Tutorial" ist nicht der Wert des title-Knotens!
- Im HTML DOM kann der Wert des Textknotens über die Eigenschaft `innerHTML` manipuliert werden.

HTML DOM

HTML DOM Knotenbaum

- Die Knoten in einem DOM-Knotenbaum haben eine hierarchische Beziehung zueinander.
- Man spricht von Eltern, Kindern und Geschwistern, um die Beziehung der Knoten zueinander zu beschreiben.
- Der Wurzelknoten wird auch als “top node” oder “root” bezeichnet.
- Jeder Knoten hat exakt einen Elternknoten, nur der Root-Knoten nicht.
- Jeder Knoten kann eine beliebige Anzahl von Kindknoten haben.
- Einen Knoten ohne Kindknoten bezeichnet man als Blatt (leaf).
- Geschwister sind Knoten mit gleichem Elternknoten.



HTML DOM

HTML DOM Eigenschaften und Methoden

- HTML DOM Eigenschaften
 - `x.innerHTML` Wert des Textknoten innerhalb eines Elementes `x`.
 - `x.nodeName` Knotenname
 - `x.nodeValue` Wert oder Inhalt des Knotens
 - `x.parentNode` Elternknoten zu Element `x`
 - `x.childNodes` Kindknoten zu Element `x`
 - `x.attributes` Attribute des Elements `x`
 - `x.firstChild` Erster Kindknoten des Elements `x`
 - `x.lastChild` Letzter Kindknoten des Elements `x`

- HTML DOM Methoden
 - `x.getElementById(id)` Element mittels seiner *id* als Objekt auslesen.
 - `x.getElementsByTagName(name)` Alle Element einer bestimmten Tag-Klasse auslesen.
 - `x.appendChild(node)` Einen neuen Kindknoten einfügen.
 - `x.removeChild(node)` Einen Kindknoten entfernen.

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel: innerHTML-Eigenschaft

```
<html>
<body>

<p id="intro">Hello World!</p>

<script type="text/javascript">

    txt=document.getElementById("intro").innerHTML;

    document.write("<p>The text from the intro paragraph: " + txt + "</p>");

</script>

</body>
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel: Auslesen aller p-Tag-Inhalte

...

```
x=document.getElementsByTagName("p");
```

```
for (i=0;i<x.length;i++)  
{  
    document.write(x[i].innerHTML);  
    document.write("<br />");  
}
```

...

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel: Eigenschaften firstChild und lastChild

```
<html>
```

```
<body>
```

```
<p id="intro">Hello World!</p>
```

```
<script type="text/javascript">
```

```
    x=document.getElementById("intro");
```

```
    document.write(x.firstChild.nodeValue);
```

```
</script>
```

```
</body>
```

```
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Spezielle Knoten
 - **document.documentElement**
referenziert den Root-Knoten des Dokuments.
 - **document.body**
referenziert das body-Tag eines Dokuments.

HTML DOM

HTML DOM Eigenschaften und Methoden

- Eigenschaften aller Knoten
 - Im HTML DOM ist jeder Knoten ein Objekt.
 - Objekte haben Methoden und Eigenschaften, die mittels JavaScript manipuliert werden können.
 - Drei wichtige Eigenschaften jedes Knotens sind:
 - **nodeName**
 - **nodeValue**
 - **nodeType**

- nodeName-Eigenschaft
 - Die nodeName-Eigenschaft bestimmt den Namen des Knotens.
 - Der nodeName ist nur lesbar (read-only).
 - Der nodeName eines Elements ist identisch zum Tag-Namen.
 - Der nodeName eines Attributs ist identisch zum Attributnamen.
 - Der nodeName eines Textknotens ist immer #text.
 - Der nodeName des Gesamtdokuments ist immer #document.

HTML DOM

HTML DOM Eigenschaften und Methoden

- nodeValue-Eigenschaft
 - Die nodeValue-Eigenschaft bestimmt den Inhalt des Knotens.
 - Die nodeValue-Eigenschaft ist für Elementknoten nicht definiert.
 - Die nodeValue-Eigenschaft eines Textknotens ist der Text selbst.
 - Die nodeValue-Eigenschaft für Attributknoten ist der Attributwert.

HTML DOM

HTML DOM Eigenschaften und Methoden

- nodeType-Eigenschaft
 - Die nodeType-Eigenschaft kann nur gelesen werden und bestimmt den Knotentyp.

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel
 - Änderung der Hintergrundfarbe des body-Tags.

```
<html>  
<body>  
  
  <script type="text/javascript">  
    document.body.backgroundColor="lavender";  
  </script>  
  
</body>  
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel
 - Änderung des Inhalts des Elementes „p1“.

```
<html>
```

```
<body>
```

```
<p id="p1">Hello World!</p>
```

```
<script type="text/javascript">
```

```
    document.getElementById("p1").innerHTML="New text!";
```

```
</script>
```

```
</body>
```

```
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel
 - Änderung einer Eigenschaft bei Mausklick.

```
<html>
```

```
<body>
```

```
<input type="button"  
      onclick="document.body.bgColor='lavender' ;"  
      value="Change background color" />
```

```
</body>
```

```
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel
 - Änderung des Inhalts des Elementes „p1“ über sein style-Objekt.

```
<html>
<head>
<script type="text/javascript">
function ChangeBackground()
{
    document.body.style.backgroundColor="lavender";
}
</script>
</head>

<body>
<input type="button" onclick="ChangeBackground() "
    value="Change background color" />
</body>
</html>
```

HTML DOM

HTML DOM Eigenschaften und Methoden

- Beispiel
 - Änderung von Font und Farbe es Elements „p1“.

```
<html>
<head>
<script type="text/javascript">
function ChangeStyle()
{
    document.getElementById("p1").style.color="blue";
    document.getElementById("p1").style.fontFamily="Arial";
}
</script>
</head>

<body>
<p id="p1">Hello world!</p>
<input type="button" onclick="ChangeStyle() "
    value="Change style" />
</body>
</html>
```

HTML DOM

HTML DOM Ereignisse (events)

- Ereignisse (events)

Jedes Element einer Webseite hat bestimmte Möglichkeiten Ereignisse an JavaScript-Funktionen weiterzuleiten:

- Mausklick
- Tastendruck
- Laden einer neuen Seite oder eines Bildes.
- Bewegung der Maus über einen Hotspot.
- Auswahl eines Eingabefeldes.
- Abschicken eines Formulars.

Beispiel: E-mail: `<input type="text" id="email" onchange="checkEmail()" />`

Referenz: Siehe Vorlesung “Javascript 1”.

HTML DOM

Beispiele

- Beispiel

Wie viele Anker enthält ein Dokument? Antwort: Nur Anker mit name-Attribut werden gezählt!

```
<html>
<body>
<a name="html">HTML Tutorial</a><br />
<a name="css">CSS Tutorial</a><br />
<a name="xml">XML Tutorial</a><br />
<a href="/js/">JavaScript Tutorial</a>

<p>Number of anchors:
<script type="text/javascript">
    document.write(document.anchors.length);
</script>
</p>
</body>
</html>
```

HTML DOM

Beispiele

- Beispiel

Rückgabe der Eigenschaft innerHTML des ersten Ankers.

```
<html>
<body>

<a id="html">HTML Tutorial</a><br />
<a id="css">CSS Tutorial</a><br />
<a id="xml">XML Tutorial</a>

<p>innerHTML of the first anchor:
<script type="text/javascript">
    document.write(document.anchors[0].innerHTML);
</script>
</p>

</body>
</html>
```

HTML DOM

Beispiele

- Beispiel

Wie viele Formulare enthält das aktuelle Dokument?

```
<html>
<body>

<form name="Form1"></form>
<form name="Form2"></form>
<form></form>

<p>Number of forms:
<script type="text/javascript">
    document.write (document.forms.length) ;
</script>
</p>

</body>
</html>
```

HTML DOM

Beispiele

- Beispiel

Wie heißt das erste Formular?

```
<html>
<body>

<form name="Form1"></form>
<form name="Form2"></form>
<form></form>

<p>Name of first form:
<script type="text/javascript">
document.write (document.forms [0] .name) ;
</script>
</p>

</body>
</html>
```

HTML DOM

Beispiele

- Beispiel

Wie viele Bilder sind im dokument referenziert?

```
<html>
<body>




<p>Number of images:
<script type="text/javascript">
    document.write(document.images.length) ;
</script>
</p>

</body>
</html>
```

siehe: http://www.w3schools.com/jsref/tryit.asp?filename=tryjsref_doc_images

HTML DOM

Beispiele

- Beispiel

Welche Cookies sind gesetzt?

```
<html>
```

```
<body>
```

Cookies associated with this document:

```
<script type="text/javascript">  
    document.write(document.cookie) ;  
</script>
```

```
</body>
```

```
</html>
```

HTML DOM

Beispiele

- Beispiel

Öffnen eines neuen Fensters und einfügen von Text in das neue HTML DOM.

```
<html>
<body>

<script type="text/javascript">
    var w=window.open();
    w.document.open();
    w.document.write("<h1>Hello World!</h1>");
    w.document.close();
</script>

</body>
</html>
```

HTML DOM

Beispiele

- ... und noch viel mehr Beispiele finden Sie hier:

`http://www.w3schools.com/jsref/dom_obj_document.asp`

Internettechniken

JavaScript Teil 1

Prof. Dr. Jürgen Heym

Hochschule Hof

JavaScript

Einleitung

- JavaScript ist eine der wichtigsten Script-Sprachen im Web.
- JavaScript wird auf unzähligen Webseiten für verschiedene Aufgaben eingesetzt:
 - Dynamische Funktionen
 - Formularvalidierung
 - Kommunikation zu einem Server

JavaScript

Einleitung

Was ist JavaScript?

- JavaScript wurde entwickelt, um HTML-Seiten Interaktivität hinzuzufügen.
- JavaScript ist eine Skriptsprache.
- JavaScript ist eine Leichtgewicht-Programmiersprache.
- JavaScript wird in HTML eingebettet.
- JavaScript wird von einem Interpreter ausgeführt, es gibt keinen JavaScript-Compiler.
- JavaScript ist lizenzfrei und kann von jedem eingesetzt werden.

JavaScript

Einleitung

JavaScript == ECMAScript

- JavaScript ist eine Implementation des ECMA Skriptsprachen Standards.
- *ECMA == European Computer Manufacturers Association.*
- JavaScript entspricht dem Standard ECMA-262.
- JavaScript wurde von Brendan Eich in Netscape Navigator 2.0 erstmals eingeführt und ist seit 1996 in allen Browsern verfügbar.
- Der offizielle ECMAScript-262 Standard wurde 1997 verabschiedet und 1998 als ISO 16262 übernommen.
- JavaScript befindet sich immer noch in der Weiterentwicklung.

JavaScript

Beispiele

Beispiel 1: Text in ein HTML-Dokument schreiben

```
<html>
<body>

<h1>My First Web Page</h1>

  <script type="text/javascript">
    document.write("<p>" + Date() + "</p>");
  </script>

</body>
</html>
```

JavaScript

Beispiele

Beispiel 2: HTML-Elemente überschreiben

```
<html>  
<body>
```

```
<h1>My First Web Page</h1>
```

```
<p id="demo"></p>
```

```
<script type="text/javascript">  
    document.getElementById("demo").innerHTML=Date();  
</script>
```

```
</body>  
</html>
```

JavaScript

Grundlagen

Manche Browser unterstützen kein JavaScript!

- Diese Browser stellen JavaScript als Seiteninhalt dar!
- JavaScript sollte daher in HTML-Kommentarzeichen eingebettet werden. Dies ist Teil des JavaScript-Standards.
- Beispiel:

```
...  
    <script type="text/javascript">  
    <!--  
        document.getElementById("demo").innerHTML=Date();  
    //-->  
    </script>  
...
```

Der Doppelslash in der vorletzten Zeile ist ein JavaScript-Kommentar, der verhindert, dass „-->“ ausgeführt wird.

JavaScript

Grundlagen

JavaScript kann an unterschiedlichen Stellen stehen:

1. im Header des HTML-Dokuments oder
2. Im Body des HTML-Dokuments .

Wir können eine unbegrenzte Anzahl JavaScripts in einem Dokument ausführen lassen.

JavaScripte können im Header und im Body oder in beiden stehen.

Häufig findet man JavaScript im Header oder ganz am Ende eines Dokuments im Body-Bereich.

Beispiel 3: HTML-Elemente überschreiben

```
<html>

<head>
  <script type="text/javascript">
    function displayDate()
    {
      document.getElementById("demo").innerHTML=Date();
    }
  </script>
</head>

<body>
<h1>My First Web Page</h1>
<p id="demo"></p>

<button type="button" onclick="displayDate()">Display Date</button>
</body>

</html>
```

JavaScript

Grundlagen

JavaScript kann auch in externe Dateien ausgelagert werden:

```
<html>
<head>
...
<script type="text/javascript" src="xxx.js"></script>
...
</head>
<body>
...
</body>
</html>
```

JavaScript

Grundlagen --- Statements

- JavaScript berücksichtigt Groß-/Kleinschreibung.
- Jedes JavaScript-Statement ist ein Kommando für den Browser.
- JavaScript-Statements enden normalerweise mit einem Strichpunkt, müssen aber nicht!
- Ein JavaScript ist eine Sequenz von JavaScript-Statements.
- Kommentare
 - einzeliger Kommentar: `//`
 - Mehrzeilige Kommentare: `/* */`

JavaScript

Grundlagen --- Variablen

- JavaScript-Variablen
 - berücksichtigen Groß-/Kleinschreibung und
 - Müssen mit einem Buchstaben oder Unterstrich beginnen.

```
<html>
<body>
<script type="text/javascript">
    var firstname;
    firstname="Hege";
    document.write(firstname);
</script>
</body>
</html>
```

JavaScript

Grundlagen --- Variablen

- JavaScript-Variablendeklaration
 - Schlüsselwort: var
 - mit und ohne Wertzuweisung

```
var firstname;
```

```
var x=5;
```

```
var carname="volvo";
```

- Es gibt in JavaScript lokale globale Variablen
 - Lokale Variablen sind nur innerhalb einer Funktion gültig und werden am Ende der Funktion ungültig.
 - Globale Variablen sind funktionsübergreifend gültig und werden am Ende der Seitenverarbeitung ungültig.
 - Globale Variablen werden ohne das Schlüsselwort var deklariert.

```
var firstname;           // lokale Variable
x=5;                    // globale Variablen
carname="volvo";
```

JavaScript

Grundlagen --- Arithmetische Operatoren

- Mit der Vorbelegung $y=5$ gilt:

Operator	Description	Example	Result	
+	Addition	$x=y+2$	$x=7$	$y=5$
-	Subtraction	$x=y-2$	$x=3$	$y=5$
*	Multiplication	$x=y*2$	$x=10$	$y=5$
/	Division	$x=y/2$	$x=2.5$	$y=5$
%	Modulus (division remainder)	$x=y\%2$	$x=1$	$y=5$
++	Increment	$x=++y$	$x=6$	$y=6$
		$x=y++$	$x=5$	$y=6$
--	Decrement	$x=--y$	$x=4$	$y=4$
		$x=y--$	$x=5$	$y=4$

JavaScript

Grundlagen --- Zuweisungsoperatoren

- Mit der Vorbelegung $x=10$ und $y=5$ gilt:

Operator	Example	Same As	Result
=	$x=y$		$x=5$
+=	$x+=y$	$x=x+y$	$x=15$
-=	$x-=y$	$x=x-y$	$x=5$
=	$x=y$	$x=x*y$	$x=50$
/=	$x/=y$	$x=x/y$	$x=2$
%=	$x%=y$	$x=x\%y$	$x=0$

JavaScript

Grundlagen --- Zeichenketten


- Verkettung von Zeichenketten

```
txt1 = "What a very";  
txt2 = "nice day";  
txt3 = txt1 + " " + txt2;  
document.write(txt3);
```

JavaScript

Grundlagen --- Vergleichsoperatoren

- Mit der Vorbelegung `x=5` gilt:



Operator	Description	Example
<code>==</code>	is equal to	<code>x==8</code> is false <code>x==5</code> is true
<code>===</code>	is exactly equal to (value and type)	<code>x===5</code> is true <code>x==="5"</code> is false
<code>!=</code>	is not equal	<code>x!=8</code> is true
<code>></code>	is greater than	<code>x>8</code> is false
<code><</code>	is less than	<code>x<8</code> is true
<code>>=</code>	is greater than or equal to	<code>x>=8</code> is false
<code><=</code>	is less than or equal to	<code>x<=8</code> is true

JavaScript

Grundlagen --- Logische Operatoren

- Mit der Vorbelegung $x=6$ und $y=3$ gilt:

Operator	Description	Example
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x == 5 \ \ y == 5)$ is false
!	not	$!(x == y)$ is true

- Bedingte Zuweisung

```
variablename = (condition) ? value1 : value2
```

Beispiel:

```
greeting = (visitor=="PRES") ? "Dear President " : "Dear ";
```

JavaScript

Grundlagen --- Bedingte Verzweigung

- Bedingte Verzweigung

```
if (condition)  
{  
    code to be executed if condition is true  
}
```

- Beispiel

```
<script type="text/javascript">  
    // Write a "Good morning" greeting if the time is less than 10  
    var d=new Date();  
    var time=d.getHours();  
  
    if (time<10)  
    {  
        document.write("<b>Good morning</b>");  
    }  
</script>
```

JavaScript

Grundlagen --- Bedingte Verzweigung

- Bedingte Verzweigung 2

```
if (condition)
{
    code to be executed if condition is true
}
else
{
    code to be executed if condition is not true
}
```

- Beispiel

```
if (time<10)
{    document.write("<b>Good morning!</b>");    }
else
{    document.write("<b>Good day!</b>");    }
```

- Bedingte Verzweigung 3

```
if (condition1)
{
  code to be executed if condition1 is true
}
else if (condition2)
{
  code to be executed if condition2 is true
}
else
{
  code to be executed if neither condition1 nor condition2 is true
}
```

JavaScript

Grundlagen --- Bedingte Verzweigung

- Beispiel mit Zufallslink

```
<html>
<body>

<script type="text/javascript">
  var r=Math.random();
  if (r>0.5)
  {
    document.write("<a href='http://www.fh-hof.de'>FH</a>");
  }
  else
  {
    document.write("<a href='http://www.uni-bt.de'>Uni BT</a>");
  }
</script>

</body>
</html>
```

JavaScript

Grundlagen --- Bedingte Verzweigung

- Switch-Statement

```
switch(n)
{
case 1:
    execute code block 1
    break;
case 2:
    execute code block 2
    break;
default:
    code to be executed if n is different from case 1 and 2
}
```


JavaScript

Grundlagen --- Bedingte Verzweigung

- Switch-Statement

```
<script type="text/javascript">
  var d=new Date();
  var theDay=d.getDay();
  switch (theDay)
  {
    case 5: document.write("Freitag");
            break;
    case 6: document.write("Samstag");
            break;
    case 0: document.write("Sonntag");
            break;
    default:
            document.write("Bald ist Wochenende!");
  }
</script>
```

JavaScript

Grundlagen --- PopUp-Boxen

- Alert-Popup

```
<html>
<head>
<script type="text/javascript">
function show_alert()
{
    alert("I am an alert box" + '\n' + "with line break!");
}
</script>
</head>
<body>

<input type="button" onclick="show_alert()" value="Show alert box" />

</body>
</html>
```

JavaScript

Grundlagen --- PopUp-Boxen

- Confirm-Popup

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
    var r=confirm("Knopf drücken!");
    if (r==true){ alert("OK!"); }
    else { alert("Cancel!"); }
}
</script>
</head>
<body>

<input type="button" onclick="show_confirm()" value="Bestätigung" />

</body>
</html>
```

JavaScript

Grundlagen --- PopUp-Boxen

- Prompt-Popup

```
<html>
<head>
<script type="text/javascript">
function show_prompt()
{
var name=prompt("Please enter your name", "Harry Potter");
if (name!=null && name!="")
    {
    document.write("Hello " + name + "! How are you today?");
    }
}
</script>
</head>
<body>

<input type="button" onclick="show_prompt()" value="Show prompt box" />

</body>
</html>
```

JavaScript

Grundlagen --- Funktionen

- Syntax

```
function functionname(var1,var2,...,varX)
{
    Funktionsblock
}
```

- Beispiel

```
...
<script type="text/javascript">
    function product(a,b)
    {
        return a*b;
    }
</script>
</head>
<body>
    <script type="text/javascript">
        document.write(product(4,3));
    </script>
...

```

Beispiel

```
<html>
<head>
<script type="text/javascript">
    function myfunc(txt) {alert (txt);}
</script>
</head>
<body>

<form>
onclick="myfunc('Hello')" value="Call function">
</form>
```

<p>By pressing the button above, a function will be called with "Hello" as a parameter. The function will alert the parameter.</p>

```
</body>
</html>
```

- For-Loop

```
for (variable=start;variable<=end;variable=variable+increment)
{
  code to be executed
}
```

- Beispiel

```
...
<script type="text/javascript">
var i=0;
for (i=0;i<=5;i++)
{
  document.write("The number is " + i);
  document.write("<br />");
}
</script>
...
```

JavaScript

Grundlagen --- Schleifen

- While-Loop

```
while (variable<=endvalue)
{
    code to be executed
}
```

- Beispiel

```
...
<script type="text/javascript">
var i=0;
while (i<=5)
    {
        document.write("The number is " + i);
        document.write("<br />");
        i++;
    }
</script>
...
```


JavaScript

Grundlagen --- Schleifen

- Do-While-Loop

```
do
  {
    code to be executed
  }
while (variable<=endvalue);
```

- Beispiel

```
...
<script type="text/javascript">
var i=0;
do
  {
    document.write("The number is " + i);
    document.write("<br />");
    i++;
  }
while (i<=5);
</script>
```

- Schleifen unterbrechen

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
  {
    if (i==3)
      {
        break;
      }
    document.write("The number is " + i);
    document.write("<br />");
  }
</script>
</body>
</html>
```

- Schleifen beim nächsten Wert fortsetzen

```
<html>
<body>
<script type="text/javascript">
var i=0
for (i=0;i<=10;i++)
  {
    if (i==3)
    {
      continue;
    }
    document.write("The number is " + i);
    document.write("<br />");
  }
</script>
</body>
</html>
```

- For-In-Schleife

Die for-in-Schleife iteriert über die Eigenschaften eines Objektes.

```
for (variable in object)
{
  code to be executed
}
```

- Beispiel

```
var person={fname:"John",lname:"Doe",age:25};

for (x in person)
{
  document.write(person[x] + " ");
}
```

JavaScript

Grundlagen --- Ereignisse

- Ereignisse können von JavaScript detektiert werden.

```
<html>
<head>
<script type="text/javascript">
function displayDate()
{
    document.getElementById("demo").innerHTML=Date();
}
</script>
</head>

<body>
<h1>My First Web Page</h1>
<p id="demo"></p>
<button type="button" onclick="displayDate()">Display Date</button>
</body>
</html>
```

JavaScript

Grundlagen --- Ereignisse

- Ereignisse sind
 - Mausklicks
 - Webseite oder Bild laden
 - Mausbewegung über einen Hotspot der Seite
 - Auswahl eines Eingabefeldes in einem Formular
 - Abschicken eines Formulars
 - Tastendruck
- Trigger
 - onLoad, onUnload
 - onFocus, onBlur, onChange
 - onSubmit
 - onMouseover

JavaScript

Grundlagen --- Ereignisse

IE: Internet Explorer, **F:** Firefox, **O:** Opera, **W3C:** W3C Standard.

Attribute	The event occurs when...	IE	F	O	W3C
<u>onblur</u>	An element loses focus	3	1	9	Yes
<u>onchange</u>	The content of a field changes	3	1	9	Yes
<u>onclick</u>	Mouse clicks an object	3	1	9	Yes
<u>ondblclick</u>	Mouse double-clicks an object	4	1	9	Yes
<u>onerror</u>	An error occurs when loading a document or an image	4	1	9	Yes
<u>onfocus</u>	An element gets focus	3	1	9	Yes
<u>onkeydown</u>	A keyboard key is pressed	3	1	No	Yes
<u>onkeypress</u>	A keyboard key is pressed or held down	3	1	9	Yes
<u>onkeyup</u>	A keyboard key is released	3	1	9	Yes
<u>onload</u>	A page or image is finished loading	3	1	9	Yes
<u>onmousedown</u>	A mouse button is pressed	4	1	9	Yes
<u>onmousemove</u>	The mouse is moved	3	1	9	Yes
<u>onmouseout</u>	The mouse is moved off an element	4	1	9	Yes
<u>onmouseover</u>	The mouse is moved over an element	3	1	9	Yes
<u>onmouseup</u>	A mouse button is released	4	1	9	Yes
<u>onresize</u>	A window or frame is resized	4	1	9	Yes
<u>onselect</u>	Text is selected	3	1	9	Yes
<u>onunload</u>	The user exits the page	3	1	9	Yes

- Maus- und Tastaturereignisse

Property	Description	IE	F	O	W3C
altKey	Returns whether or not the "ALT" key was pressed when an event was triggered	6	1	9	Yes
button	Returns which mouse button was clicked when an event was triggered	6	1	9	Yes
clientX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
clientY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
ctrlKey	Returns whether or not the "CTRL" key was pressed when an event was triggered	6	1	9	Yes
metaKey	Returns whether or not the "meta" key was pressed when an event was triggered	6	1	9	Yes
relatedTarget	Returns the element related to the element that triggered the event	No	1	9	Yes
screenX	Returns the horizontal coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
screenY	Returns the vertical coordinate of the mouse pointer when an event was triggered	6	1	9	Yes
shiftKey	Returns whether or not the "SHIFT" key was pressed when an event was triggered	6	1	9	Yes

JavaScript

Grundlagen --- Fehlerbehandlung

- Try & Catch

```
try
  {
    //Run some code here
  }
catch(err)
  {
    //Handle errors here
  }
```

JavaScript

Grundlagen --- Fehlerbehandlung

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
    try
    {
        addAlert("Welcome guest!");
    }
    catch(err)
    {
        txt="There was an error on this page.\n\n";
        txt+="Error description: " + err.description + "\n\n";
        txt+="Click OK to continue.\n\n";
        alert(txt);
    }
}
</script>
</head>

<body>
    <input type="button" value="View message" onclick="message()" />
</body>
</html>
```

Debugging JavaScript

Joe Oakes

jxo19@psu.edu

Why Perform Debugging

- Helps you understand the coding syntax
- Helps you understand the code logic better
- Helps you understand the code and follow code pathways
- Helps you find bugs and logic errors
- Makes you a better programmer

```
<!DOCTYPE html>
```

```
<html>
```

```
<head></head>
```

```
<body>
```

```
  <a id= "add" onclick="addTwoNumbers()" href="#test">add two numbers</a>
```

```
<script>
```

```
  function addTwoNumbers(var1, var2){
```

```
    return var1+var2;
```

```
  }
```

```
</script>
```

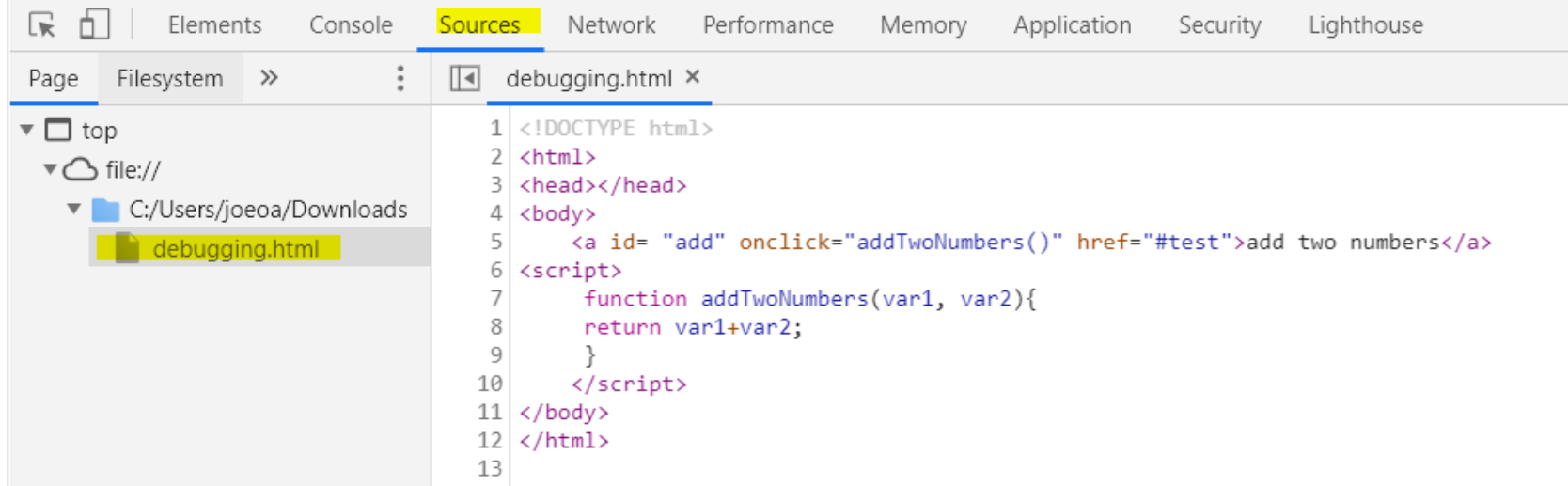
```
</body>
```

```
</html>
```

Source HTML and JavaScript code you can use to cut and paste into an editor

Open the Browser's Developer Tools
Control+Shift+I

[add two numbers](#)

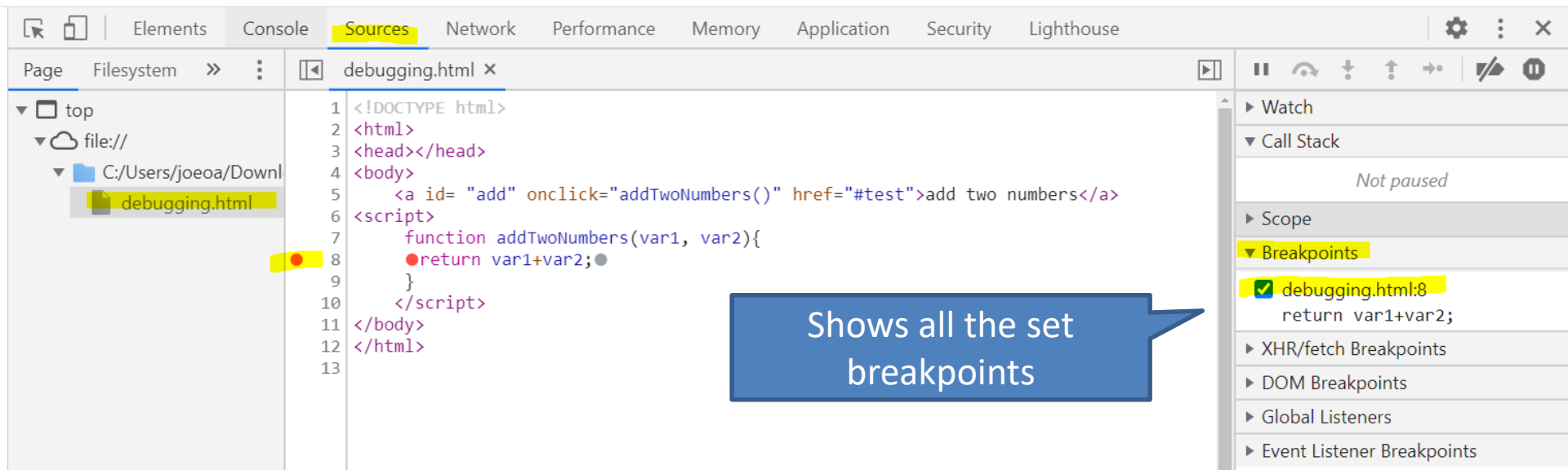


JavaScript Debugging Details

- **Breakpoints:** Set code execution points to stop
- **Instruction Pointer:** Current location in the code
- **Step over:** Step over a function it still executes
- **Step into:** Step into a function to trace through it
- **Step out:** Step out of a function
- **Call stack:** show what function calls have been push onto the stack
- **Watch:** watch variables/objects in the memory
- **Scope:** show variables in scope

Debugging Details: Breakpoint

- Breakpoint: Set code execution points to stop
 - Set the location by clicking on the gutter area
 - This can be toggled by clicking it again



Debugging Details: Breakpoint

- Reload the page and select the link
- Notice var1 and var2 are undefined variables

The screenshot shows the Chrome DevTools interface. The 'Sources' panel is open, displaying the code for 'debugging.html'. A red dot indicates a breakpoint is set at line 8, which is the return statement of the 'addTwoNumbers' function. The console shows the function was called with 'var1 = undefined, var2 = undefined'. The call stack shows the function was called from an 'onclick' event. The scope shows 'this: Window', 'var1: undefined', and 'var2: undefined'.

var1 and var2 are undefined


```
<!DOCTYPE html>
```

```
<html>
```

```
<head></head>
```

```
<body>
```

```
  <a id= "add" onclick="addTwoNumbers(2,4)" href="#test">add two numbers</a>
```

```
<script>
```

```
  function addTwoNumbers(var1, var2){
```

```
    return var1+var2;
```

```
  }
```

```
</script>
```

```
</body>
```

```
</html>
```

Added two input
argument values

debugging.html x

```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
8   ● return var1+var2; ●
9   }
10 </script>
11 </body>
12 </html>
13
```

Paused on breakpoint

Watch

Call Stack

addTwoNumbers

debugging.html:8

onclick

debugging.html:5

Scope

Local

this: Window

var1: 2

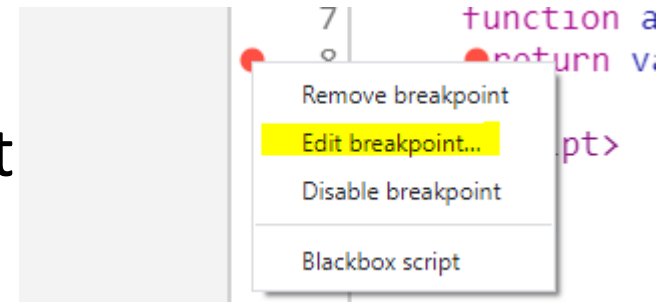
var2: 4

Global

Window

Debugging Details: Conditional Breakpoint

- Edit Breakpoint: right click
 - You can set conditional breakpoint

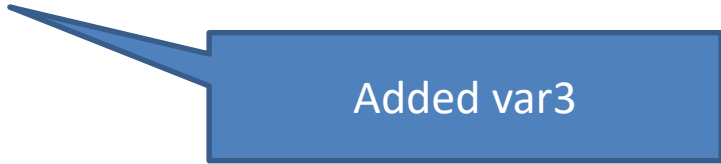


```
debugging.html x
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers()" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){
8     return var1+var2;
9   }
10 </script>
11 </body>
12 </html>
13
```

Line 8: Conditional breakpoint ▼

Expression to check before pausing, e.g. x > 5

```
<!DOCTYPE html>
<html>
<head></head>
<body>
  <a id= "add" onclick="addTwoNumbers(2,4)" href="#test">add two numbers</a>
<script>
  function addTwoNumbers(var1, var2){
    return var3 = var1+var2;
  }
</script>
</body>
</html>
```



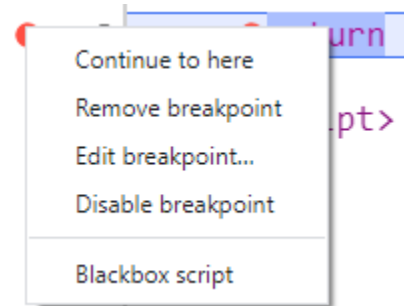
```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){
8     return var3 = var1+var2;
9   }
10 </script>
11 </body>
12 </html>
```

Line 8: Conditional breakpoint ▾

var3 > 5

Debugging Details: Conditional Breakpoint

- Conditional Breakpoint hit



The screenshot shows a browser window with the file 'debugging.html' open. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
8   return var3 = var1+var2;
9   }
10 </script>
11 </body>
12 </html>
13
```

The developer tools are open on the right, showing the following details:

- Paused on breakpoint** (highlighted in yellow)
- Call Stack**:
 - addTwoNumbers debugging.html:8
 - onclick VM847 debugging.html:5
- Scope**
- Local**:
 - this: Window
 - var1: 2
 - var2: 4
- Global**: Window
- Breakpoints**:
 - debugging.html:8 return var3 = var1+var2;

Debugging Details: Blackboxing

- Blackbox
 - When you don't know the internal workings of a given system
 - Blackboxing gives you a way to denote library (or other abstraction) code so that the debugger can route around it.

Continue to here
Remove breakpoint
Edit breakpoint...
Disable breakpoint

Blackbox script

Framework Blackboxing

Debugger will skip through the scripts and will not stop on exceptions thrown by them.

Debugging Details: Controls

- Debugger Controls



- **Resume:** Continue to the next breakpoint
- **Step over:** Step over a function it still executes
- **Step into:** Step into a function to step through it
- **Step out:** Step out of a function
- **Step:** Step to the next line
- **Deactivate Breakpoints**
- **Pause on exceptions**

Debugging Details: Examine Values

- You can use the Console to examine or change values

The screenshot shows the Chrome DevTools interface with a breakpoint set on line 8 of a JavaScript function. The function is named `addTwoNumbers` and takes two arguments, `var1` and `var2`. The breakpoint is located on the `return` statement, which is `return var3 = var1+var2;`. A callout box points to this line with the text: "This line has not been executed so var3 is not defined yet".

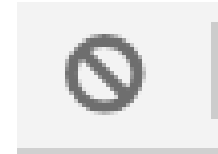
The right-hand side of the interface shows the "Paused on breakpoint" panel. The "Call Stack" shows the current call to `addTwoNumbers` at line 8 of `debugging.html`. The "Scope" panel shows the local variables `var1: 2` and `var2: 4`, and the global `Window` object.

The bottom of the screenshot shows the Console with the command `> var3` entered. The console displays an error: `Uncaught ReferenceError: var3 is not defined`. The error stack trace is as follows:

```
at eval (eval at addTwoNumbers (debugging.html:1), <anonymous>:1:1)
at addTwoNumbers (debugging.html:8)
at HTMLAnchorElement.onclick (VM45 debugging.html:5)
```

Debugging Details: Examine Values

- Use the Step icon to execute the line
- Clear the Console using the icon



The screenshot shows the Chrome DevTools interface with the Sources panel open to a file named 'debugging.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
8   return var3 = var1+var2;
9   }
10 </script>
11 </body>
12 </html>
13
```

The function call on line 5 and the function definition on lines 7-9 are highlighted. The return statement on line 8 is also highlighted. The status bar at the bottom indicates 'Line 8, Column 27' and 'Coverage: n/a'.

The right-hand side of the interface shows the 'Debugger paused' state. The 'Call Stack' panel shows the current call to 'addTwoNumbers' at 'debugging.html:8'. The 'Scope' panel shows the 'Local' scope with the following values:

- Return value: 6
- this: Window
- var1: 2
- var2: 4

The 'Global' scope is also visible, showing 'Window'.

> var3

< 6

Debugging Details: Watch

- You can use the Watch feature to watch a variable
- Click on the + to select the variable to watch

The screenshot shows a web browser's developer tools interface. The main pane displays HTML and JavaScript code. A red dot on line 8 indicates the current execution point. The 'Watch' panel on the right shows a list of variables being monitored, with 'var3: 6' highlighted. A blue callout box points to the '+' icon in the Watch panel, with the text 'Use the + to add an expression'.

```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
8   ● return var3 = var1+var2; ●
9   }
10 </script>
11 </body>
12 </html>
13
```

Debugger paused

Watch + ↻

var3: 6

Call Stack

- addTwoNumbers debugging.html:8
- onclick VM45 debugging.html:5

Scope

Local

- Return value: 6
- this: Window
- var1: 2
- var2: 4

Line 13, Column 1 Coverage: n/a

Console What's New

```
> var3
< 6
```

Debugging Details: Assignment

- You can change the value of a variable in the console

The screenshot displays the Chrome DevTools interface during a debugging session. The Sources panel shows the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head></head>
4 <body>
5   <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
6 <script>
7   function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
8   return var3 = var1+var2;
9   }
10 </script>
11 </body>
12 </html>
13
```

The Watch panel on the right shows the current state of variables:

- var3: 7

The Console panel at the bottom shows the following commands and their results:

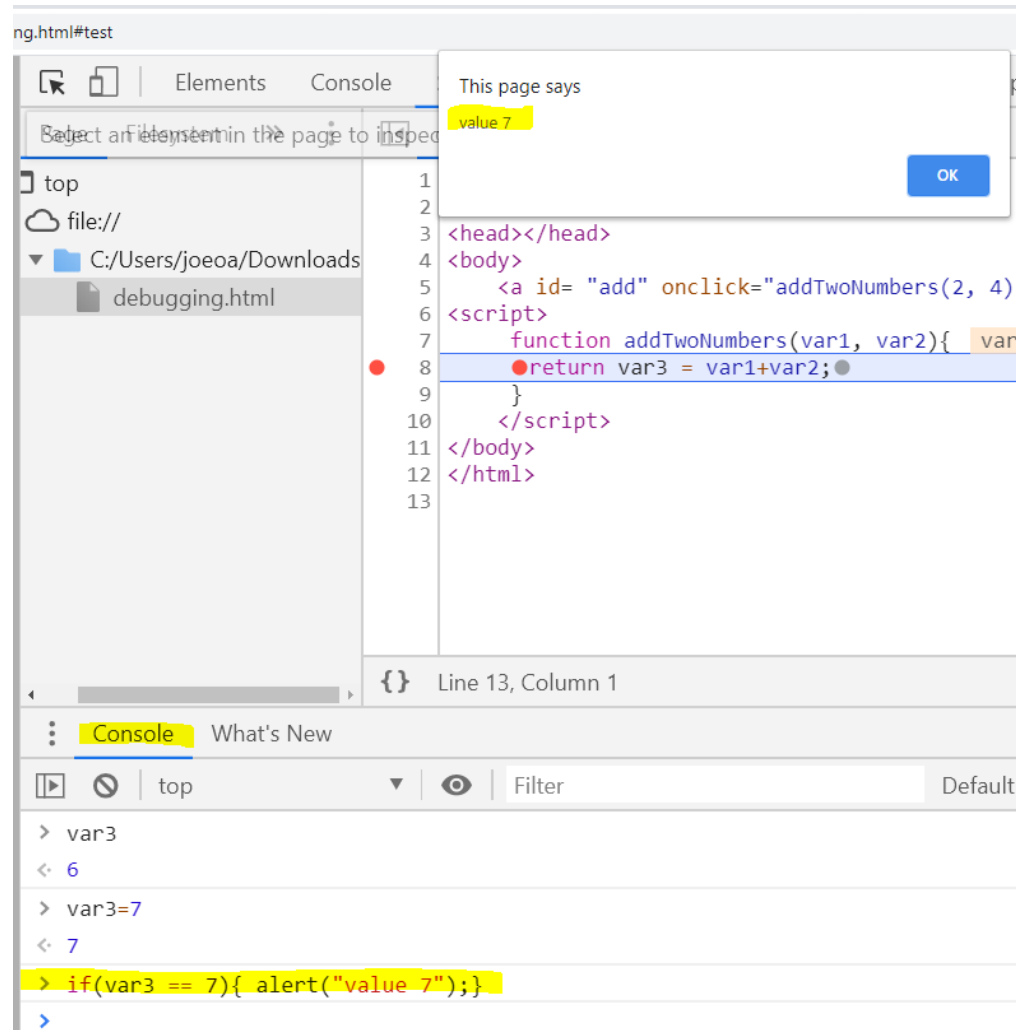
```
> var3
< 6
> var3=7
< 7
```

Two callout boxes provide additional context:

- A blue box with the text "Make sure to refresh" points to the Watch panel, indicating that the variable's value is updated only when the page is refreshed.
- A blue box with the text "Using the assignment operator changed the value from 6 to 7" points to the Console panel, highlighting the effect of the assignment operator (=) on the variable's value.

Debugging Details: Console

- You can put Javascript code in the console to execute



The screenshot shows a web browser's developer console with the following components:

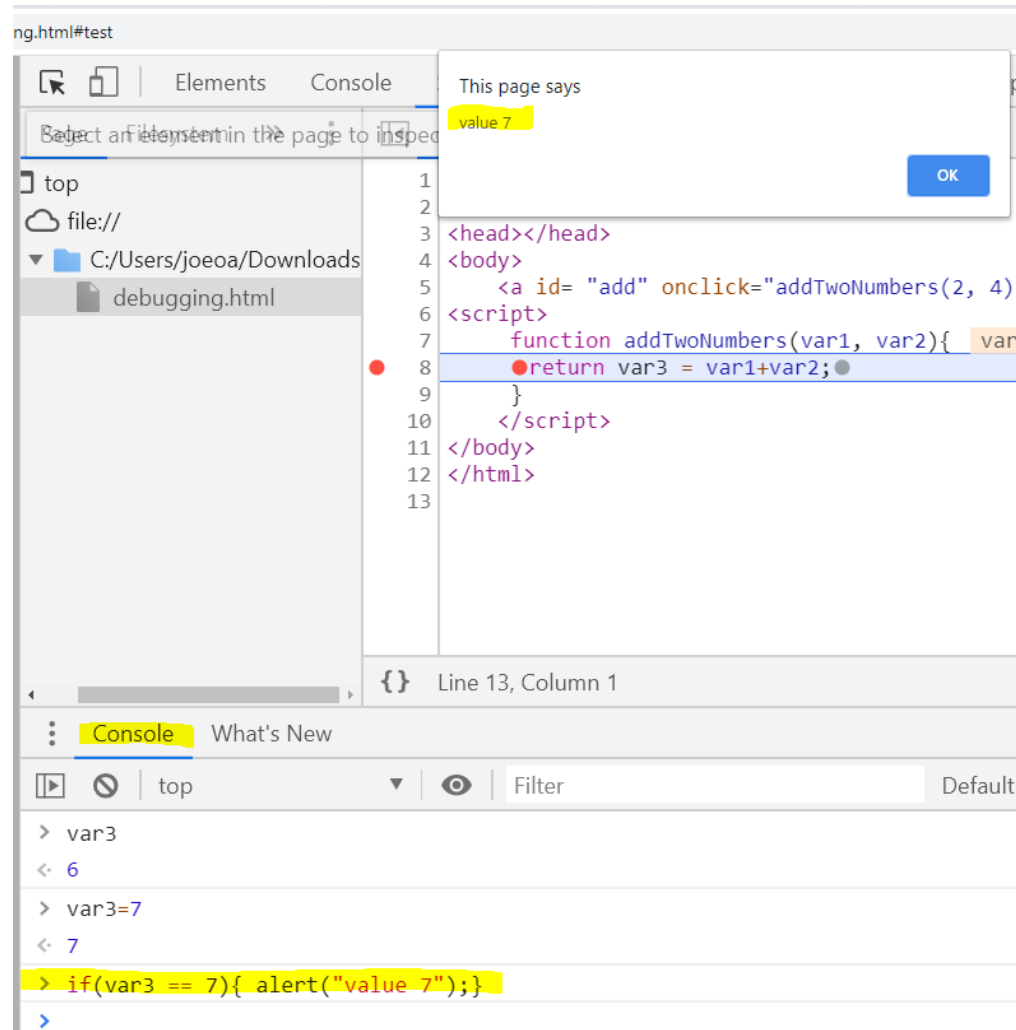
- Elements Panel:** Shows the file path `C:/Users/joeoa/Downloads/debugging.html`.
- Code Editor:** Contains the following HTML and JavaScript code:

```
1  
2  
3 <head></head>  
4 <body>  
5   <a id= "add" onclick="addTwoNumbers(2, 4)"  
6 <script>  
7   function addTwoNumbers(var1, var2){ var  
8   return var3 = var1+var2;  
9   }  
10  </script>  
11 </body>  
12 </html>  
13
```
- Alert Dialog:** A dialog box titled "This page says" with the text "value 7" and an "OK" button.
- Console Panel:** Shows the execution of the code:

```
> var3  
< 6  
> var3=7  
< 7  
> if(var3 == 7){ alert("value 7");}  
>
```

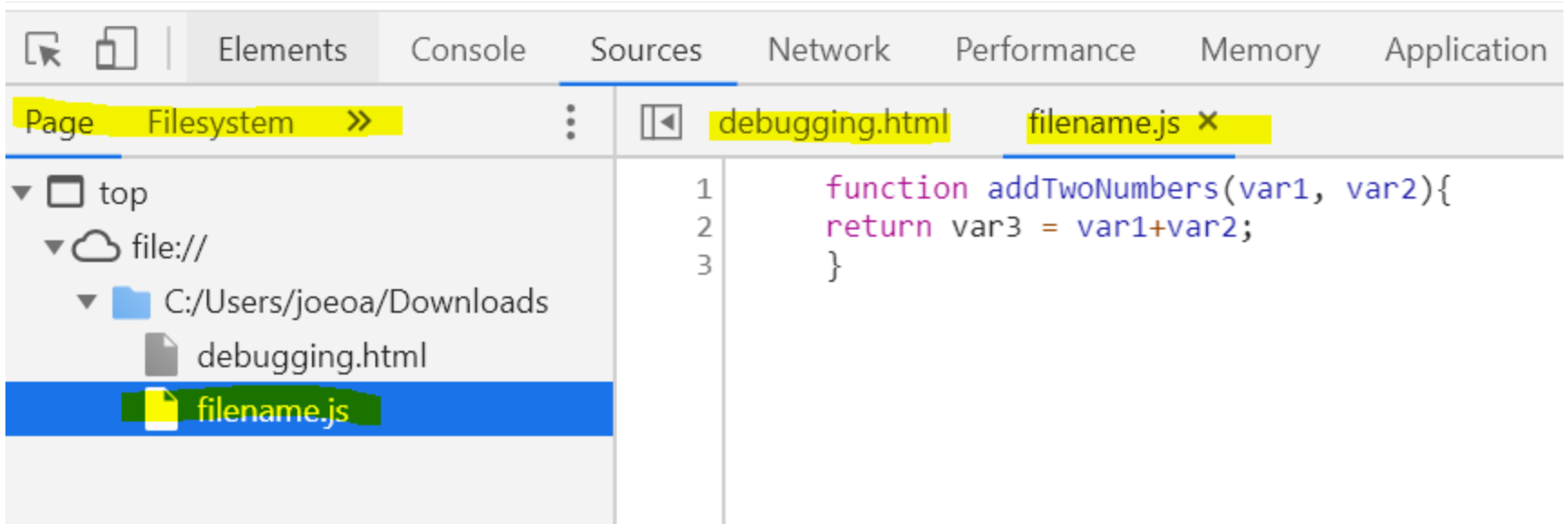
Debugging Details: Console

- You can put Javascript code in the console to execute



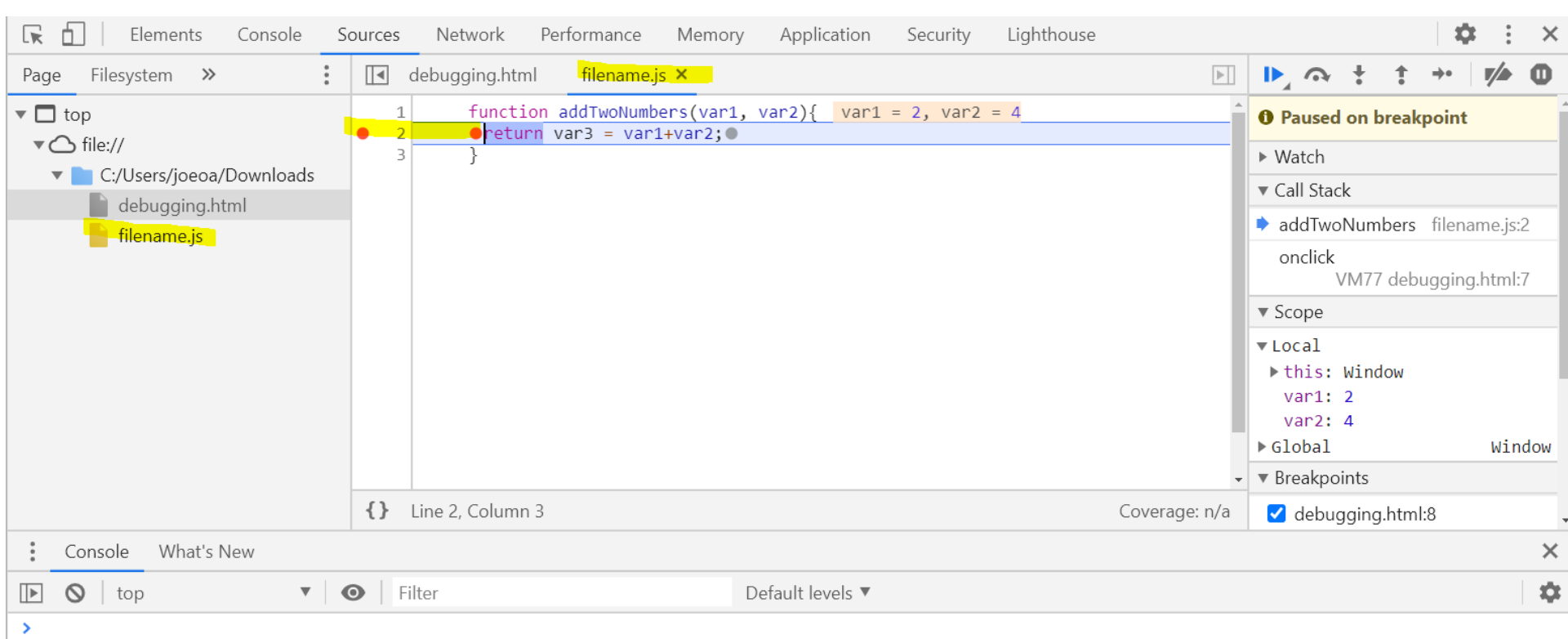
Debugging Details: External File

- Select Page -> Filesystem -> filename.js



Debugging Details: External File

- Select Page -> Filesystem -> filename.js
- Set your breakpoints in the gutter area



Debugging Details: Breakpoints

- You have many breakpoints set
 - You can manage them in the Breakpoints area

The screenshot shows a web browser's developer tools interface. The main pane displays the source code for 'filename.js' with two functions: 'addTwoNumbers' and 'subTwoNumbers'. Both functions have a red dot breakpoint on their respective 'return' statements. The right-hand sidebar contains several panels: 'Watch' (showing 'total: <not available>'), 'Call Stack' (showing 'Not paused'), 'Scope' (showing 'Not paused'), and 'Breakpoints'. The 'Breakpoints' panel is expanded and shows two active breakpoints: 'filename.js:2' and 'filename.js:5', both with checkboxes checked and the corresponding code lines highlighted in yellow.

```
1 function addTwoNumbers(var1, var2){
2   return var3 = var1+var2;
3 }
4 function subTwoNumbers(var1, var2){
5   return var3 = var1-var2;
6 }
7
```

▼ Watch + ↻
total: <not available>

▼ Call Stack
Not paused

▼ Scope
Not paused

▼ Breakpoints

- filename.js:2
return var3 = var1+var2;
- filename.js:5
return var3 = var1-var2;

Debugging Details: Breakpoints

- When the breakpoint is hit notice the area will turn yellow

The screenshot shows a web browser's developer console with a JavaScript file named 'filename.js' open. The code in the editor is as follows:

```
1 function addTwoNumbers(var1, var2){
2   return var3 = var1+var2;
3 }
4 function subTwoNumbers(var1, var2){ var1 = 4, var2 = 4
5   return var3 = var1-var2;
6 }
7
```

The breakpoint is set on line 5, and it has been hit, as indicated by the yellow highlight on the return statement. The right-hand sidebar shows the following details:

- Paused on breakpoint**
- Watch**
- Call Stack**
 - subTwoNumbers filename.js:5
 - onclick VM188 debugging.html:8
- Scope**
- Breakpoints**
 - filename.js:2
return var3 = var1+var2;
 - filename.js:5
return var3 = var1-var2;

A blue callout box with the text "Breakpoint hit" points to the breakpoint in the sidebar.

JavaScript: Variable Scope

- **Variable scope:** what is the value of a specific variable name at the current line of code being executed
- JavaScript allows for both **global** and **local** scope of a variable
- **Global:** defined in the main JavaScript
- **Local:** defined in a function a new variable is created in memory
- Outside of the function referencing global

The image shows a JavaScript code editor with a function `addTwoNumbers` defined. The function has two parameters, `var1` and `var2`, and a return statement that calculates `var3 = var1 + var2`. A breakpoint is set at the return statement. The debugger is paused on this breakpoint. The `Scope` panel on the right shows the local variables `var1` and `var2` with values 2 and 4, respectively. A blue callout box labeled "Local variables" points to the local scope in the debugger.

```
1 function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
2 return var3 = var1+var2;
3 }
```

Paused on breakpoint

- ▶ Watch
- ▼ Call Stack
 - ▶ addTwoNumbers filename.js:2
 - onclick VM77 debugging.html:7
- ▼ Scope
 - ▼ Local
 - ▶ this: Window
 - var1: 2
 - var2: 4
 - ▶ Global Window

Local variables

JavaScript: Variable Scope

- A variable can either be global or local
 - **Global variable** can be referenced from anywhere in the script
 - **Local variable** only exists with the function in which it is declared

The screenshot displays a JavaScript debugger interface. The main window shows the following code:

```
1 var total = 50;
2 function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
3 return var3 = var1+var2;
4 }
```

A breakpoint is set at line 3. The right-hand sidebar shows the following information:

- Paused on breakpoint**
- Watch:** total: 50
- Call Stack:** addTwoNumbers filename.js:3, onclick VM98 debugging.html:7
- Scope:**
 - Local:** this: Window, var1: 2, var2: 4
 - Global:** Window
 - PERSISTENT: 1
 - TEMPORARY: 0
 - addEventListener: f addEve...
 - addTwoNumbers: f addTwoNum...
 - alert: f alert()
 - applicationCache: Applicat...
 - atob: f atob()

The bottom-left pane shows the current object's properties, including `total: 50`. A blue callout box points to the Global scope in the right sidebar with the text: "Global variables Vertical Scroll down to see total".

Line 3, Column 3 Coverage: n/a

JavaScript: Variable Scope

- myVar on line 2 is a global variable
- myVar on line 5 is a local variable
- Notice line 9 will use the global variable value since it is in scope

```
01 <script>
02   var myVar = 1;
03   function writeIt(){
04     var myVar = 2;
05     document.write(myVar);
06     writeMore();
07   }
08   function writeMore(){
09     document.write(myVar);
10   }
11 </script>
```

JavaScript: Log and line of code Breakpoint

- debugger allows you set in the code where the breakpoint will be hit
- console.log() allows you write to the console area as the code is executing

```
debugging.html  filename.js x
1  function addTwoNumbers(var1, var2){ var1 = 2, var2 = 4
2  var3 = var1+var2;
3  debugger;
4  return var3;
5  }
```

The screenshot shows the Chrome DevTools interface. The top bar includes tabs for Elements, Console, Sources, Network, Performance, Memory, and Application. The Sources tab is active, showing a file named 'filename.js' with a breakpoint set at line 3. The code in the editor is as follows:

```
1  function addTwoNumbers(var1, var2){
2  var3 = var1+var2;
3  debugger;
4  return var3;
5  }
6  function subTwoNumbers(var1, var2){
7  var3 = var1-var2;
8  console.log("var3:" + var3);
9  return var3;
10 }
```

The console at the bottom shows the output of the log statement: 'var3:0'. The status bar at the bottom indicates the current position is 'Line 3, Column 4'.

JavaScript: Console Log

- debugger allows you set in the code where the breakpoint will be hit
- console.log() allows you write to the console area as the code is executing

The screenshot displays a web browser's developer console. The top-left pane shows a file explorer with 'debugging.html' and 'filename.js'. The top-right pane shows JavaScript code with a breakpoint at line 10, column 5. The bottom pane shows the console output, including 'var3:0', the document head structure, and the document body structure.

```
1 function addTwoNumbers(var1, var2){
2   var3 = var1+var2;
3   debugger;
4   return var3;
5 }
6 function subTwoNumbers(var1, var2){
7   var3 = var1-var2;
8   console.clear();
9   console.log("var3:" + var3);
10  console.log(document.head);
11  console.log(document.body);
12  return var3;
13 }
14 function ajax(){
15   var xhr = new XMLHttpRequest();
16   xhr.open('GET', 'https://reqres.in/api/users', true);
17
18   xhr.send();
19 }
```

Line 10, Column 5

Console What's New

var3:0

```
<head>
  <script type="text/javascript" src="filename.js"></script>
</head>
<body>
  <a id="add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a>
  <br>
  <a id="sub" onclick="subTwoNumbers(4, 4)" href="#test">sub two numbers</a>
  <br>
  <a id="ajax" onclick="ajax()" href="#test">Ajax request</a>
</body>
```

JavaScript: AJAX

The `XMLHttpRequest` method `send()` sends the request to the server. If the request is asynchronous (which is the default), this method returns as soon as the request is sent and the result is delivered using events. If the request is synchronous, this method doesn't return until the response has arrived.

```
debugging.html  filename.js x
1  function addTwoNumbers(var1, var2){
2  return var3 = var1+var2;
3  }
4  function subTwoNumbers(var1, var2){
5  return var3 = var1-var2;
6  }
7  function ajax(){
8      var xhr = new XMLHttpRequest();
9      xhr.open('GET', 'https://reqres.in/api/users', true);
10
11     xhr.send();
12
13     xhr.onload = function() {
14         let responseObj = xhr.response;
15         alert(responseObj);
16     };
17 }
```

```
/<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script type="text/javascript" src="filename.js"></script>
```

```
</head>
```

```
<body>
```

```
  <a id= "add" onclick="addTwoNumbers(2, 4)" href="#test">add two numbers</a><br />
```

```
    <a id= "sub" onclick="subTwoNumbers(4, 4)" href="#test">sub two numbers</a><br />
```

```
    <a id= "ajax" onclick="ajax()" href="#test">Ajax request</a>
```

```
</body>
```

```
</html>
```

```
function addTwoNumbers(var1, var2){
```

```
  return var3 = var1+var2;
```

```
}
```

```
function subTwoNumbers(var1, var2){
```

```
  return var3 = var1-var2;
```

```
}
```

```
function ajax(){
```

```
  var xhr = new XMLHttpRequest();
```

```
  xhr.open('GET', 'https://reqres.in/api/users', true);
```

```
  xhr.send();
```

```
  xhr.onload = function() {
```

```
    let responseObj = xhr.response;
```

```
    alert(responseObj);
```

```
  };
```

```
}
```

HTML for AJAX

JavaScript code for AJAX

JavaScript: AJAX

- Notice the three xhr methods: open, send, onload

```
1 function addTwoNumbers(var1, var2){
2   return var3 = var1+var2;
3 }
4 function subTwoNumbers(var1, var2){
5   return var3 = var1-var2;
6 }
7 function ajax(){
8   var xhr = new XMLHttpRequest();
9   xhr.open('GET', 'https://reqres.in/api/users', true);
10
11  xhr.send();
12
13  xhr.onload = function() {
14    let responseObj = xhr.response;
15    alert(responseObj);
16  };
17 }
18
```

Paused on breakpoint

- ▶ Watch
- ▶ Call Stack
- ▶ Scope
- ▶ Breakpoints
- ▼ XHR/fetch Breakpoints +
No breakpoints
- ▶ DOM Breakpoints
- ▶ Global Listeners
- ▼ Event Listener Breakpoints
 - ▶ Animation
 - ▶ Canvas
 - ▶ Clipboard
 - ▶ Control
 - ▶ DOM Mutation

Line 15, Column 5 Coverage: n/a

Console What's New

```
> xhr.response
{"page":1,"per_page":6,"total":12,"total_pages":2,"data":[{"id":1,"email":"george.bluth@reqres.in","first_name":"George","last_name":"Bluth","avata...
```


JavaScript: AJAX

- You can set a breakpoint on any XHR or fetch

The image shows a browser's developer console with a JavaScript file named 'filename.js' open. The code defines three functions: 'addTwoNumbers', 'subTwoNumbers', and 'ajax'. The 'ajax' function creates an XMLHttpRequest object and sends a GET request to 'https://reqres.in/api/users'. A breakpoint is set on the 'xhr.send()' line (line 11). The right-hand sidebar shows the 'Paused on XHR or fetch' status and a list of breakpoint categories, with 'Any XHR or fetch' checked.

```
1 function addTwoNumbers(var1, var2){
2   return var3 = var1+var2;
3 }
4 function subTwoNumbers(var1, var2){
5   return var3 = var1-var2;
6 }
7 function ajax(){
8   var xhr = new XMLHttpRequest(); xhr = XMLHttpRequest {onreadystatechange
9   xhr.open('GET', 'https://reqres.in/api/users', true);
10
11  xhr.send();
12
13  xhr.onload = function() {
14    let responseObj = xhr.response;
15    alert(responseObj);
16  };
17 }
18
```

Paused on XHR or fetch
https://reqres.in/api/users

- ▶ Watch
- ▶ Call Stack
- ▶ Scope
- ▶ Breakpoints
- ▼ XHR/fetch Breakpoints +
 - Any XHR or fetch
 - ▶ DOM Breakpoints
 - ▶ Global Listeners
 - ▼ Event Listener Breakpoints

JavaScript: AJAX

- You can set a breakpoint on an Event Listener - XHR

The screenshot shows a code editor with the following JavaScript code:

```
1 function addTwoNumbers(var1, var2){
2   return var3 = var1+var2;
3 }
4 function subTwoNumbers(var1, var2){
5   return var3 = var1-var2;
6 }
7 function ajax(){
8   var xhr = new XMLHttpRequest();
9   xhr.open('GET', 'https://reqres.in/api/users', true);
10  xhr.send();
11
12  xhr.onload = function() {
13    let responseObj = xhr.response;
14    alert(responseObj);
15  };
16 }
17
18
```

The code is displayed in a window titled "debugging.html" with a sub-tab "filename.js". The current line of code is highlighted in blue, indicating a breakpoint is set on line 14: `let responseObj = xhr.response;`. The status bar at the bottom shows "Line 14, Column 23" and "Coverage: n/a".

On the right side, the "Event Listener Breakpoints" panel is open, showing a list of event types. The "XHR" event type is selected and checked. Underneath, the following event types are also checked:

- readystatechange
- load
- loadstart
- loadend

JavaScript: Log and line of code Breakpoint

- debugger allows you set in the code where the breakpoint will be hit
- console.log() method allows you write to the console area as the code is executing

```
function addTwoNumbers(var1, var2){  
    var3 = var1+var2;  
    debugger;  
    return var3;  
}  
function subTwoNumbers(var1, var2){  
    var3 = var1-var2;  
    console.log("var3:" + var3);  
    return var3;  
}
```

JavaScript: Events

- Events
 - Actions that are preformed by a user that can be detected by JavaScript
 - Every element in the DOM has certain events linked to them
 - Examples
 - Mouse click
 - Page or image loading
 - Mouse over
 - Selecting an input field
 - Submitting form
 - Keystroke

JavaScript: Events

- Events Contd.
 - onLoad and unload
 - Triggered when the user enters or leaves a page
 - Can be used for browser detection and cookies
 - onFocus, onBlur, and onChange
 - Used to validate form fields
 - onSubmit
 - Used to validate all form fields before it is submitted
 - onMouseOver and onMouseOut
 - Used for animations and effects

Internettechniken

JavaScript Teil 2 --- AJAX

Prof. Dr. Jürgen Heym

Hochschule Hof

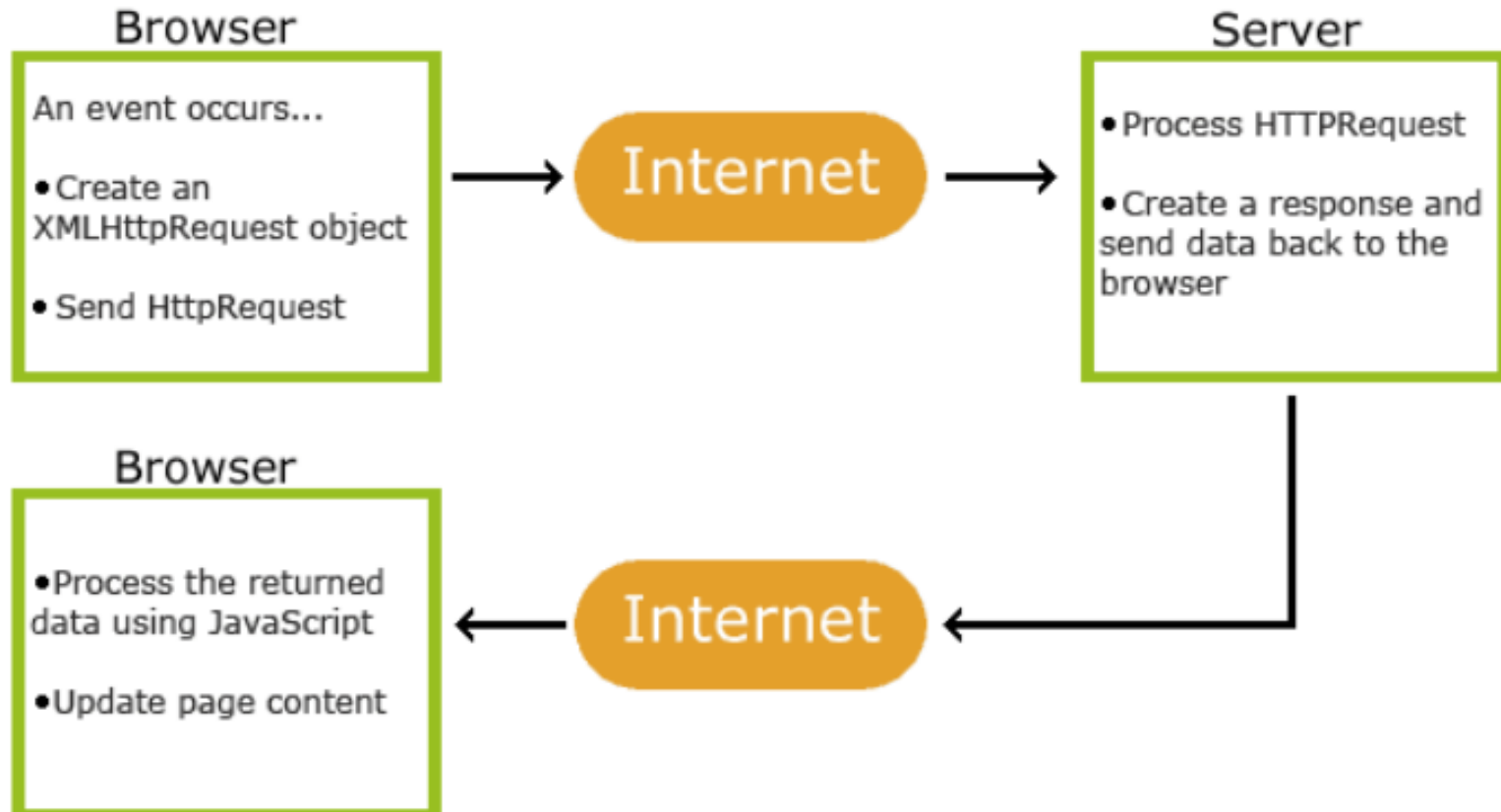
JavaScript

Einleitung

- AJAX == Asynchronous JavaScript and XML
- AJAX unterstützt den Datenaustausch mit einem Server und die Teile einer Webseite auszutauschen, ohne die gesamte Seite neu zu laden.
- AJAX ist keine neue Programmiersprache, sondern ein anderer Weg bekannt Standards einzusetzen.
 - XMLHttpRequest Objekt (asynchroner Datenaustausch)
 - JavaScript/DOM
 - CSS
 - XML
- AJAX–Anwendungen sind (meist) unabhängig von Browser und Plattform.

JavaScript

Datenflussdiagramm



- Beispielanwendung
 - Das Dokument enthält eine DIV-Sektion, die dynamisch mit Inhalten vom Server gefüllt werden soll, sobald der Knopf aktiviert wird.

```
<html>
```

```
<body>
```

```
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
```

```
<button type="button" onclick="loadXMLDoc()">Change Content</button>
```

```
</body>
```

```
</html>
```

JavaScript

Beispiel

- Beispielanwendung --- 2. Schritt
 - Fügen Sie einen JavaScript-Rumpf hinzu.

```
...  
<head>  
<script type="text/javascript">  
    function loadXMLDoc()  
    {  
        //.... AJAX script goes here ...  
    }  
</script>  
</head>  
...
```

JavaScript

Beispiel

- Beispielanwendung --- 3. Schritt
 - Wir erzeugen ein XMLHttpRequest-Objekt für den Datenaustausch mit dem Server.

...

```
var xmlhttp;  
if (window.XMLHttpRequest)  
{ // code for IE7+, Firefox, Chrome, Opera, Safari  
  xmlhttp=new XMLHttpRequest();  
}  
else  
{ // code for IE6, IE5  
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
}
```

...

JavaScript

Beispiel

- Beispielanwendung --- 4. Schritt
 - Als nächstes nutzen wir die Methoden `open` und `send` des XMLHttpRequest-Objekts, um eine Datei nachzuladen.

...

```
xmlhttp.open("GET", "ajax_info.txt", true);
```

```
xmlhttp.send();
```

...

Method	Description
<code>open(method,url,async)</code>	Specifies the type of request, the URL, and if the request should be handled asynchronously or not. <i>method</i> : the type of request: GET or POST <i>url</i> : the location of the file on the server <i>async</i> : true (asynchronous) or false (synchronous)
<code>send(string)</code>	Sends the request off to the server. <i>string</i> : Only used for POST requests

JavaScript

GET-Request

- Beispiele für GET-Requests

...

```
xmlhttp.open("GET", "demo_get.asp", true);  
xmlhttp.send();
```

...

- Könnte eine Seite aus dem Cache liefern.
Deswegen fügt meine eine zufällige ID der URL hinzu:

...

```
xmlhttp.open("GET", "demo_get.asp?t=" + Math.random(), true);  
xmlhttp.send();
```

...

JavaScript

POST-Request

- POST-Requests sind robuster und sicherer.
Wir können damit ein Formular „imitieren“.
Der Header muss in diesem Fall gesetzt werden:

...

```
xmlhttp.open("POST", "ajax_test.asp", true);  
xmlhttp.setRequestHeader("Content-type", "application/x-www-form-urlencoded");  
xmlhttp.send("fname=Henry&lname=Ford");
```

...

Method	Description
<code>setRequestHeader(<i>header</i>,<i>value</i>)</code>	Adds HTTP headers to the request. <i>header</i> : specifies the header name <i>value</i> : specifies the header value

JavaScript

Asynchron --- True oder False?

- Damit wir echtes AJAX nutzen, ist der Parameter „asynchron“ auf den Wert „true“ zu setzen.
- Man muss in diesem Fall eine Funktion definieren, die ausgeführt wird, sobald die Antwort fertig ist.

...

```
xmlhttp.onreadystatechange=function()  
{  
  if (xmlhttp.readyState==4 && xmlhttp.status==200)  
  {  
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  
  }  
}  
xmlhttp.open("GET","ajax_info.txt",true);  
xmlhttp.send();
```

...

JavaScript

Gesamtes Beispiel ajax-1.html + ajax_info.txt

```
<html>
<head>
<script type="text/javascript">
function loadXMLDoc()
{
var xmlhttp;
if (window.XMLHttpRequest)
  {// code for IE7+, Firefox, Chrome, Opera, Safari
  xmlhttp=new XMLHttpRequest();
  }
else
  {// code for IE6, IE5
  xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
xmlhttp.onreadystatechange=function()
  {
  if (xmlhttp.readyState==4 && xmlhttp.status==200)
    {
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
    }
  }
xmlhttp.open("GET","ajax_info.txt",true);
xmlhttp.send();
}
</script>
</head>
<body>

<div id="myDiv"><h2>Let AJAX change this text</h2></div>
<button type="button" onclick="loadXMLDoc()">Change Content</button>

</body>
</html>
```


JavaScript

HttpResponse

- Server-Antwort als String oder XML
 - Um die Antwort vom Server zu verarbeiten nutzen wir die Eigenschaften `responseText` oder `responseXML` des XHR-Objekts:

Property	Description
<code>responseText</code>	get the response data as a string
<code>responseXML</code>	get the response data as XML data

- Beispiel für `responseText`

```
document.getElementById("myDiv").innerHTML=xmlhttp.responseText;
```

JavaScript

responseXML

- Server-Antwort als String oder XML
 - Beispiel für responseXML

```
xmlDoc=xmlhttp.responseXML;  
  
txt="";  
  
x=xmlDoc.getElementsByTagName("ARTIST");  
  
for (i=0;i<x.length;i++){  
    txt=txt + x[i].childNodes[0].nodeValue + "<br />";  
}  
  
document.getElementById("myDiv").innerHTML=txt;
```

- Sobald der Request zum Server geschickt wurde, möchte man Aktionen ausführen, die von der Antwort abhängig sind.
- Das Ereignis “onreadystatechange” wird immer, wenn der Zustand “readyState” sich ändert, ausgelöst.
- Die Eigenschaft “readyState” enthält den Status des XMLHttpRequest.
- Die drei wichtigsten Eigenschaften des XMLHttpRequest-Objekts sind:

Property	Description
onreadystatechange	Stores a function (or the name of a function) to be called automatically each time the readyState property changes
readyState	Holds the status of the XMLHttpRequest. Changes from 0 to 4: 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
status	200: "OK" 404: Page not found

- Wir spezifizieren die Funktion, die ausgeführt werden soll, wenn die Antwort des Servers vorliegt:

```
xmlhttp.onreadystatechange=function(  
{  
  if (xmlhttp.readyState==4 && xmlhttp.status==200)  
  {  
    document.getElementById("myDiv").innerHTML=xmlhttp.responseText;  
  }  
}
```

- Sobald man mehrere AJAX-Funktionen nutzen möchte sollte man Callback-Funktionen einsetzen:

```
...  
<script type="text/javascript">  
var xmlhttp;  
function loadXMLDoc(url, cfunc)  
{  
if (window.XMLHttpRequest)  
    {  
        // code for IE7+, Firefox, Chrome, Opera, Safari  
        xmlhttp=new XMLHttpRequest();  
    }  
else  
    {  
        // code for IE6, IE5  
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");  
    }  
xmlhttp.onreadystatechange=cfunc;  
xmlhttp.open("GET",url,true);  
xmlhttp.send();  
}  
...
```

- In myFunction wird die soeben definierte Funktion loadXMLDoc genutzt:

...

```
function myFunction1 ()
{
loadXMLDoc ("ajax_info.txt", function ()
    {
    if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
        document.getElementById ("myDiv").innerHTML=xmlhttp.responseText;
        }
    });
}
```

</script>

...

JavaScript

Callback-Funktion

- ... und hier wird myFunction eingesetzt:

...

```
<body>
```

```
<div id="myDiv"><h2>Let AJAX change this text</h2></div>
```

```
<button type="button" onclick="myFunction1 () ">Change Content</button>
```

```
</body>
```

```
</html>
```

Internettechniken

JavaScript Teil 3 --- PopUp-Fenster

Prof. Dr. Jürgen Heym

Hochschule Hof

JavaScript

PopUp-Fenster mit dem Window-Objekt

- Das JavaScript-Objekt „Window“ repräsentiert ein offenes Fenster in einem Browser (Root-Window, PopUp-Windows, Frames, etc.).
- Das Window-Objekt ist nicht standardisiert, aber alle Browser unterstützen es.
- Das Windows-Objekt hat Eigenschaften (properties) und Methoden (methods).

Property	Description
closed	Returns a Boolean value indicating whether a window has been closed or not
defaultStatus	Sets or returns the default text in the statusbar of a window
document	Returns the Document object for the window (See Document object)
frames	Returns an array of all the frames (including iframes) in the current window
history	Returns the History object for the window (See History object)
innerHeight	Sets or returns the the inner height of a window's content area
innerWidth	Sets or returns the the inner width of a window's content area
length	Returns the number of frames (including iframes) in a window
location	Returns the Location object for the window (See Location object)
name	Sets or returns the name of a window
navigator	Returns the Navigator object for the window (See Navigator object)
opener	Returns a reference to the window that created the window
outerHeight	Sets or returns the outer height of a window, including toolbars/scrollbars
outerWidth	Sets or returns the outer width of a window, including toolbars/scrollbars
pageXOffset	Returns the pixels the current document has been scrolled (horizontally) from the upper left corner of the window
pageYOffset	Returns the pixels the current document has been scrolled (vertically) from the upper left corner of the window
parent	Returns the parent window of the current window
screen	Returns the Screen object for the window (See Screen object)
screenLeft	Returns the x coordinate of the window relative to the screen
screenTop	Returns the y coordinate of the window relative to the screen
screenX	Returns the x coordinate of the window relative to the screen
screenY	Returns the y coordinate of the window relative to the screen
self	Returns the current window
status	Sets the text in the statusbar of a window
top	Returns the topmost browser window

JavaScript

Methoden des Window-Objekts

Method	Description
<u>alert()</u>	Displays an alert box with a message and an OK button
<u>blur()</u>	Removes focus from the current window
<u>clearInterval()</u>	Clears a timer set with setInterval()
<u>clearTimeout()</u>	Clears a timer set with setTimeout()
<u>close()</u>	Closes the current window
<u>confirm()</u>	Displays a dialog box with a message and an OK and a Cancel button
<u>createPopup()</u>	Creates a pop-up window
<u>focus()</u>	Sets focus to the current window
<u>moveBy()</u>	Moves a window relative to its current position
<u>moveTo()</u>	Moves a window to the specified position
<u>open()</u>	Opens a new browser window
<u>print()</u>	Prints the content of the current window
<u>prompt()</u>	Displays a dialog box that prompts the visitor for input
<u>resizeBy()</u>	Resizes the window by the specified pixels
<u>resizeTo()</u>	Resizes the window to the specified width and height
<u>scroll()</u>	
<u>scrollBy()</u>	Scrolls the content by the specified number of pixels
<u>scrollTo()</u>	Scrolls the content to the specified coordinates
<u>setInterval()</u>	Calls a function or evaluates an expression at specified intervals (in milliseconds)
<u>setTimeout()</u>	Calls a function or evaluates an expression after a specified number of milliseconds

JavaScript

Die Methode `windows.open()`

- Syntax

```
window.open(URL, name, specs, replace)
```

Parameter	Description
URL	Optional. Specifies the URL of the page to open. If no URL is specified, a new window with <code>about:blank</code> is opened
name	Optional. Specifies the target attribute or the name of the window. The following values are supported: <ul style="list-style-type: none">• <code>_blank</code> - URL is loaded into a new window. This is default• <code>_parent</code> - URL is loaded into the parent frame• <code>_self</code> - URL replaces the current page• <code>_top</code> - URL replaces any framesets that may be loaded• <code>name</code> - The name of the window
specs	Optional. A comma-separated list of items. The following values are supported:
replace	Optional. Specifies whether the URL creates a new entry or replaces the current entry in the history list. The following values are supported: <ul style="list-style-type: none">• <code>true</code> - URL replaces the current document in the history list• <code>false</code> - URL creates a new entry in the history list

JavaScript

Die Methode `windows.open()`

- Option specs
 - Komma-separierte Liste folgender Optionen

<code>channelmode=yes no 1 0</code>	Whether or not to display the window in theater mode. Default is no. IE only
<code>directories=yes no 1 0</code>	Whether or not to add directory buttons. Default is yes. IE only
<code>fullscreen=yes no 1 0</code>	Whether or not to display the browser in full-screen mode. Default is no. A window in full-screen mode must also be in theater mode. IE only
<code>height=pixels</code>	The height of the window. Min. value is 100
<code>left=pixels</code>	The left position of the window
<code>location=yes no 1 0</code>	Whether or not to display the address field. Default is yes
<code>menubar=yes no 1 0</code>	Whether or not to display the menu bar. Default is yes
<code>resizable=yes no 1 0</code>	Whether or not the window is resizable. Default is yes
<code>scrollbars=yes no 1 0</code>	Whether or not to display scroll bars. Default is yes
<code>status=yes no 1 0</code>	Whether or not to add a status bar. Default is yes
<code>titlebar=yes no 1 0</code>	Whether or not to display the title bar. Ignored unless the calling application is an HTML Application or a trusted dialog box. Default is yes
<code>toolbar=yes no 1 0</code>	Whether or not to display the browser toolbar. Default is yes
<code>top=pixels</code>	The top position of the window. IE only
<code>width=pixels</code>	The width of the window. Min. value is 100

JavaScript

Die Methode `windows.open()`

- Beispiel

```
<html>
<head>
<title>Test</title>
<script type="text/javascript">
function FensterOeffnen (Adresse) {

    MeinFenster = window.open(Adresse, "Zweitfenster",
        "width=300,height=400,left=100,top=200"); MeinFenster.focus();

}
</script>
</head>
<body>
<p><a href="datei.htm" onclick="FensterOeffnen(this.href); return
    false">Link mit Fenster</a>
</p>
</body>
</html>
```

Internettechniken

JavaScript / AJAX / PHP

Prof. Dr. Jürgen Heym

Hochschule Hof

JavaScript / AJAX / PHP

Übersicht

- Aufgabenstellung

Programmieren Sie eine Beispielanwendung, die ein HTML-Eingabefeld für Postleitzahlen (PLZ) und eines für den Ortsnamen realisiert, das folgende Eigenschaften aufweist:

1. Sobald ein weiteres Zeichen in das Eingabefeld PLZ des Formulars eingegeben wird, werden passende Vorschläge für vorhandene Ortsnamen mit dieser PLZ unterbreitet. Maximal jedoch 7 Vorschläge.
2. Jeder Vorschlag soll als Link ausgeführt werden, der bei Betätigung die PLZ und den Ortsnamen in das Formular überträgt. Die Hinweise werden anschließend gelöscht.

Lösung --- HTML-Formular

```
<html>
<head>
<script src="1-plz.js" type="text/javascript"></script>
</head>
<body>

<h3>Geben Sie bitte ihre Postleitzahl ein:</h3>

<form action="">
PLZ: <input type="text" id="PLZ" onkeyup="showCity(this.value)" /><br />
Ort: <input type="text" id="ORT" />
</form>

</body>
</html>
```

JavaScript / AJAX / PHP

1-plz.js

Lösung --- AJAX / JavaScript (Variante 1 mit statischer PHP-Abfrage)

Dateinamen: 1-plz.html, 1-plz.js und 1-sucheORT.php

```
function showCity(str) {
    var xmlhttp;

    if (str.length==0) {
        document.forms[0].ORT.value="Ihre Stadt ...";
        return;
    }

    xmlhttp=new XMLHttpRequest();

    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {
            document.forms[0].ORT.value = xmlhttp.responseText;
        }
    }

    xmlhttp.open("GET","1-sucheORT.php?q="+str,true);
    xmlhttp.send();
}
```

JavaScript / AJAX / PHP

1-sucheORT.php

Lösung --- AJAX / JavaScript (Variante 1 mit statischer PHP-Abfrage)

Dateinamen: 1-plz.html, 1-plz.js und 1-sucheORT.php

```
<?php  
  
echo "1-sucheORT.php ...";  
  
?>
```

JavaScript / AJAX / PHP

2-plz.html

Lösung --- AJAX / JavaScript (Variante 2 mit statischer PHP-Abfrage)

Dateinamen: 2-plz.html, 2-plz.js und 2-sucheORT.php

```
<html>
<head>
<script src= "2-plz.js" type="text/javascript"></script>
</head>
<body>

<h3>Geben Sie bitte ihre Postleitzahl ein:</h3>

<form action="">
<table>
<tr><td>PLZ</td><td><input type="text" name="PLZ" onkeyup="showCity(this.value)"
    /></td></tr>
<tr><td>Ort</td><td><input type="text" name="ORT" /></td></tr>
</table>
</form>
<p>Vorschläge: <span id="txtVorschlag"></span></p>

</body>
</html>
```

JavaScript / AJAX / PHP

2-plz.js

Lösung --- AJAX / JavaScript (Variante 1 mit statischer PHP-Abfrage)

Dateinamen: 2-plz.html, 2-plz.js und 2-sucheORT.php

```
function showCity(str) {
    var xmlhttp;

    if (str.length==0) {
        document.forms[0].ORT.value="Ihre Stadt ...";
        return;
    } else {
        document.forms[0].ORT.value="";
    }

    xmlhttp=new XMLHttpRequest();

    xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200) {

            document.getElementById("txtVorschlag").innerHTML=xmlhttp.responseText;
        }
    }

    xmlhttp.open("GET", "2-sucheORT.php?plz="+str, true);
    xmlhttp.send();
}
```

JavaScript / AJAX / PHP

2-plz.html

Lösung --- AJAX / JavaScript (Variante 1 mit statischer PHP-Abfrage)

Dateinamen: 2-plz.html, 2-plz.js und 2-sucheORT.php

```
<?php

$plz = $_GET["plz"];

$out = "<p>";
$out .= "<ul>";
$out .= "<li>";
$out .= "<a href=' ' onclick='alert(\"Hallo Welt\")';>PLZ = $plz ...</a>";
$out .= "</li>";
$out .= "</ul>";
$out .= "</p>";

echo $out;

?>
```

WebDev 1

3 – AngularJS

Prof. Dr. Jürgen Heym

Hochschule Hof

1 –Angular & AngularJS

- *AngularJS is a structural framework for dynamic web apps.*
- *AngularJS lets you use HTML as your template language and lets you extend HTML's syntax to express your application's components clearly and succinctly.*
- *AngularJS is what HTML would have been, had it been designed for applications.*
- Latest stable release is 1.8.2 (Nov 12th, 2020).
- **AngularJS and Angular are incompatible!**

Source: Official website about AngularJS: <https://docs.angularjs.org/guide/introduction>

1 –Angular & AngularJS

- Sources / Material

1. Official AngularJS website
<https://angularjs.org/>
2. Wikipedia website about AngularJS
<https://en.wikipedia.org/wiki/AngularJS>
3. W3Schools Website about AngularJS
https://www.w3schools.com/angular/angular_intro.asp
4. AngularJS Code Directories
<https://code.angularjs.org/>

2 – What is AngularJS ?

- **AngularJS 1.X --- Development History & Releases**
 1. AngularJS was originally developed in 2009 by Miško Hevery at Brat Tech LLC and released in 2012 as AngularJS 1.0.
 2. Angular is an incompatible rewrite of AngularJS.
 3. Both AngularJS and Angular are currently supported.
 4. The current stable release of AngularJS is 1.8.2.

Source: Wikipedia website about AngularJS: <https://en.wikipedia.org/wiki/AngularJS>

2 – What is AngularJS ?

- **AngularJS / Angular.js / AngularJS 1.X --- Overview**
 - JavaScript-based open-source front-end web application framework
 - mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing **single-page (web) applications**
 - simplify both the development and the testing of such single-page applications by providing
 - a framework for client-side model–view–controller (**MVC**) architectures
 - a framework for model–view–viewmodel (**MVVM**) architectures,
 - components commonly used in rich Internet applications

Source: Wikipedia website about AngularJS: <https://en.wikipedia.org/wiki/AngularJS>

2 – What is AngularJS ?

■ AngularJS 1.X --- Paradigms

1. [Declarative programming](#) should be used to create user interfaces and connect software components
2. [Imperative programming](#) is better suited to defining an application's business logic.
3. Adapt and extend traditional HTML to present dynamic content through two-way data-binding.
4. Allow for the automatic synchronization of models and views.
5. Avoid explicit DOM manipulation with the goal of improving testability and performance.

Source: Wikipedia website about AngularJS: <https://en.wikipedia.org/wiki/AngularJS>

2 – What is AngularJS ?

■ AngularJS 1.X --- Design Goals

1. Decouple DOM manipulation from application logic.

The difficulty of this is dramatically affected by the way the code is structured.

2. Decouple the client side of an application from the server side.

This allows development work to progress in parallel, and allows for reuse of both sides.

3. Provide structure for the journey of building an application:

- designing the UI
- writing the business logic
- testing

Source: Wikipedia website about AngularJS: <https://en.wikipedia.org/wiki/AngularJS>

2 – What is AngularJS ?

■ AngularJS 1.X --- Implementation

1. Angular implements the MVC pattern to separate presentation, data, and logic components.
2. Angular uses [dependency injection](#).

Dependency injection is a technique whereby one object supplies the dependencies of another object. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern. [2]

3. Angular brings traditionally server-side services, such as view-dependent controllers, to client-side web applications.
4. Consequently, much of the burden on the server can be reduced.

Sources:

[1] Wikipedia website about AngularJS: <https://en.wikipedia.org/wiki/AngularJS>

[2] Wikipedia website about Dependency Injection: https://en.wikipedia.org/wiki/Dependency_injection

3 – Hello World Script

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>AngularJS --- Hello World 1 (myApp-1.html)</title>
  <script src="https://ajax.googleapis.com/ajax/libs/
                                angularjs/1.8.2/angular.min.js">
  </script>
</head>
<body>

<div ng-app="">
  <p>Give me your firstname : <input type="text" ng-model="firstname"></p>
  <h1>Hello {{firstname}}</h1>
</div>

</body>
</html>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp
You find this code in Moodle: myApp-1.html

3 – Hello World Script

- AngularJS is a Javascript Framework

```
<script src="https://ajax.googleapis.com/ajax/libs/  
angularjs/1.8.2/angular.min.js"></script>
```

- AngularJS extends HTML with ng-directives

- The **ng-app** directive defines an AngularJS application: `<div ng-app="">`
- The **ng-model** directive binds the value of HTML controls (input, select, textarea) to application data:
`<input type="text" ng-model="firstname">`
- Use `{{ }}` for ng-expressions.
- The **ng-bind** directive binds application data to the innerHTML view:
``
- The **ng-init** directive initializes AngularJS application variables:
`<div ng-app="" ng-init="firstname='Jürgen'">`

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

3 – Hello World Script 2

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>AngularJS --- Hello World 2 (myApp-2.html)</title>
  <script src="https://ajax.googleapis.com/ajax/libs/
                                angularjs/1.8.2/angular.min.js">
  </script>
</head>
<body>

<div ng-app="" ng-init="firstName='Angela'">
  <p>Your firstname is <span ng-bind="firstName"></span></p>
  <p>Give me your lastname: <input type="text" ng-model="lastName"></p>
  <p>Your name is {{firstName + " " + lastName}}</p>
</div>

</body>
</html>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp
You find this code in Moodle: myApp-2.html

3 – Hello World Script 3

- You can write expressions wherever you like, AngularJS will simply resolve the expression and return the result.
 - For example: Let AngularJS change the value of CSS properties.

...

```
<div ng-app="" ng-init="myColor='lightblue' ">  
  
    <input style="background-color: {{myColor}} "  
          ng-model="myColor"  
          value="{{myColor}} ">  
  
</div>
```

...

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp
You find this code in Moodle: myApp-3.html

4 – AngularJS Datatypes

- AngularJS numbers are like JavaScript numbers

```
<div ng-app="" ng-init="quantity=1;cost=5">  
  <p>Total price in Euro is {{ quantity * cost }} </p>  
</div>
```

- AngularJS strings are like JavaScript strings

```
<div ng-app="" ng-init="firstName='Joseph';lastName='Biden'">  
  <p>Mr. Biden's fullname is {{ firstName + " " + lastName }} </p>  
</div>
```

- AngularJS objects are like JavaScript objects

```
<div ng-app="" ng-init="person={fName:'Angela',lName:'Merkel'} ">  
  <p>Angelas lastname is {{ person.lName }} </p>  
</div>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

4 – AngularJS Datatypes

- AngularJS arrays are like JavaScript arrays

```
<div ng-app="" ng-init="pointList=[1,15,19,2,40]">  
  <p>The third result is {{ pointList[2] }}</p>  
</div>
```

- AngularJS Expressions vs. JavaScript Expressions
 - Like JavaScript expressions, AngularJS expressions can contain literals, operators, and variables.
 - Unlike JavaScript expressions, AngularJS expressions can be written inside HTML.
 - AngularJS expressions do not support conditionals, loops, and exceptions, while JavaScript expressions do.
 - AngularJS expressions support filters, while JavaScript expressions do not.

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

5 – AngularJS Modules

- An AngularJS module defines an application.
- The module is a container for the different parts of an application.
- The module is a container for the application controllers.
- Controllers always belong to a module.

- A module is created by using the AngularJS function `angular.module`

```
<div ng-app="myApp">...</div>
```

```
<script>  
    var app = angular.module("myApp", []);  
</script>
```

- The "myApp" parameter refers to an HTML element in which the application will run.

- Now you can add controllers, directives, filters, ... to your AngularJS application.

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

6 – AngularJS Controllers

- Add a controller to your application, and refer to the controller with the ng-controller directive.

```
<div ng-app="myApp" ng-controller="myCtrl">
    {{ firstName + " " + lastName }}
</div>

<script>

    var app = angular.module("myApp", []);

    app.controller("myCtrl", function($scope) {
        $scope.firstName = "Adam";
        $scope.lastName = "Quincey";
    });

</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

6 – AngularJS Controllers

- A controller may have methods. Add a second controller with a function `fullname()` to your app.

```
<div ng-app="myApp" ng-controller="myCtrl2">
  Fullname: {{ fullname() }}
</div>
```

```
<script>
```

```
var app = angular.module("myApp", []);

app.controller("myCtrl2", function($scope) {
  $scope.firstName = "Adam";
  $scope.lastName = "Quincey";
  $scope.fullName = function() {
    return $scope.firstName + " " + $scope.lastName;
  };
});
```

```
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

7 – AngularJS Modules & Controllers in Files

- It's good programming style to put the module and the controllers in JavaScript files.

```
<!DOCTYPE html>
<html lang="en-US">
<head>
  <meta charset="UTF-8">
  <title>AngularJS --- Modules & Controllers 1</title>
  <script src="https://.../1.8.2/angular.min.js"></script>
</head>
<body>

<div ng-app="myApp" ng-controller="myCtrl">
  {{ firstName + " " + lastName }}
</div>

<script src="myModule.js"></script>
<script src="myControllers.js"></script>

</body>
</html>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

8 – AngularJS Directives

- AngularJS has a set of built-in directives which you can use to add functionality to your application: [AngularJS directive reference](#)
- Now, just add your own directives!

```
<div ng-app="myApp" my-test-directive></div>
```

```
<script>
```

```
var app = angular.module("myApp", []);

app.directive("myTestDirective", function() {
    return {
        template : "The constructor made this!"
    };
});
```

```
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

8 – AngularJS Directives

- Invoke your own directives using one of the following really equivalent methods:
 - Element name (E) `<my-test-directive></my-test-directive>`
 - Attribute (A) `<div my-test-directive></div>`
 - Class (C) `<div class="my-test-directive"></div>`
 - Comment (M) `<!-- directive: my-test-directive -->`

- You can restrict the use cases of your directives:
 - ✓ Valid restrictions are E, A, C, and M.
 - ✓ Default are restrictions E and A.
 - ✓ Restriction M needs `replace="true"`.

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

9 – AngularJS Scopes

- The scope is the **binding part between** the HTML (**view**) and the JavaScript (**controller**).
- The scope is an object with the available properties and methods.
- The scope is available for both the view and the controller.
- How do we use the scope?
 - When you make a controller in AngularJS, you pass the \$scope object as an argument.
 - Properties made in the controller, can be referred to in the view.
 - In the view, you do not use the prefix \$scope, you just refer to a propertyname, like {{carname}}.
 - We used this in the preceding examples!

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

9 – AngularJS Scopes

- If we consider an AngularJS application to consist of
 - a View, which is the HTML,
 - a Model, which is the data available for the current view,
 - a Controller, which is the JavaScript function that makes/changes/removes/controls the data,

then the scope is the Model.

- Remember, the scope is a JavaScript object with properties and methods, which are available for both the view and the controller.
- If you make changes in the view, the model and the controller will be updated!

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

9 – AngularJS Scopes

■ Root Scope

- All applications have a `$rootScope` which is the scope created on the HTML element that contains the `ng-app` directive.
- The `rootScope` is available in the entire application.
- If a variable has the same name in both the current scope and in the `rootScope`, the application use the one in the current scope.
- Examples

```
app.run(function($rootScope) {
    $rootScope.color = 'blue';
});
app.controller('myCtrl', function($scope) {
    $scope.color = "red";
});
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

9 – AngularJS Scopes

```
<body ng-app="myApp">
```

```
<p>We are in the rootScope. The rootScope's favorite color is:</p>
```

```
<h1 style="color:{{color}}">{{color}}</h1>
```

```
<div ng-controller="myCtrl">
```

```
  <p>Now, we are in the the scope of the controller.
```

```
  The controller's favorite color is:</p>
```

```
  <h1 style="color:{{color}}">{{color}}</h1>
```

```
</div>
```

```
<p>Back to the rootScope. The rootScope's favorite color is still:</p>
```

```
<h1 style="color:{{color}}">{{color}}</h1>
```

```
<script>
```

```
  var app = angular.module('myApp', []);
```

```
  app.run(function($rootScope) { $rootScope.color = 'blue'; });
```

```
  app.controller('myCtrl', function($scope) { $scope.color = "red"; });
```

```
</script>
```

```
</body>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

10 – AngularJS Routing

- The ngRoute module helps your application to become a **Single Page Application (SPA)**.
- What is Routing in AngularJS?
 - If you want to navigate to different pages in your application, but you also want the application to be a SPA, with no page reloading, you can use the ngRoute module.
 - The ngRoute module routes your application to different pages without reloading the entire application.
- This is the essential feature that we want to have, a **Single Page Application!**

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

10 – AngularJS Routing

- How can we use the Route Modul in AngularJS?

1. Include the AngularJS Route module

```
<script src="https://...../1.8.2/angular-route.js"></script>
```

2. Add the ngRoute as a dependency in the application module. This provides the \$routeProvider.

```
var app = angular.module("myApp", ["ngRoute"]);
```

3. Use the \$routeProvider to configure different routes in your application.
4. Define the \$routeProvider using the config method of your application. Work registered in the config method will be performed when the application is loading.

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

10 – AngularJS Routing

```
<body ng-app="myApp">

<a href="#/">Main</a>
<a href="#!london">London</a>      <!-- this is the navigation -->
<a href="#!paris">Paris</a>

<div ng-view></div>      <!-- or:  <ng-view></ng-view>      -->

<script>
  var app = angular.module("myApp", ["ngRoute"]);
  app.config(function($routeProvider) {
    $routeProvider
      .when("/", {
        templateUrl : "main.html"
      })
    ... snip-snap ...
      .when("/paris", {
        templateUrl : "paris.html"
      });
  });
</script>

</body>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

10 – AngularJS Routing

- How can we use the Route Modul in AngularJS?

5. With the `$routeProvider` you can also define a controller for each "view".

```
app.config(function($routeProvider) {  
  $routeProvider  
    .when("/london", {  
      templateUrl : "london.html",  
      controller : "londonCtrl"  
    })  
    .otherwise({templateUrl : "main.html"});  
});
```

6. The "london.htm" and "paris.htm" are normal HTML files, which you can add AngularJS expressions as you would with any other HTML sections of your AngularJS application.

```
app.controller("londonCtrl", function ($scope) {  
  $scope.msg = "I love London!";  
});
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

11 – AngularJS Model

- The ngModel directive binds an input, select, textarea (or custom form control) to a property on the scope using NgModelController, which is created and exposed by this directive.
 - The value of an input field is bound to a variable created in AngularJS.
 - Access this variable in your controller using the \$scope.
 - Two-way data binding means the binding goes forth and back.
If you change the name in the input field, the AngularJS variable is changed as well and vice versa.

```
<div ng-app="myApp" ng-controller="myCtrl">
    Name: <input ng-model="name">
</div>
<script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope) {
        $scope.name = "John Doe";
    });
</script>
```

Sources:

- [1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_model
- [2] AngularJS Reference: <https://docs.angularjs.org/api/ng/directive/ngModel>

11 – AngularJS Model

- Validation of user input
 - The ng-model directive provides type validation for application data (number, e-mail, required).
 - The span will be displayed only if the expression in the ng-show attribute returns true.

```
<body ng-app="">
```

```
<form name="myForm">
```

```
Email:
```

```
<input type="email" name="myAddress" ng-model="text">
```

```
<span ng-show="myForm.myAddress.$error.email">
```

```
    Not a valid e-mail address
```

```
</span>
```

```
</form>
```

```
</body>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_model_validate

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/directive/ngModel>

11 – AngularJS Model

- Validation of user input
 - The ng-model directive provides four different status for the application data:
 - ✓ invalid, dirty, touched, error

```
<body ng-app="">
<form name="myForm">
Email:
  <input type="email" name="myAddress" ng-model="text">
  <p>Edit the e-mail address, and observe the status.</p>
  <h1>Status</h1>
  <p>Valid: {{myForm.myAddress.$valid}} </p>
  <p>Dirty: {{myForm.myAddress.$dirty}} </p>
  <p>Touched: {{myForm.myAddress.$touched}} </p>
</form>
</body>
```

Sources:

- [1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_model_status
- [2] AngularJS Reference: <https://docs.angularjs.org/api/ng/directive/ngModel>

11 – AngularJS Model

- Validation of user input
 - The ng-model directive provides CSS classes for HTML elements, depending on their status.
 - The ng-model directive adds/removes the following classes, according to the status of the form field:
 - ✓ ng-empty, ng-not-empty, ng-touched, ng-untouched, ng-valid, ng-invalid, ng-dirty, ng-pending and ng-pristine
 - ✓ AngularJS Ref: <https://docs.angularjs.org/api/ng/directive/ngModel>

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_model_css

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/directive/ngModel>

12 – AngularJS Filters

- Use the following AngularJS filters to transform data:

Name	Description
filter	Selects a subset of items from array and returns it as a new array.
currency	Formats a number as a currency (ie \$1,234.56). When no currency symbol is provided, default symbol for current locale is used.
number	Formats a number as text.
date	Formats date to a string based on the requested format.
json	Allows you to convert a JavaScript object into JSON string.
lowercase / uppercase	Converts string to lowercase / uppercase.
limitTo	Creates a new array or string containing only a specified number of elements. ...
orderBy	Returns an array containing the items from the specified collection, ordered by a comparator function based on the values computed using the expression predicate.

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_model_status

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/filter>

12 – AngularJS Filters

- Adding filters to expressions and directives.

```
<div ng-app="myApp" ng-controller="namesCtrl">

<p>Looping with objects:</p>
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name | uppercase}}, {{ x.country }}
  </li>
</ul>

</div>
<script>
  app = angular.module('myApp', []);
  app.controller('namesCtrl', function($scope) {
    $scope.names = [
      {name: 'Steinmeier', country: 'Germany'},
      {name: 'Macron', country: 'France'}
    ];
  });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/filter>

12 – AngularJS Filters

- Filter an array based on user input.

```
<div ng-app="myApp" ng-controller="namesCtrl">
<p>
  Type a letter in the input field:
  <input type="text" ng-model="yourInput">
</p>
<ul>
  <li ng-repeat="x in names | filter:yourInput">
    {{ x }}
  </li>
</ul>
</div>
<script>
  app = angular.module('myApp', []);
  app.controller('namesCtrl', function($scope) {
    $scope.names = [ 'Jani', 'Carl' ]; });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/filter>

12 – AngularJS Filters

- Sort an array based on user input.
 - Run a function that changes the sorting order of an array by adding the ng-click directive on the table headers.
 - Then, click the table headers to change the sort order.

```
<div ng-app="myApp" ng-controller="namesCtrl">
  <table border="1" width="100%">
    <tr>
      <th ng-click="orderByMe('name')">Name</th>
      <th ng-click="orderByMe('country')">Country</th>
    </tr>
    <tr ng-repeat="x in names | orderBy:myOrderBy">
      <td>{{x.name}}</td>
      <td>{{x.country}}</td>
    </tr>
  </table>
</div>
```

Sources:

- [1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby_click
[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/filter>

12 – AngularJS Filters

- Sort an array based on user input (continued).
 - Run a function that changes the sorting order of an array by adding the ng-click directive on the table headers.
 - Then, click the table headers to change the sort order.

```
<script>
  app = angular.module('myApp', []);
  app.controller('namesCtrl', function($scope) {
    $scope.names = [
      {name: 'Jani', country: 'Norway'},
      {name: 'Carl', country: 'Sweden'},
      {name: 'Margareth', country: 'England'},
    ];
    $scope.orderByMe = function(x) {
      $scope.myOrderBy = x;
    }
  });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_filters_orderby

[2] AngularJS Reference: <https://docs.angularjs.org/api/ng/filter>

13 – AngularJS Services

- AngularJS has a couple of built-in service components (≈ 30).
 - See AngularJS Ref: <https://docs.angularjs.org/api/ng/service>
 - A service is a function, or object, that is available for, and limited to, your AngularJS application.
- \$location Service
 - The built-in \$location service has methods which return information about the location of the current web page, e.g. the absolute URL, the path, the host and many more.
 - AngularJS constantly supervises your application, and for it to handle changes and events properly, AngularJS prefers that you use the \$location service instead of the window.location object.

```
<div ng-app="myApp" ng-controller="myCtrl">  
  <p>The url of this page is: {{myUrl}}</p>  
</div>
```

```
<script>  
  var app = angular.module('myApp', []);  
  app.controller('myCtrl', function($scope, $location) {  
    $scope.myUrl = $location.absUrl();  
  });  
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_services

[2] AngularJS Reference: [https://docs.angularjs.org/api/ng/service/\\$location](https://docs.angularjs.org/api/ng/service/$location)

13 – AngularJS Services

▪ \$http Service

- The \$http service wraps AJAX requests in AngularJS applications.
- The service makes a request to the server, and lets your application handle the response.
- See AngularJS Ref: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)
- The \$http service is a core AngularJS service that facilitates communication with the remote HTTP servers via the browser's XMLHttpRequest object or via JSONP.

```
<div ng-app="myApp" ng-controller="myCtrl">
    <p>The message of the day is: {{motd}}</p>
</div>
```

```
<script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope, $http) {
        $http.get("myMotd.htm").then(function (response) {
            $scope.motd = response.data; });
    });
</script>
```

Sources:

- [1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_services_http
- [2] AngularJS Reference: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

13 – AngularJS Services

▪ Details of \$http Service

- The example executes the \$http service with an object as an argument. The object is specifying the HTTP method, the url, what to do on success, and what to do on failure.
- The response from the server is an object with five properties:
 - ✓ .config the object used to generate the request,
 - ✓ .data a string, or an object, carrying the response from the server,
 - ✓ .headers a function to use to get header information,
 - ✓ .status a number defining the HTTP status,
 - ✓ .statusText a string defining the HTTP status.

```
<script>
  var app = angular.module('myApp', []);
  app.controller('myCtrl', function($scope, $http) {
    $http.get(
      { method: "GET", url: "myMotd.htm« }
    ).then(
      function mySuccess(response) {
        $scope.motd = response.data; },
      function myError(response) {
        $scope.motd = response.statusText; }
    );
  });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_services_http

[2] AngularJS Reference: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

13 – AngularJS Services

- Create your own service.
 1. Connect your own service to your app.
 2. Use your custom service within your controllers adding the dependency.

```
<script>
  var app = angular.module('myApp', []);
  app.service('hexify', function() {
    this.toHex = function (x) {return x.toString(16);} });

  app.controller('myCtrl', function($scope, hexify) {
    $scope.hexNum = hexify.toHex(145);
  });
</script>
```

Sources:

- [1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_services_http
[2] AngularJS Reference: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

13 – AngularJS Services

■ Example Responsedata

1. The response data are expected to be in JSON format!
2. Example: The server (locations.php) returns a JSON-Object containing 2 locations.

```
<div ng-app="myApp" ng-controller="locationsCtrl">
  <ul>
    <li ng-repeat="x in myDataProvider">
      {{ x.Name + ', ' + x.City }}
    </li>
  </ul>
</div>

<script>
  var app = angular.module('myApp', []);
  app.controller('locationsCtrl', function($scope, $http) {
    $http.get("locations.php").then(function (response) {
      $scope.myDataprovider = response.data.records;
    });
  });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/tryit.asp?filename=try_ng_services_http

[2] AngularJS Reference: [https://docs.angularjs.org/api/ng/service/\\$http](https://docs.angularjs.org/api/ng/service/$http)

13 – AngularJS Services

- Example Responsedata (locations.php)

```
<?php
```

```
$resultSet = '{  
  "records": [  
    {"Name": "Loci Loft", "City": "Berlin", "Country": "Germany"} ,  
    {"Name": "Spring Stone", "City": "London", "Country": "Brexit"}  
  ]  
';
```

```
echo $resultSet;
```

```
?>
```

- Next example is to move your data in a MySQL server and fetch them using PHP.
https://www.w3schools.com/angular/angular_sql.asp

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_sql.asp

14 – AngularJS Events

- Event directives allows us to run AngularJS functions at certain user events.
- Ng-events do not overwrite HTML events, both events will be executed!
- Add AngularJS event listeners to HTML elements by using one or more of the following directives:

+ ng-click	+ ng-mouseenter
+ ng-copy	+ ng-mouseleave
+ ng-cut	+ ng-mousemove
+ ng-dblclick	+ ng-mouseover
+ ng-focus	+ ng-mouseup
+ ng-keydown	+ ng-mousedown
+ ng-keypress	
+ ng-keyup	
+ ng-paste	
+ ng-change	
+ ng-blur	

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_events.asp

14 – AngularJS Mouse Events

- Take care of the order in which mouse events occur!

- Mouse events occur when the cursor moves over an element.
The sequence mouse move events are treated is the following:
 1. ng-mouseenter
 2. ng-mouseover
 3. ng-mousemove
 4. ng-mouseleave

- Take care
 1. ng-mouseup
 2. ng-mousedown
 3. ng-click

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_events.asp

14 – AngularJS Mouse Events

- Example: <http://cosd-fe.lx-lehre.hof-university.de/#/mouse-simple>

```
<div ng-app="myApp" ng-controller="myCtrl">

    <button ng-click="myFunc()">Click Me!</button>

    <div ng-show="showMe">
        <p>Text blablabla ... </p>
    </div>
</div>

<script>
    var app = angular.module('myApp', []);
    app.controller('myCtrl', function($scope) {
        $scope.showMe = false;
        $scope.myFunc = function() {
            $scope.showMe = !$scope.showMe; }
    });
</script>
```

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_events.asp

40 – AngularJS Application

- If we have enough time, we will implement the Angular Application from W3-Schools website:

https://www.w3schools.com/angular/angular_application.asp

Sources:

[1] W3Schools Website about AngularJS: https://www.w3schools.com/angular/angular_intro.asp

Web Development 1

jQuery Part 1

Prof. Dr. Jürgen Heym

Hof University of Applied Sciences

What is jQuery?

- jQuery is a fast, small JavaScript library having a lot of different features.
- jQuery simplifies
 - the access to HTML elements,
 - the manipulation of HTML elements,
 - event handling,
 - animation of HTML elements
- jQuery simplifies Ajax providing a much simpler browser independent interface.
- jQuery is usable in various ways and extensible.
- jQuery was a revolution to JavaScript.

- DOM & Manipulation of HTML elements

Load the <button> element of class 'continue' and change the HTML text to be “continue”

```
$( "button.continue" ).html("continue" )
```


- Event handling

Show the #banner message element hidden using display:none as soon as any element with id #button-container or class button was clicked.

```
var hiddenBox = $( "#banner-message" );  
  
$( "#button-container button" ).on( "click", function( event ){  
    hiddenBox.show(); } );
```

- AJAX

Call the script `/api/getWeather` on your webserver with parameter `zipcode=95028` and replace element `#weather-temp` with the response text.

```
$.ajax({  
    url: "/api/getWeather",  
    data: { zipcode: 95028 },  
    success: function( data ) {  
        $( "#weather-temp" ).html( "<b>" + data + "</b> degrees" );  
    }  
});
```

- Original documentation to the jQuery API you'll find here

<http://api.jquery.com/>
- Versioning
 - jQuery 1.x.x supports most browsers
 - jQuery 2.x.x no support for IE 6, 7 + 8
 - jQuery 3.6.0 today's version as of 3rd of May 2022
- jQuery is provided as compressed and uncompressed library for production and development.
- All versions of jQuery below version 3.0 are marked with status „deprecated“ !

- jQuery usage
 - Content Distribution Network (CDN)

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"  
  integrity="sha256-/xUj+3OJU5yExlq6GSYGHk7tPXikynS7ogEvDej/m4="\  
  crossorigin="anonymous"></script>
```

Wurde jQuery richtig eingebunden?

- jQuery usage
 - Use the following code snippet to check if jQuery is loaded successfully.

```
if(typeof(jQuery) !== undefined){  
    alert('jQuery loaded');  
}
```

Web Development 1

jQuery Part 2 --- Getting Started

Prof. Dr. Jürgen Heym

Hochschule Hof

- With jQuery we select HTML element and start action on these elements.

`$(selector) .action()`

- The \$ character leads us to the jQuery library.
- The selector permits to specify the HTML element(s).
- The jQuery action operates on the selected HTML element(s).

- Examples

Hide the actual element:

```
$(this).hide()
```

Hide all <p> elements:

```
$("p").hide()
```

Hide all elements of class=„test“:

```
$(".test").hide()
```

Hide all element with id=„test“:

```
$("#test").hide()
```

jQuery

Document Ready Event

- In order to prevent the execution of jQuery-Code before the document is loaded completely, we use the „Document Ready Event“.

All jQuery methods will be called in a save way:

```
$(document).ready(function() {  
    // jQuery methods go here...  
});
```

Short version:

```
$(function() {  
    // jQuery methods go here...  
});
```


jQuery

Selectors

- jQuery selectors permit the selection and manipulation HTML elements.
- jQuery selectors are used to select HTML elements based on their ID, class, type, attribute, attribute values and much more.
- All jQuery selectors start with the \$-character and are enclosed by parens: \$().
- Example: element selector

```
$(document).ready(function() {  
    $("button").click(function() {  
        $("p").hide();  
    });  
});
```

As soon as an user clicks on any button, all <p> elements will be hidden.

jQuery

Selectors

- Example: Id selector

```
$(document).ready(function() {  
    $("button").click(function() {  
        $("#test").hide();  
    });  
});
```

As soon as an user clicks on any button, the element with id=„test“ will be hidden.

jQuery

Selectors

- Example: Class selector

```
$(document).ready(function() {  
    $("button").click(function() {  
        $(".test").hide();  
    });  
});
```

As soon as an user clicks on any button, all elements with class=„test“ will be hidden.

jQuery

Selectors

- For more examples see http://www.w3schools.com/jquery/jquery_selectors.asp

Syntax	Selektion
<code>\$ ("*")</code>	all elements
<code>\$ (this)</code>	actual element
<code>\$ ("p.intro")</code>	all <p> elements with class="intro"
<code>\$ ("p:first")</code>	the first <p> element
<code>\$ ("ul li:first")</code>	the first element of the first element
<code>\$ ("ul li:first-child")</code>	the first element all elements
<code>\$ (" [href] ")</code>	all elements with HREF attribute
<code>\$ ("a [target=' _blank'] ")</code>	all <a> elements with TARGET attribute „_blank“
<code>\$ (":button")</code>	all <button> elements and <input> elements with type=„button“
<code>\$ ("tr:even")</code>	all even <tr> elements
<code>\$ ("tr:odd")</code>	all odd <tr> elements

- JQuery code is JavaScript. Of course, you may store your jQuery source code in external files.

```
<head>  
<script src="../../../jquery/3.6.0/jquery.min.js"></script>  
<script src="my_jquery_functions.js"></script>  
</head>
```

- Website visitors fire events:
 - mouse: click, dblclick, mouseenter, mouseleave, ...
 - keyboard: keypress, keydown, keyup, ...
 - forms: submit, change, focus, blur, ...
 - documents: load, unload, ...
 - windows: resize, scroll, ...

- In jQuery nearly all DOM events are handled by equivalent jQuery methods.

- Examples

```
$("#p1").mouseenter(function() {  
    alert("You entered p1!");  
});
```

```
$("#p1").mouseleave(function() {  
    alert("Bye! You now left p1!");  
});
```

```
$("#p1").mousedown(function() {  
    alert("Mouse down over p1!");  
});
```

```
$("#p1").hover(function() {  
    alert("You entered p1!");  
},  
function() {  
    alert("Bye! You now left p1!");  
});
```

- The jQuery `on()` method allows to activate several event handlers for an element:

```
$( "p" ).on({  
    mouseenter: function() {  
        $(this).css("background-color", "lightgray");  
    },  
    mouseleave: function() {  
        $(this).css("background-color", "lightblue");  
    },  
    click: function() {  
        $(this).css("background-color", "yellow");  
    }  
});
```


jQuery

Hide, show and toggle elements

- methods `hide()` and `show()` are used to hide respectively show elements.
- Method `toggle()` shows hidden elements and hides shown elements.
- Syntax

```
$(selector).hide([options]);  
$(selector).show([options]);  
$(selector).toggle([options]);
```

- See jQuery reference “Basic Effects”

<http://api.jquery.com/category/effects/basics/>

- Example
 - Hiding a paragraph slowly when clicking on the paragraph.

```
$(document).ready(function() {  
    $("p").click(function() {  
        $(this).hide("slow");  
    });  
});
```

- Example
 - Toggle

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
<script>
$(document).ready(function() {
    $("button").click(function() {
        $("p").toggle();
    });
});
</script>
</head>
<body>
<button>Toggle</button>
<p>This is a paragraph.</p>
</body>
</html>
```

Changing the Visibility of Elements

- jQuery knows about four methods to fade in and out
- Syntax

```
$(selector).fadeIn(options);  
$(selector).fadeOut(options);  
$(selector).fadeToggle(options);  
$(selector).fadeTo(options);
```

- Example

```
<script>
  $(document).ready(function() {
    $("#button").click(function() {
      $("#div1").fadeIn();
      $("#div2").fadeIn("slow");
      $("#div3").fadeIn(3000);
    });
  });
</script>
...
<p>Demonstrate fadeIn() with different parameters.</p>
<button>Click to fade in boxes</button><br><br>

<div id="div1" style="width:80px;height:80px;display:none;background-color:red;"></div><br>

<div id="div2" style="width:80px;height:80px;display:none;background-color:green;"></div><br>

<div id="div3" style="width:80px;height:80px;display:none;background-color:blue;"></div>
```

jQuery

Opening and Closing Elements

- jQuery knows about three methods to open and close elements.

- Syntax

```
$(selector).slideDown(options);  
$(selector).slideUp(options);  
$(selector).slideToggle(options);
```

- Examples

<http://api.jquery.com/category/effects/sliding/>

- jQuery methode `animate()` to animate arbitrary, even several CSS properties at the same time.
- Syntax

```
$(selector).animate({params}, options);
```

- Example:

Move a DIV element to the right until property `left` has the value `250px`.

```
$("#button").click(function() {  
    $("#div").animate({left: '250px'});  
});
```

- Need more examples?

<http://api.jquery.com/animate/>

- Method `animate()` knows about concatenation of animations.
- Example:

```
$("#button").click(function() {  
    var div = $("#div");  
    div.animate({height: '300px', opacity: '0.4'}, "slow");  
    div.animate({width: '300px', opacity: '0.8'}, "slow");  
    div.animate({height: '100px', opacity: '0.4'}, "slow");  
    div.animate({width: '100px', opacity: '0.8'}, "slow");  
});
```

- Need more examples?
<http://api.jquery.com/animate/>

jQuery

Stopping Animations

- Method `stop()` stops an animation before it is completed.
- Example:

```
$("#stop").click(function() {  
    $("#panel").stop();  
});
```

- More examples

<http://api.jquery.com/stop/>

- Example:

```
$("#button").click(function() {  
    $("#p1").css("color", "red").slideUp(2000).slideDown(2000);  
});
```

- More examples

<http://api.jquery.com/queue/>

- jQuery has useful methods to manipulate DOM properties:
 - `text()` Get the combined text contents of each element in the set of matched elements, including their descendants, or set the text contents of the matched elements.
 - `html()` Get the HTML contents of the first element in the set of matched elements or set the HTML contents of every matched element.
 - `val()` Get the current value of the first element in the set of matched elements or set the value of every matched element.
 - `attr()` Get the value of an attribute for the first element in the set of matched elements or set one or more attributes for every matched element.
- Examples

<http://api.jquery.com/text/>

<http://api.jquery.com/html/>

<http://api.jquery.com/val/>

<http://api.jquery.com/attr/>

- jQuery has methods to remove HTML content and nodes:
 - `remove()` Remove the set of matched elements from the DOM.
 - `empty()` Remove all child nodes of the set of matched elements from the DOM.

- Examples

<http://api.jquery.com/remove/>

<http://api.jquery.com/empty/>

- jQuery has methods to manipulate CSS properties:
 - `addClass()` Adds the specified class(es) to each element in the set of matched elements.
 - `removeClass()` Remove a single class, multiple classes, or all classes from each element in the set of matched elements.
 - `toggleClass()` Add or remove one or more classes from each element in the set of matched elements, depending on either the class's presence or the value of the state argument.
 - `css()` Get the value of a computed style property for the first element in the set of matched elements or set one or more CSS properties for every matched element.

- Examples

<http://api.jquery.com/addClass/>

<http://api.jquery.com/removeClass/>

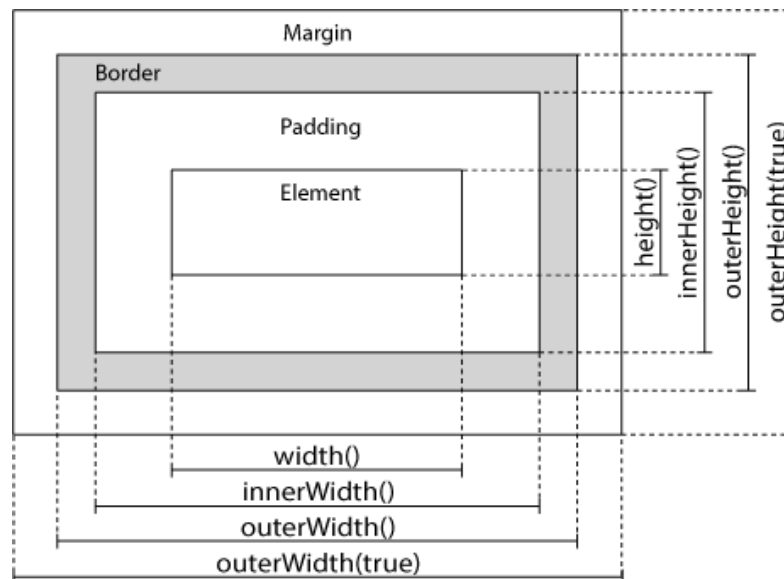
<http://api.jquery.com/toggleclass/>

<http://api.jquery.com/css/>

- jQuery has methods to manipulate width and height of elements:

- Syntax

- `width()`
- `height()`
- `innerWidth()`
- `innerHeight()`
- `outerWidth()`
- `outerHeight()`



- Examples

<http://api.jquery.com/category/manipulation/>

- jQuery has methods to search & find elements in the DOM structure of a document.
 - Syntax
 - parent()
 - parents()
 - parentsUntil()
 - children()
 - find()
 - siblings()
 - next() / prev()
 - nextAll() / prevAll()
 - nextUntil() / prevUntil()
 - first() / last() / eq() / filter() / not()

- Use method `load()` to load data from the server and place the returned HTML into the matched element.

- Syntax

```
$(selector).load(url[,data][,complete]);
```

- Examples

1. Load file `demo_test.txt` in a specific DIV element:
`$("#div1").load("demo_test.txt");`
2. Load element with `id=„p1“` from file `demo_test.txt` in a specific element:
`$("#div1").load("demo_test.txt #p1");`
3. <http://api.jquery.com/category/ajax/>

- Another example about .load()

```
$("#button").click(function(){
    $("#div1").load("demo_test.txt", function(responseTxt, statusTxt, xhr){
        if(statusTxt == "success")
            alert("External content loaded successfully!");
        if(statusTxt == "error")
            alert("Error: " + xhr.status + ": " + xhr.statusText);
    });
});
```

- responseTxt Contains the response of the AJAX call.
- statusTxt Contains the status of the AJAX call.
- Xhr Contains the XMLHttpRequest object of the AJAX call.

- Methods `get()` and `post()` are used to get the data of GET resp. POST requests.

- Syntax

```
$.get(URL, callback);  
$.post(URL, data, callback);
```

- Example

```
$("#button").click(function() {  
    $.post("demo_test_post.asp",  
    {  
        name: "Donald Duck",  
        city: "Duckburg"  
    },  
    function(data, status){  
        alert("Data: " + data + "\nStatus: " + status);  
    });  
});
```

Literature

- jQuery Foundation API
<http://api.jquery.com>
- w3schools jQuery Tutorial
<http://www.w3schools.com/jquery/>



PennState

College of Information
Sciences and Technology

IST 256

NodeJS

Install, Setup, Run



NodeJS: Download and Install

- Download <https://nodejs.org/en/download/>
- Pick your Operating System
- Select LTS Long Term Support




node

HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION

Downloads

Latest LTS Version: 12.13.0 (includes npm 6.12.0)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer <small>node-v12.13.0-x64.msi</small>	 macOS Installer <small>node-v12.13.0.pkg</small>	 Source Code <small>node-v12.13.0.tar.gz</small>
Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit	
macOS Binary (.tar.gz)	64-bit	
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v12.13.0.tar.gz	

Node.js Setup

Welcome to the Node.js Setup Wizard

The Setup Wizard will install Node.js on your computer.

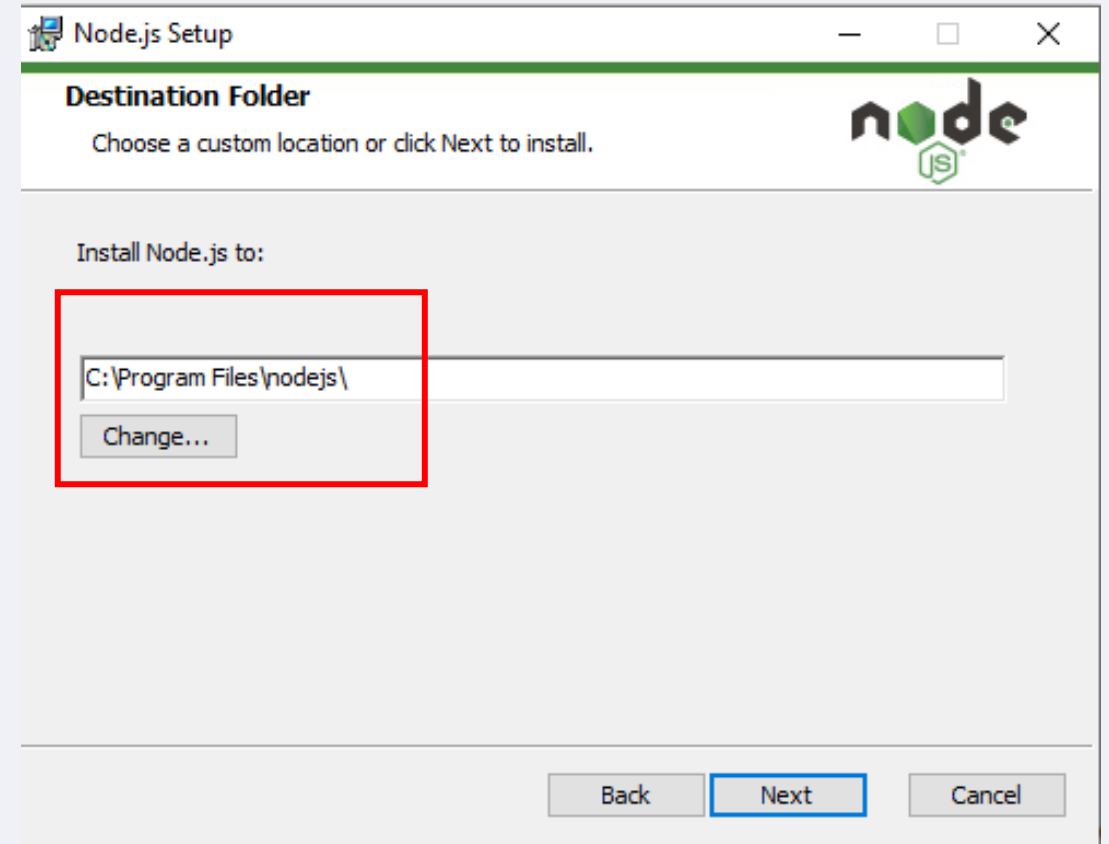
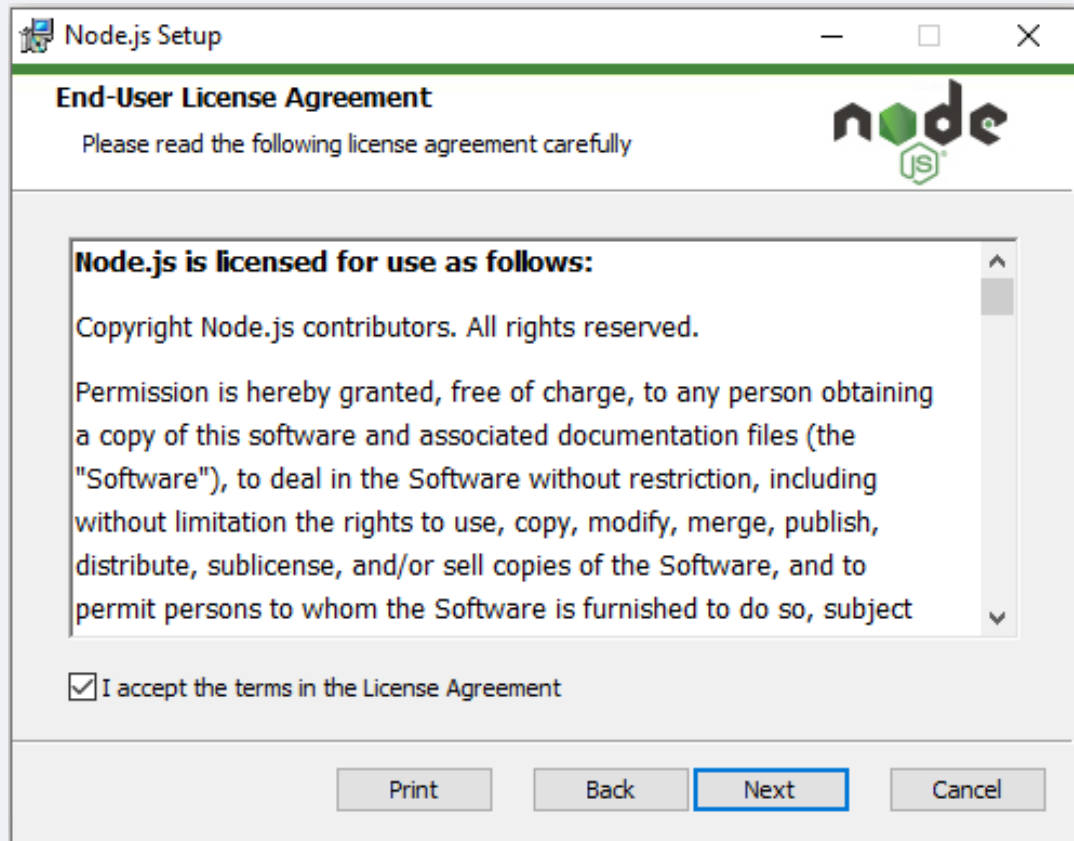
node JS

Back Next Cancel



NodeJS: Download and Install

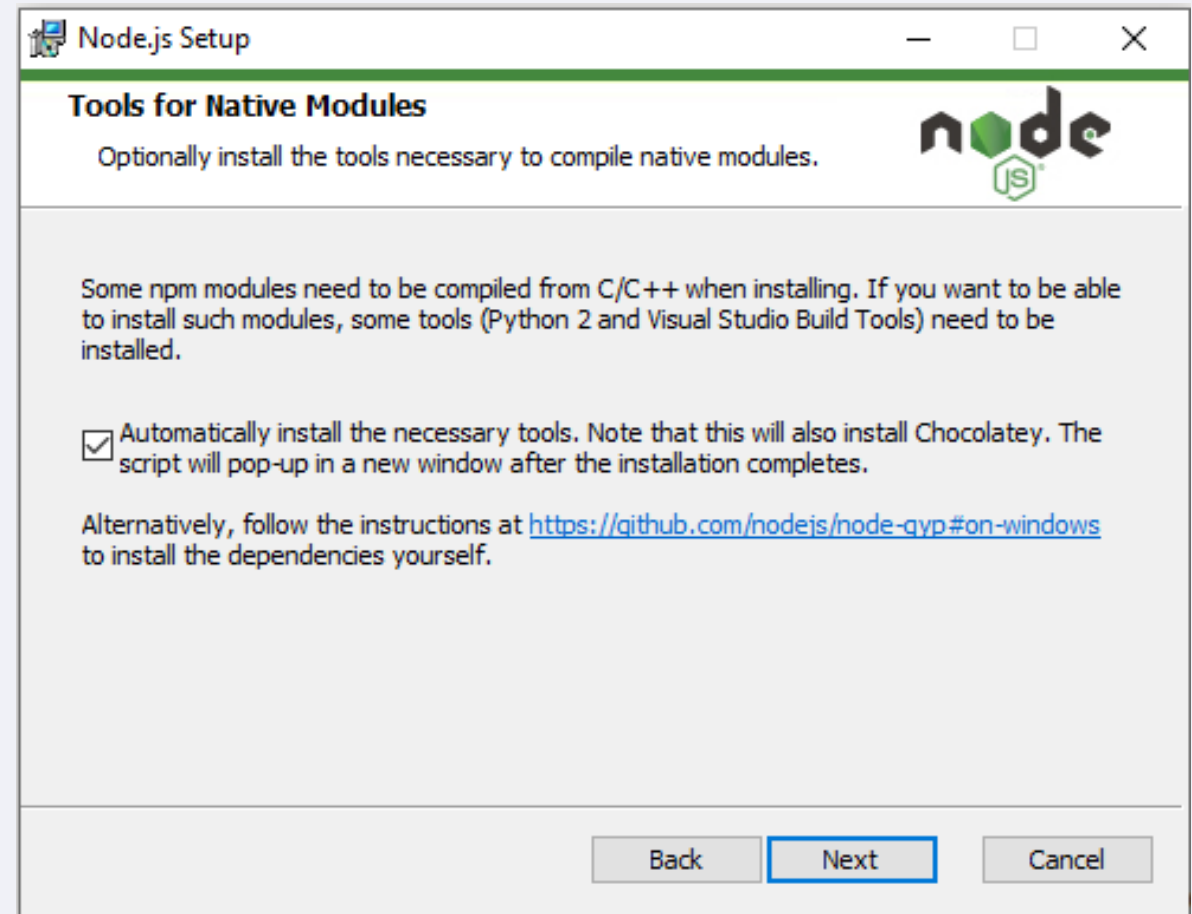
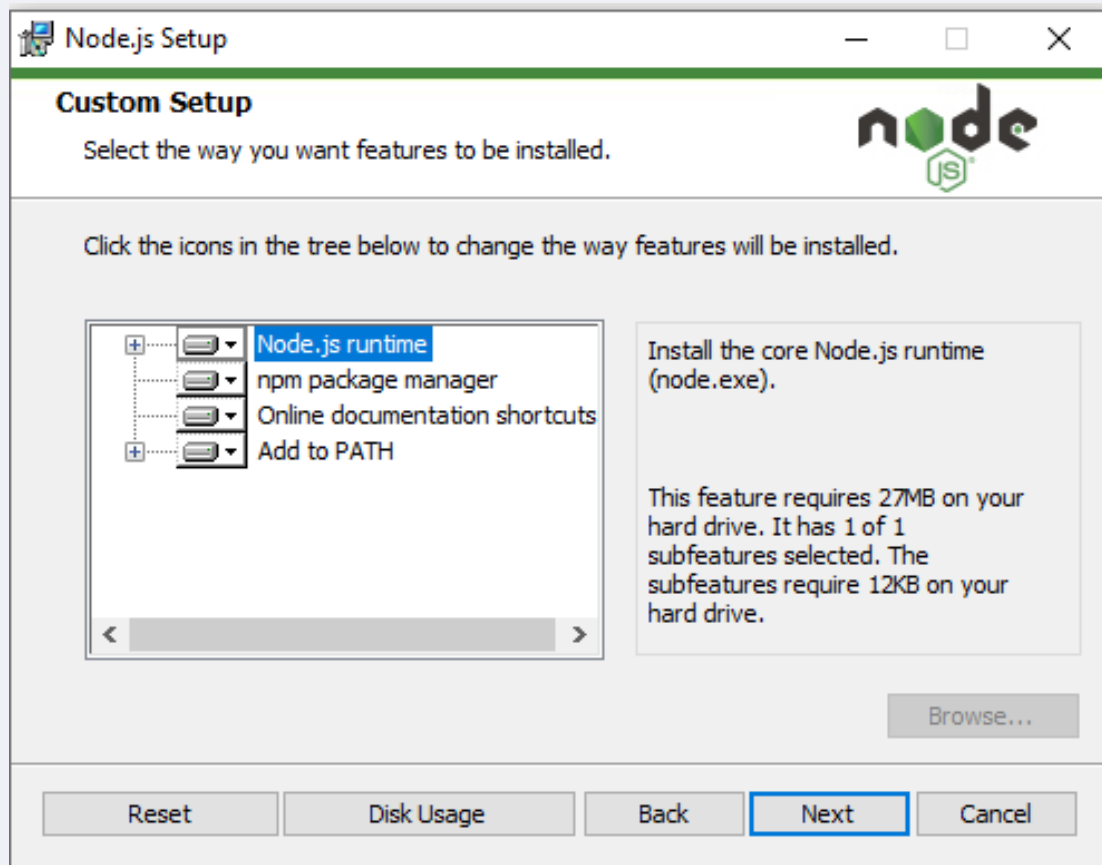
- Notice where it is installed





NodeJS: Download and Install

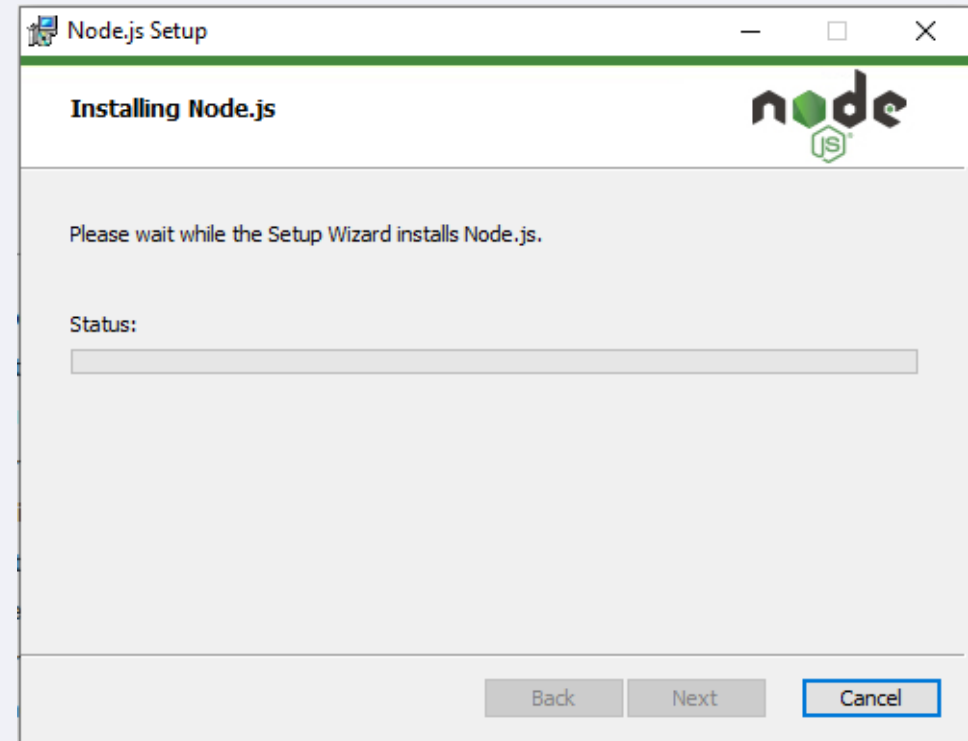
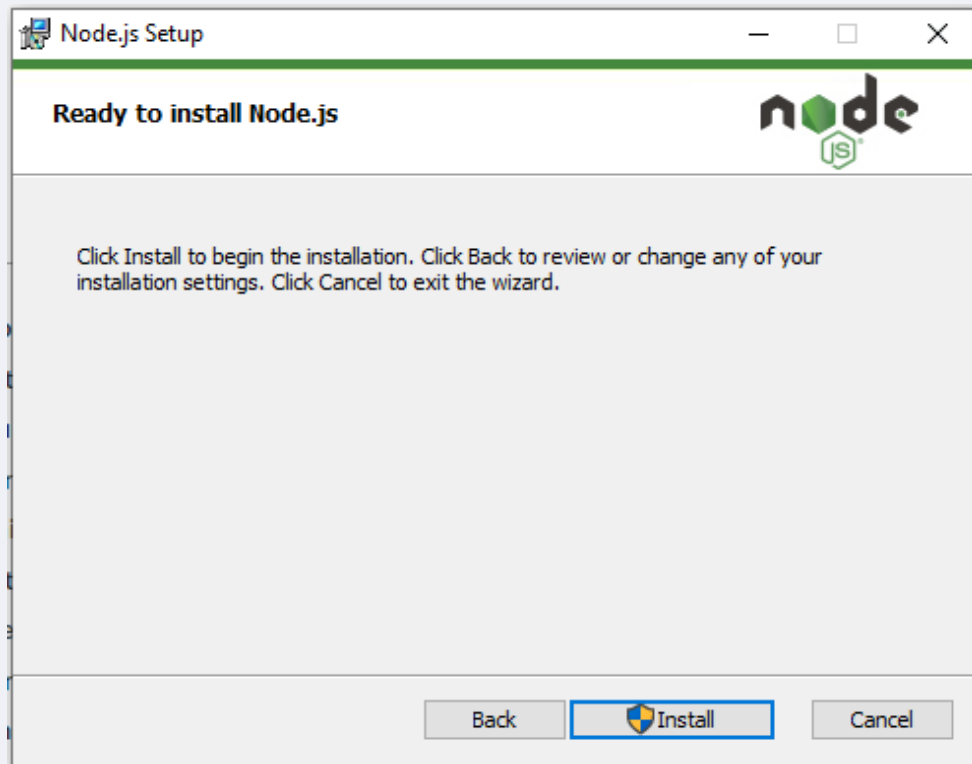
- Step through the Wizard





NodeJS: Download and Install

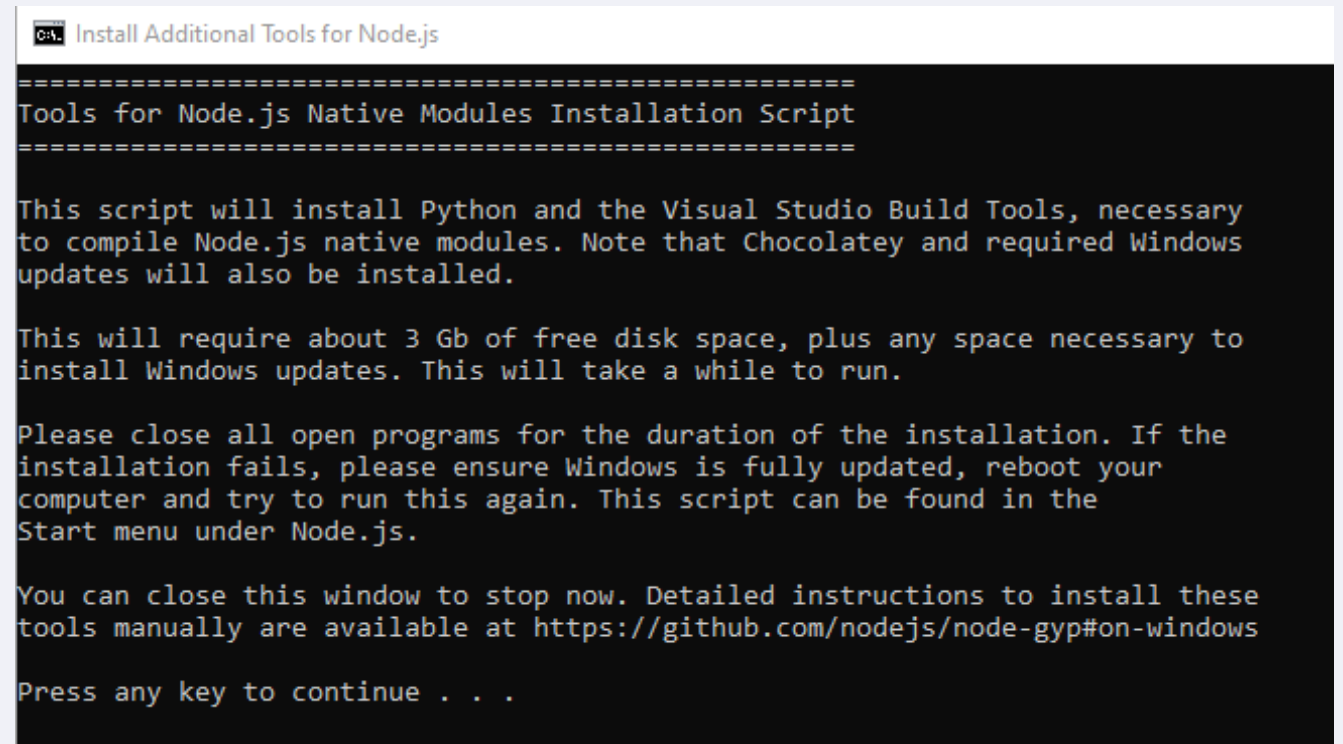
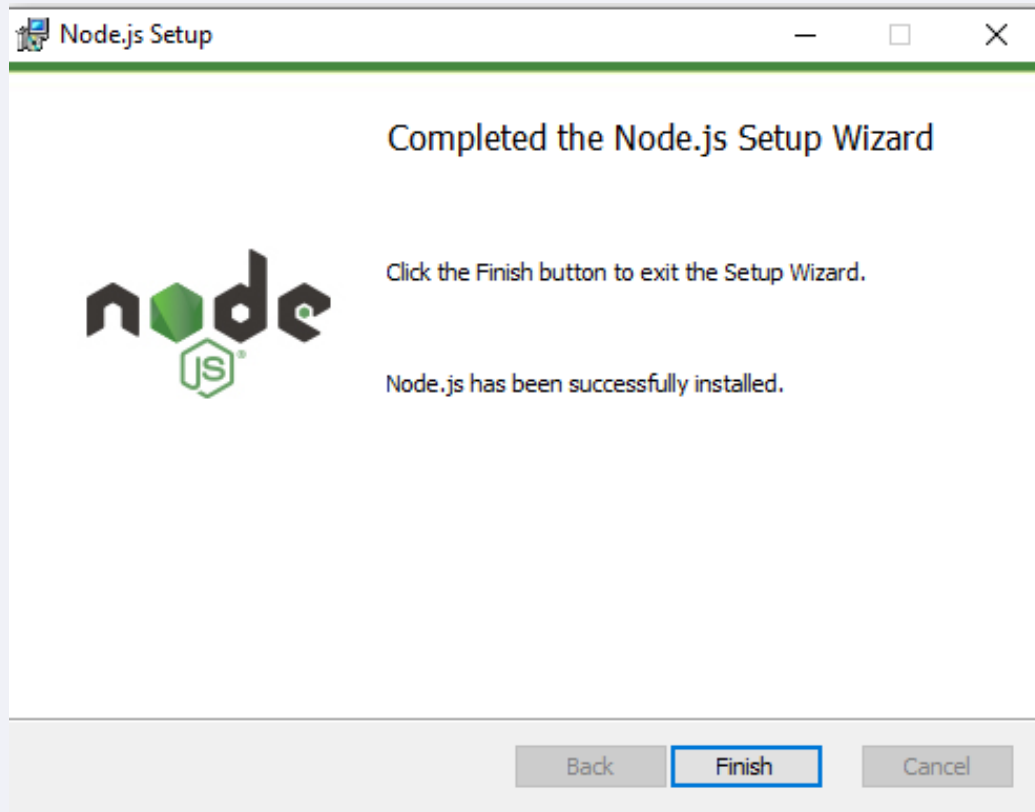
- Step through the Wizard





NodeJS: Download and Install

- Step through the Wizard





NodeJS: API Documentation

- <https://nodejs.org/dist/latest-v12.x/docs/api/>

The screenshot shows a web browser displaying the Node.js v12.13.0 API documentation. The browser's address bar shows the URL `https://nodejs.org/dist/latest-v12.x/docs/api/`. The page has a dark sidebar on the left with the Node.js logo and a list of navigation links. The main content area has a white background and features the title "Node.js v12.13.0 Documentation" at the top. Below the title are links for "Index", "View on single page", "View as JSON", "View another version", and "Edit on GitHub". A "Table of Contents" section follows, listing various API topics with bullet points.

Node.js

About these Docs
Usage & Example

Assertion Testing
Async Hooks
Buffer
C++ Addons
C/C++ Addons - N-API
Child Processes
Cluster
Command Line Options
Console
Crypto
Debugger
Deprecated APIs
DNS
Domain
ECMAScript Modules
Errors
Events
File System
Globals
HTTP

Node.js v12.13.0 Documentation

[Index](#) | [View on single page](#) | [View as JSON](#) | [View another version](#) | [Edit on GitHub](#)

Table of Contents

- [About these Docs](#)
- [Usage & Example](#)
- [Assertion Testing](#)
- [Async Hooks](#)
- [Buffer](#)
- [C++ Addons](#)
- [C/C++ Addons - N-API](#)
- [Child Processes](#)
- [Cluster](#)
- [Command Line Options](#)
- [Console](#)
- [Crypto](#)
- [Debugger](#)
- [Deprecated APIs](#)
- [DNS](#)
- [Domain](#)
- [ECMAScript Modules](#)
- [Errors](#)
- [Events](#)
- [File System](#)
- [Globals](#)
- [HTTP](#)



NodeJS: New Projects Folder

- Open a Command Terminal Window
- Windows Search for cmd
- Apple Mac search for Terminal
- Make the directory projects in your user area

```
Command Prompt
Microsoft Windows [Version 10.0.18362.418]
(c) 2019 Microsoft Corporation. All rights reserved.

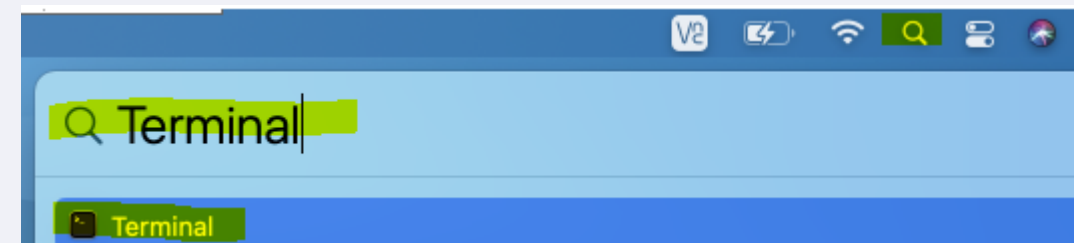
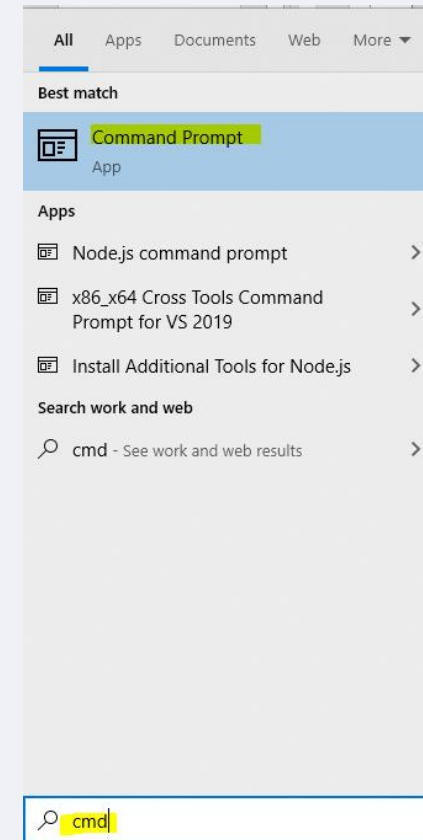
C:\Users\joeoa>mkdir projects

C:\Users\joeoa>cd projects

C:\Users\joeoa\projects>
```

```
projects — -zsh — 80x24

0:09:47 on console
~ % mkdir projects
~ % cd projects
projects % █
```





NodeJS: hello-world.js

- In the projects folder create a new file hello-world.js
- On Windows use the notepad editor
- On Apple Mac use the nano editor
- Type in the following JavaScript code

```
CA: Command Prompt
C:\Users\joeoa>cd projects
C:\Users\joeoa\projects>notepad hello-world.js
```

```
projects — -zsh — 80x24
10:09:47 on console
~ % mkdir projects
~ % cd projects
projects % nano hello-world.js
```

```
hello-world.js - Notepad
File Edit Format View Help
const http = require('http');
const hostname = '127.0.0.1';
const port = 3000;
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, Joe!\n');
});
server.listen(port, hostname, () => {
  console.log(port);
  console.log(hostname);
});
```



NodeJS: hello-world.js

- JavaScript Keyword `const` constant value cannot be changed
 - Node `require()` module **require**('/path/to/file');
 - Node hostname '127.0.0.1' is the local host which means network traffic doesn't leave the computer and it communicates with itself
 - Node port #: listening port number that Node Server will use to accept requests from clients
- Method Input arguments
 - Request received from web browser to Node **req**
 - Respond sent from Node to web browser **res**
- Parameters: http status codes – next page
- Response HTTP header `res.setHeader()` sets the MIME Multi-purpose Internet Mail Extensions – file type
- Response `end()` sends the HTML content string 'Hello World\n' \n backslash n is the newline character

```
const http = require('http');
```

```
const hostname = '127.0.0.1';
```

```
const port = 3000;
```

```
const server = http.createServer((req, res) => {  
  res.statusCode = 200;  
  res.setHeader('Content-Type', 'text/plain');  
  res.end('Hello, World!\n');  
});
```

```
server.listen(port, hostname, () => {  
  console.log(`Server running at  
http://${hostname}:${port}/`);  
});
```

NodeJS: hello-world.js



- HTTP status codes

1xx Informational

100 Continue

101 Switching Protocols

102 Processing (WebDAV)

2xx Success

★ 200 OK

203 Non-Authoritative Information

206 Partial Content

226 IM Used

★ 201 Created

★ 204 No Content

207 Multi-Status (WebDAV)

202 Accepted

205 Reset Content

208 Already Reported (WebDAV)

3xx Redirection

300 Multiple Choices

303 See Other

306 (Unused)

301 Moved Permanently

★ 304 Not Modified

307 Temporary Redirect

302 Found

305 Use Proxy

308 Permanent Redirect (experimental)

4xx Client Error

★ 400 Bad Request

★ 403 Forbidden

406 Not Acceptable

★ 409 Conflict

412 Precondition Failed

415 Unsupported Media Type

418 I'm a teapot (RFC 2324)

423 Locked (WebDAV)

426 Upgrade Required

431 Request Header Fields Too Large

450 Blocked by Windows Parental Controls (Microsoft)

★ 401 Unauthorized

★ 404 Not Found

407 Proxy Authentication Required

410 Gone

413 Request Entity Too Large

416 Requested Range Not Satisfiable

420 Enhance Your Calm (Twitter)

424 Failed Dependency (WebDAV)

428 Precondition Required

444 No Response (Nginx)

451 Unavailable For Legal Reasons

402 Payment Required

405 Method Not Allowed

408 Request Timeout

411 Length Required

414 Request-URI Too Long

417 Expectation Failed

422 Unprocessable Entity (WebDAV)

425 Reserved for WebDAV

429 Too Many Requests

449 Retry With (Microsoft)

499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error

503 Service Unavailable

506 Variant Also Negotiates (Experimental)

509 Bandwidth Limit Exceeded (Apache)

598 Network read timeout error

501 Not Implemented

504 Gateway Timeout

507 Insufficient Storage (WebDAV)

510 Not Extended

599 Network connect timeout error

502 Bad Gateway

505 HTTP Version Not Supported

508 Loop Detected (WebDAV)

511 Network Authentication Required



NodeJS: Run the Node Server

Save the file, go back to the terminal window, and enter the following command:

```
$ node hello-world.js
```

Output like this should appear in the terminal:

```
Server running at http://127.0.0.1:3000/
```

Now, open any preferred web browser and visit `http://127.0.0.1:3000`.

```
C:\Users\joeoa\projects>dir
Volume in drive C is Windows
Volume Serial Number is 6C63-187E

Directory of C:\Users\joeoa\projects

11/11/2019  09:56 AM    <DIR>          .
11/11/2019  09:56 AM    <DIR>          ..
11/11/2019  09:56 AM                354 hello-world.js
                1 File(s)                354 bytes
                2 Dir(s)  906,716,893,184 bytes free

C:\Users\joeoa\projects>node hello-world.js
Server running at http://127.0.0.1:3000/
```

```
← → ↻ ⓘ 127.0.0.1:3000
```

Hello, World!



PennState

College of Information
Sciences and Technology

IST 256

MEAN Stack



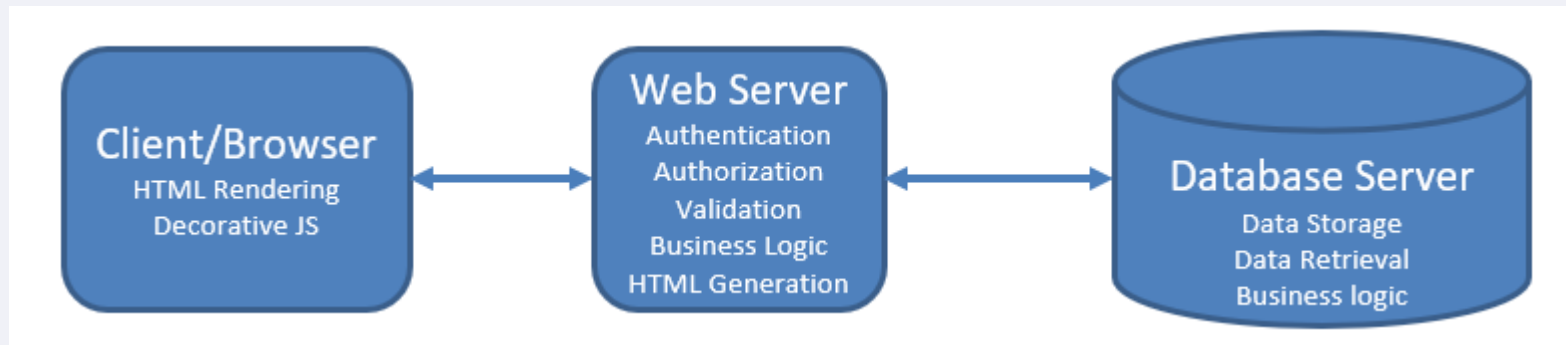
MEAN Stack: Introduction

- MEAN Stack
 - **MongoDB**: a NoSQL database
 - **Express.js**: a web application framework that runs on Node.js
 - **AngularJS**: a JavaScript MVC framework that runs in a browser JavaScript engines
 - **Node.js**: an execution environment for event-driven server-side and networking applications



MEAN Stack: Technologies

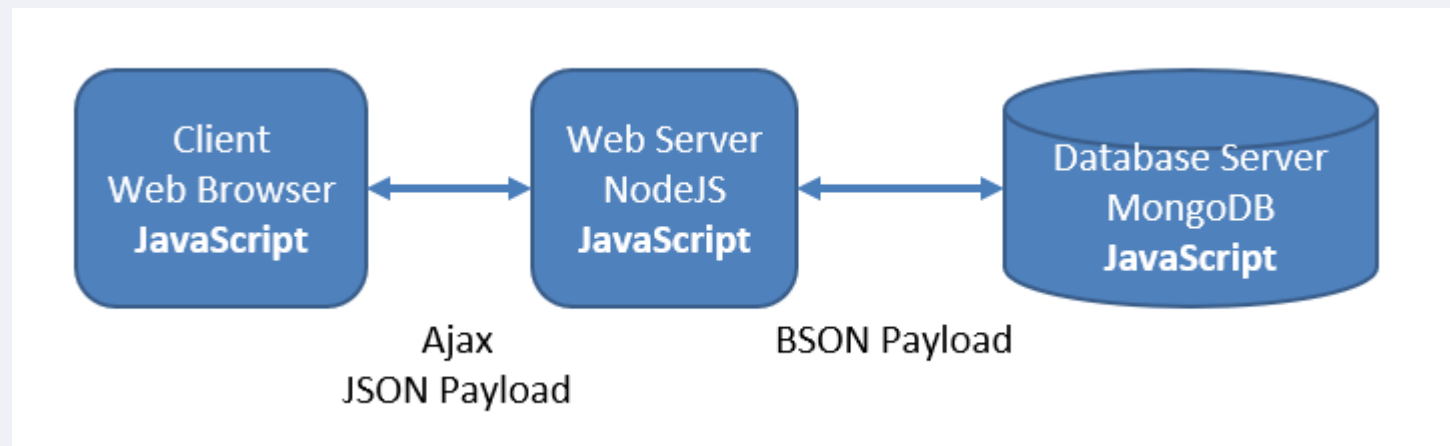
- Technologies
 - **Client-Side Browser:** HTML5, CSS3, SVG, AJAX, DOM, JavaScript, Knockout, AngularJS, jQuery, JSON
 - **Web Server Side:** Node.js, Express, Pug, Mongoose, Passport
 - **Backend Database:** BSON, MongoDB, RoboMongo





MEAN Stack: Technologies

- JavaScript End-to-End – Front end Browser, Middle ware, Backend Database
- JavaScript can be the language to work with the Database MongoDB
- JavaScript Web Server – for example NodeJS
- JavaScript works well within the Web browser
- Deployment of JavaScript is easy – text files libraries
- JavaScript contains advanced features
- JavaScript can be fast
- JavaScript is mature
- JavaScript is useful for cross-platform development





MEAN Stack: JSON

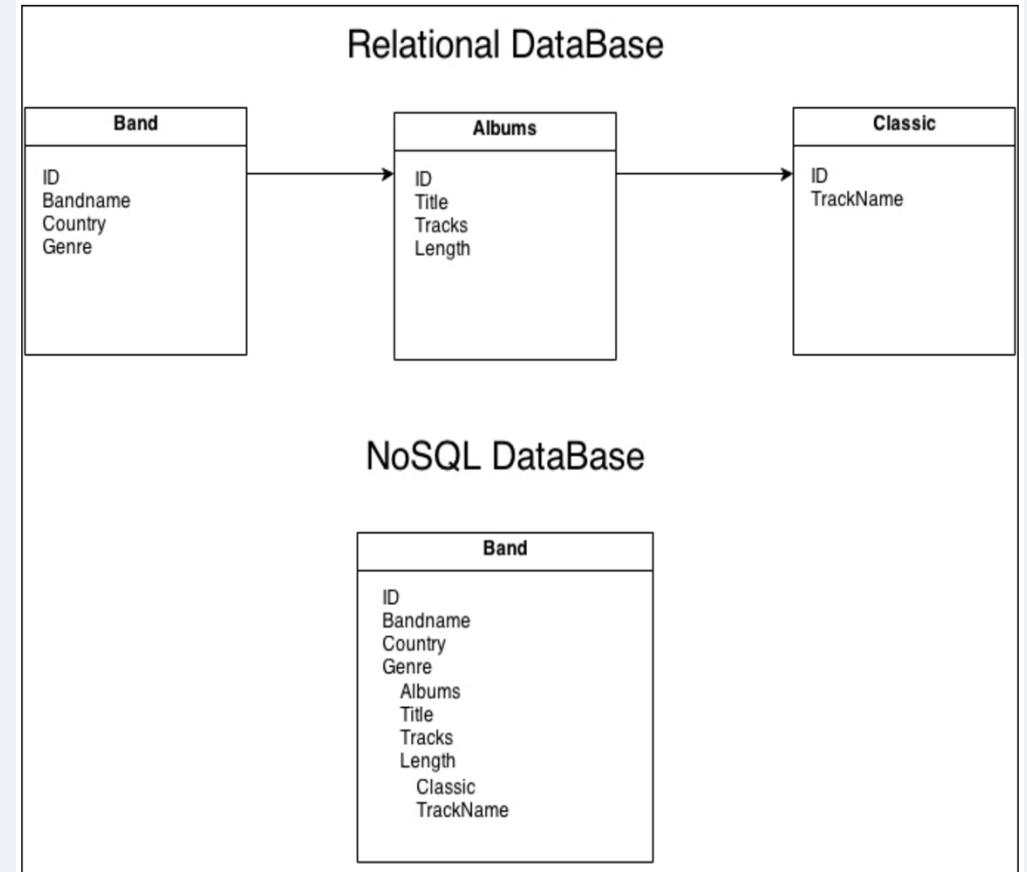
- **JSON: JavaScript Object Notation** is an open, human and machine-readable standard that facilitates data interchange
- Supports types: numbers, strings, and Boolean values, as well as arrays and hashes.

```
{  
  "object": { "a": "b", "c": "d", "e": "f" },  
  "array": [ 1, 2 ], "string": "Hello World"  
}
```



MEAN Stack: MongoDB

- Agile, scalable, document-oriented, schema less, and high performance
- All of the data is stored like a JSON file
- Stores this data in the key value format using Binary JSON BSON





MEAN Stack: MongoDB

- Binary JSON (BSON)
- **MongoDB** represents JSON documents in binary-encoded format called BSON behind the scenes.
- Document database such as MongoDB use JSON documents in order to store records, just as tables and rows store records in a relational database.
- **BSON** extends the JSON model to provide additional data types and to be efficient for encoding and decoding within different languages



MEAN Stack: MongoDB

- The document is self-contained with all the information in the same place
- Multiple joins takes time
- MongoDB executes a single query
- Document-based NoSQL with transactions and joins
- No schemes need to be predefined to insert the data

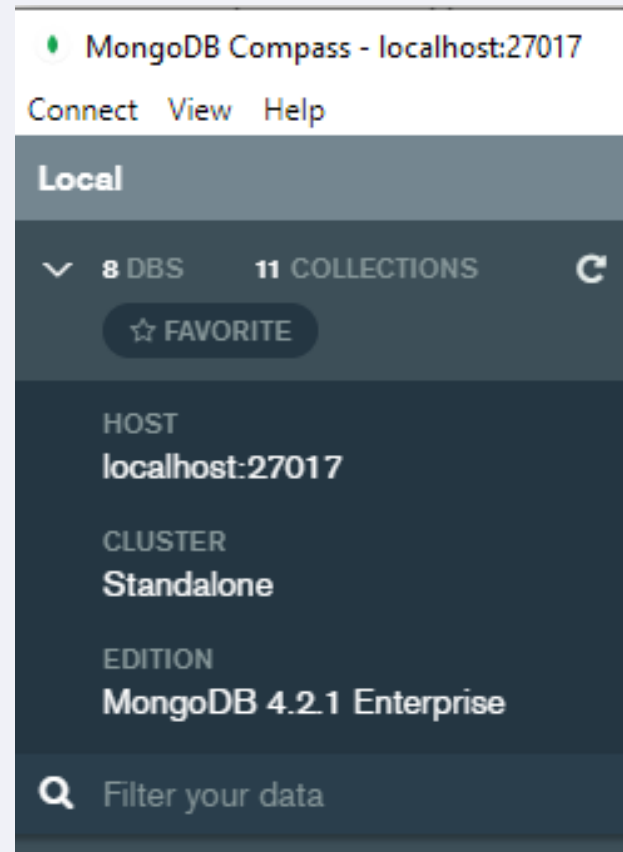
MongoDB	SQL
<code>db.name.insert({ A:1, B:2 })</code>	<code>INSERT INTO NAME VALUES(A, B)</code>
<code>db.users.find()</code>	<code>SELECT * FROM users</code>
<code>db.bands.find({ name: "Metallica" })</code>	<code>SELECT * FROM bands WHERE name = "Metallica"</code>
<code>db.bands.find().limit(5).skip(10)</code>	<code>SELECT * FROM bands LIMIT 5 SKIP 10</code>



MEAN Stack: MongoDB

- Mongo Interactive Shell or Compass GUI Interface

```
tiger@ubuntu: ~  
tiger@ubuntu:~$ mongo  
MongoDB shell version: 3.2.9  
connecting to: test  
Server has startup warnings:  
2016-09-12T17:41:50.002-0400  
2016-09-12T17:41:50.002-0400  
mm/transparent_hugepage/enabl  
2016-09-12T17:41:50.002-0400  
ting it to 'never'  
2016-09-12T17:41:50.002-0400  
2016-09-12T17:41:50.002-0400  
mm/transparent_hugepage/defra  
2016-09-12T17:41:50.002-0400  
ting it to 'never'  
2016-09-12T17:41:50.002-0400  
>
```





MEAN Stack: MongoDB

- Terminology differences between MongoDB and SQL

MongoDB terminology	SQL terminology
database	database
collection	table
document/BSON document	row
field	column
index	index
embedded document/linking	table joins
primary key is <code>_id</code> field	column or column combination as primary key
aggregation pipeline	aggregation: group by



MEAN Stack: MongoDB

- MongoDB commands use, insert, find
- use `exampledb`
- `db.bands.insert({name: "Devo", album: "freedom of choice", tracks: 9, year: "1985"})`
- `db.bands.find()`

```
> db.bands.insert({name: "Devo", album: "freedom of choice", tracks: 9, year: "1985"})
WriteResult({ "nInserted" : 1 })
> db.bands.find()
{ "_id" : ObjectId("57d73c2c2d4dcff383228553"), "name" : "Devo", "album" : "freedom of choice", "tracks" : 9, "year" : "1985" }
> █
```



MEAN Stack: Express

- Flexible framework to build web applications on Node
- RESTful API: supports main HTTP methods: GET, POST, PUT, DELETE
- Supports templates
- PUG rendering engine - HTML server-side rendering
- Powerful routing management
- Routing is a way of organizing and managing application states.
- A routing framework in JavaScript helps you to change the state of the application--perhaps moving from one admin panel section to another--while maintaining *application persistence*.



MEAN Stack: AngularJS

- Open Source JavaScript framework by Google
- <http://angularjs.org/>
- Allows you to declare dynamic views in web-applications
- Extend the HTML vocabulary for the web application
- Angular modules solve the problem of removing global state from the application and uses an injector
- CDN Content Delivery Network
- <https://ajax.googleapis.com/ajax/libs/angularjs/1.2.14/angular.min.js>

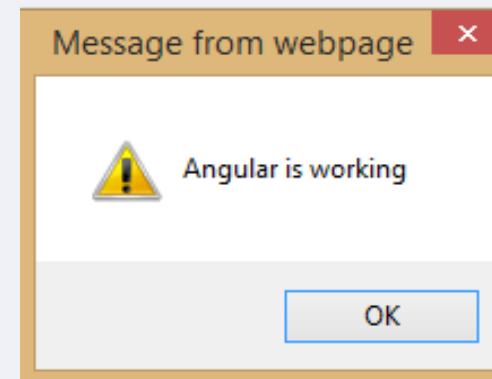
```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head><meta charset="utf-8" /><title></title>
  <script
    type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.14/angular.min.js">
  </script>
</head>
```



MEAN Stack: AngularJS

- To tell AngularJS to start paying attention to an area of your HTML DOM include an attribute **data-ng-app** this attribute is called a directive
- All AngularJS directives start either with **ng-** or **data-ng** name-with-dashes
- Controller is a unit of code that be executed when AngularJS detects the controller directive: **data-ng-controller**
- The name defined in the controller directive refers to a named object usually a function

```
control.html - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head><meta charset="utf-8" /><title></title>
  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.14/angular.min.js"
  </script>
</head>
<body>
<div data-ng-app>
<div data-ng-controller="controller">
  <script>
    // $scope is an object
    function controller($scope) {
      alert("Angular is working");
    }
  </script>
</div>
</body></html>
```





MEAN Stack: AngularJS

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head><meta charset="utf-8" /><title></title>
  <script
    type="text/javascript"
    src="https://ajax.googleapis.com/ajax/libs/angularjs/1.2.14/angular.min.js">
  </script>
</head>
<body>
<div data-ng-app>
<div data-ng-controller="Controller">
  My name is {{ name }} and my favorite sport is {{ sport }}.
</div>
  <script>
    // $scope is an object
    function Controller($scope) {
      $scope.name = "Joe Smith";
      $scope.sport = "MTB";
    }
  </script>
</body></html>
```

← → ↻ ⓘ File | C:/Users/joeoa/projects/sportcontrol.html

My name is Joe Smith and my favorite sport is MTB.



MEAN Stack: NodeJS

- An event-driven I/O server-side JavaScript environment
- Uses a non-blocking I/O model that makes it lightweight and efficient.
- Designed to build scalable network applications.
- Node.js's package ecosystem, npm, is currently the largest ecosystem of open-source libraries.
- Web Server-side runs JavaScript applications
- Platform layer: interact with the OS: write and read files, networking operations, spawn child processes
- Single Threaded event loop
- Built-in asynchronous I/O



MEAN Stack: NodeJS NPM

- NPM: Node Package Manager
- command -line utility that interacts with an online open-source repository for projects – 76,000 packages <http://www.npmjs.org>
- Install, manage dependencies, versions - through a Package.json
- npm init - setup for package.json file
- npm help
- npm update
- npm install modeule_name
- npm install modeule_name-save
- npm list - list all the modules in the project
- npm list-g - global modules
- npm remove modeule_name
- module_name npm update
- npm -v - displays the current version
- npm adduser username
- npm whoami - public profile on the NPM repository
- npm publish - publishes your module on the NPM repository



MEAN Stack: Mongoose

- Object modeling package for Node
- Allows us to have access to the MongoDB commands for CRUD
- Ability to create and validate schemas for objects in a database

```
1 const mongoose = require('mongoose');
2 const statuses = ['new', 'in progress', 'complete'];
3 const priorities = ['high', 'medium', 'low'];
4
5 const todoSchema = new mongoose.Schema({
6   name: { type: String, required: true, unique: true },
7   description: String,
8   dueDate: Date,
9   status: {type: String, enum: statuses},
10  priority: {type: String, enum: priorities},
11  assignedTo: String,
12  createdBy: String
13
14 }, { timestamps: true });
15
16
17 const ToDo = mongoose.model('ToDo', todoSchema);
18
19 module.exports = ToDo;
```



MEAN Stack: Pug

- Is a clean, whitespace-sensitive template language for writing HTML
- Jade has been renamed to Pug
- Jade/Pug is a templating engine for nodejs
- A template engine enables you to use static template files in your application.
- At runtime, the template engine replaces variables in a template file with actual values, and transforms the template into an HTML file sent to the client.
- Some popular template engines are Pug, Mustache, Dust, EJS
- Separate your HTML from dynamic content
- Generating HTML with a preprocessor allows for more readable code & easier to maintain



MEAN Stack: Pug

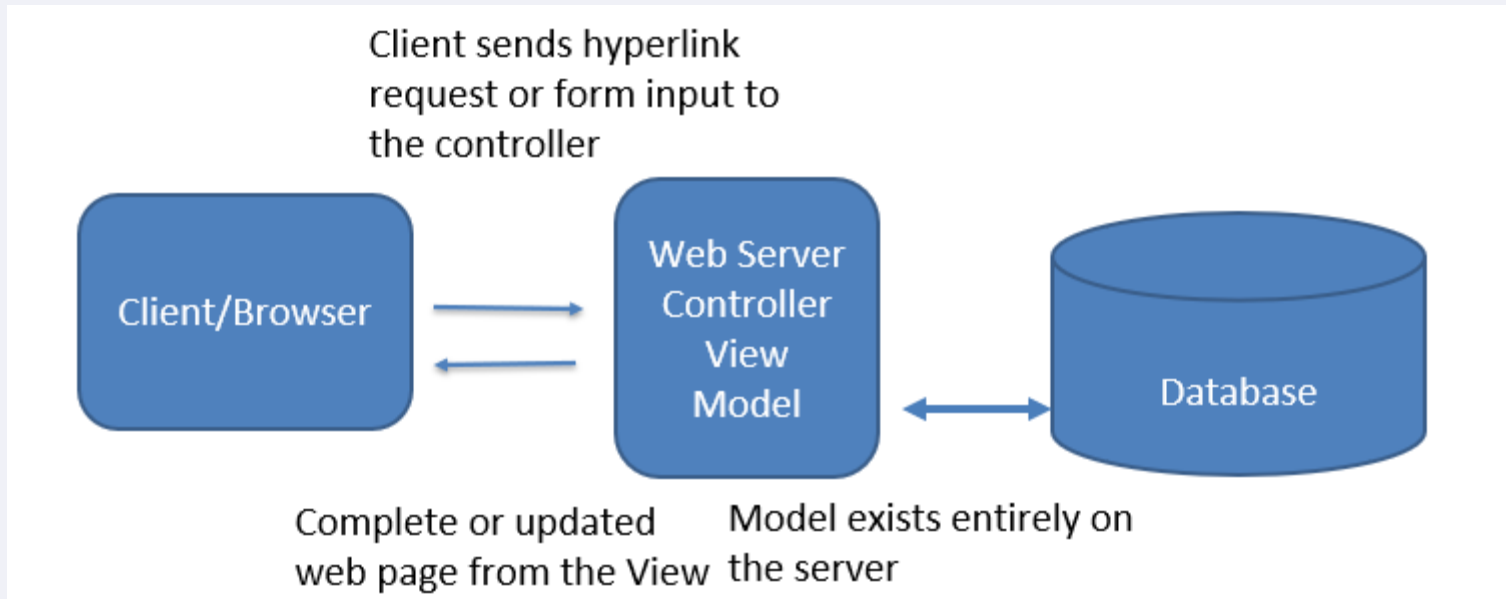
```
doctype html
html(lang="en")
  head
    title= pageTitle
    script(type='text/javascript').
      if (foo) bar(1 + 5)
  body
    h1 Jade - node template engine
    #container.col
      if youAreUsingJade
        p You are amazing
      else
        p Get on it!
      p.
        Jade is a terse and simple templating language with a
        strong focus on performance and powerful features.
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Jade</title>
    <script type="text/javascript">
      if (foo) bar(1 + 5)
    </script>
  </head>
  <body>
    <h1>Jade - node template engine</h1>
    <div id="container" class="col">
      <p>You are amazing</p>
      <p>Jade is a terse and simple templating language with a strong focus on performance and powerful features.</p>
    </div>
  </body>
</html>
```



MEAN Stack: MVC

- **Model View Controller (MVC):** software architecture pattern separates the visual representation of information from the user's interaction
- Thin Client approach: ASP.NET MVC, Ruby on Rails, Django, Express





MEAN Stack: MVC

- A ***model*** stores data that is retrieved according to commands from the controller and displayed in the view.
- A ***view*** generates new output to the user based on changes in the model.
- A ***controller*** can send commands to the model to update the model's state (e.g. editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g. by scrolling through a document).



MEAN Stack: MVC

- The ***model*** directly manages the data, logic and rules of the application. The model is the application object.
- A ***view*** can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart and a tabular view. The view is the screen presentation.
- The ***controller*** accepts input and converts it to commands for the model or view. The controller defines the way the user interface reacts to user input.

NoSQL-Datenbanken

4. MongoDB

Prof. Dr. Jürgen Heym

Hochschule Hof

4 – MongoDB – DDL / DML

- DDL == Data Definition Language
 - Databases
 - Create database `use DATABASE_NAME`
 - Current database `db`
 - List databases `show dbs`
 - Databases are created as soon as you insert the first document.
 - Delete a database `db.dropDatabase()`

4 – MongoDB – DDL / DML

- DDL == Data Definition Language

- Collections

- Create a collection `db.createCollection(NAME, OPTIONS)`
- Example:

```
db.createCollection("mycol",
                    { capped : true, size : 6142800, max : 10000 })
```
- In mongodb you don't need to create collection.
MongoDB creates collections automatically, when you insert the first document.
- Drop a collection `db.COLLECTION_NAME.drop()`
- The drop() method will return true, if the selected collection is dropped successfully otherwise it will return false.

Parameter	Type	Description
Name	String	Name of the collection to be created
Options	Document	(Optional) Specify options about memory size and indexing

4 – MongoDB – DDL / DML

- DDL == Data Definition Language
 - Collection Options

Field	Type	Description
capped	Boolean	(Optional) If true, enables a capped collection. A capped collection is a size limited collection that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify the size parameter also.
autoIndexID	Boolean	(Optional) If true, automatically create index on _id fields. Default is false.
size	number	(Optional) Specifies a maximum size in bytes for a capped collection.
max	number	(Optional) Specifies the maximum number of documents allowed in the capped collection.

4 – MongoDB – DDL / DML

- DDL == Data Definition Language

- Indexes

- Indexes support the efficient resolution of queries. Without indexes, MongoDB must scan every document of a collection to select those documents that match the query statement. This scan is highly inefficient and require the mongod to process eventually a large volume of data.
- Indexes are special data structures, that store a small portion of the data set in an easy to traverse form. The index stores the value of a specific field or set of fields, ordered by the value of the field as specified in index.
- To create an index you need to use `ensureIndex()` method of `mongodb`.
- Ascending index: 1, descending index: -1
- Syntax

```
db.col.ensureIndex({KEY:1})
```

- Example

```
db.col.ensureIndex({"title":1,"description":-1})
```

4 – MongoDB – DDL / DML

- DDL == Data Definition Language

- Index Options (1)

Parameter	Type	Description
background	boolean	Builds the index in the background so that building an index does not block other database activities. Specify true to build in the background. The default value is false .
unique	boolean	Creates a unique index so that the collection will not accept insertion of documents where the index key or keys match an existing value in the index. Specify true to create a unique index. The default value is false .
name	string	The name of the index. If unspecified, MongoDB generates an index name by concatenating the names of the indexed fields and the sort order.
dropDups	boolean	Creates a unique index on a field that may have duplicates. MongoDB indexes only the first occurrence of a key and removes all documents from the collection that contain subsequent occurrences of that key. Specify true to create unique index. The default value is false .
sparse	boolean	If true, the index only references documents with the specified field. These indexes use less space but behave differently in some situations (particularly sorts). The default value is false .

4 – MongoDB – DDL / DML

- DDL == Data Definition Language
 - Index Options (2)

Parameter	Type	Description
expireAfterSeconds	integer	Specifies a value, in seconds, as a TTL to control how long MongoDB retains documents in this collection.
v	index version	The index version number. The default index version depends on the version of mongod running when creating the index.
weights	document	The weight is a number ranging from 1 to 99,999 and denotes the significance of the field relative to the other indexed fields in terms of the score.
default_language	string	For a text index, the language that determines the list of stop words and the rules for the stemmer and tokenizer. The default value is english .
language_override	string	For a text index, specify the name of the field in the document that contains, the language to override the default language. The default value is language.

4 – MongoDB – DDL / DML

- MongoDB Data Types

Datatype	Description
String	This is most commonly used data type to store the data. String in mongodb must be UTF-8 valid.
Integer	This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
Boolean	This type is used to store a boolean (true/ false) value.
Double	This type is used to store floating point values.
Min / Max keys	This type is used to compare a value against the lowest and highest BSON elements.
Arrays	This type is used to store arrays or list or multiple values into one key.
Timestamp	Ctimestamp. This can be handy for recording when a document has been modified or added.
Object	This data type is used for embedded documents.
Null	This type is used to store a Null value.

4 – MongoDB – DDL / DML

- MongoDB Data Types

Datatype	Description
Date	This data type is used to store the current date or time in UNIX time format. You can specify your own date time by creating an object of Date and passing day, month, year into it.
Object ID	This data type is used to store the document's ID.
Binary data	This data type is used to store binary data.
Code	This data type is used to store JavaScript code into a document.
Regular expression	This data type is used to store regular expressions.

4 – MongoDB – DDL / DML

- DML == Data Manipulation Language
 - Insert Data
 - To insert data into a MongoDB collection, you need MongoDB's **insert()** or **save()** method:


```
db.COLLECTION_NAME.insert(document)  
db.COLLECTION_NAME.save(document)
```
 - If the collection doesn't exist in the database, MongoDB will create this collection and then insert the document the collection.
 - If we don't specify the `_id` parameter, then MongoDB assigns an unique ObjectId for this document.
 - The ObjectId (`_id`) is a 12 bytes hexadecimal number, unique for every document in a collection: 4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer.
 - If you specify the `_id`-Parameter and you use the method **save()**, then the document will be replaced.

4 – MongoDB – DDL / DML

- DML == Data Manipulation Language

- Replace a document

- Example

```
db.books.save({
  _id: ObjectId(7df78ad8902c),
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100
})
```

4 – MongoDB – DDL / DML

- DML == Data Manipulation Language
 - Insert multiple documents
 - Example

```
db.books.insert([
  {
    title: 'MongoDB Overview',
    description: 'MongoDB is no sql database',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 100
  }, {
    title: 'NoSQL Database',
    description: 'NoSQL databases don't have tables',
    tags: ['mongodb', 'database', 'NoSQL'],
    likes: 20
  }
])
```

4 – MongoDB – DDL / DML

- Querying the database

- Basic syntax of method **find()**

```
db.COLLECTION_NAME.find()
```

- Basic syntax for formatting the output with method **pretty()**

```
db.COLLECTION_NAME.find().pretty()
```

4 – MongoDB – DDL / DML

- Querying the database
 - Comparison Operators

Operation	Syntax	Example	RDBMS Equivalent
Equality	{<key>:<value>}	<code>db.col.find({"by":"tutorials"})</code>	where by = 'tutorials'
Less Than	{<key>:{\$lt:<value>}}	<code>db.col.find({"likes":{\$lt:50}})</code>	where likes < 50
Less Than Equals	{<key>:{\$lte:<value>}}	<code>db.col.find({"likes":{\$lte:50}})</code>	where likes <= 50
Greater Than	{<key>:{\$gt:<value>}}	<code>db.col.find({"likes":{\$gt:50}})</code>	where likes > 50
Greater Than Equals	{<key>:{\$gte:<value>}}	<code>db.col.find({"likes":{\$gte:50}})</code>	where likes >= 50
Not Equals	{<key>:{\$ne:<value>}}	<code>db.col.find({"likes":{\$ne:50}})</code>	where likes != 50

4 – MongoDB – DDL / DML

■ Querying the database

- Logical AND

- If you pass in the **find()** method multiple keys by separating them by ',' then MongoDB treats it as an **AND** condition.
- Example

```
db.col.find({key1:value1, key2:value2})
```

- Logical OR

- Syntax:

```
db.col.find( { $or: [ {key1: value1}, {key2:value2} ] } )
```

4 – MongoDB – DDL / DML

■ Übung

1. Erzeugen Sie eine MongoDB-Datenbank: `library_<<xy>>`, wobei `xy` ihre Initialen sind. Beachten Sie eventuelle Initialenüberschneidungen.
2. Erzeugen Sie eine Kollektion `books` in ihrer Datenbank.
3. Fügen Sie drei Dokumente in ihre Kollektion ein.
4. Üben Sie das Auffinden von Dokumenten mit der `find()`-Methode. Entwickeln Sie hierzu entsprechende Fragestellungen (Anwendungsfälle), die alle gültigen Vergleichsoperatoren und logischen Verknüpfungsoperatoren in geeigneter Weise nutzen.

4 – MongoDB – DDL / DML

■ Updating Documents

- MongoDB's **update()** and **save()** methods are used to update documents in a collection.

The `update()` method updates values in the existing document while the `save()` method replaces the existing document with the document passed in the `save()` method.

- Syntax:

```
db.col.update(SELECTION_CRITERIA, UPDATED_DATA)
```

- By default mongodb will update only a single document, to update multiple documents you need to set a parameter 'multi' to true.

Syntax:

```
db.col.update(SELECTION_CRITERIA, UPDATED_DATA, {multi:true})
```

4 – MongoDB – DDL / DML

- Updating Documents

- Example

```
db.col.find()
```

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}  
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

```
db.col.update({'title':'MongoDB Overview'},{$set: {'title':'New MongoDB  
Tutorial'}})
```

```
db.col.find()
```

```
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}  
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```


4 – MongoDB – DDL / DML

■ Replacing Documents

- The **save()** method replaces the existing document with the new document passed in the save() method.

Basic syntax of mongodb **save()** method is shown below:

- Syntax:

```
db.col.save({_id:ObjectId(),NEW_DATA})
```

4 – MongoDB – DDL / DML

- Replacing Documents

- Example

```
db.col.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}

db.col.save(
{ "_id" : ObjectId(5983548781331adf45ec7),
  "title":"Tutorials Point New Topic",
  "by":"Tutorials Point"
}

db.col.find()
{ "_id" : ObjectId(5983548781331adf45ec5), "title":"New MongoDB Tutorial"}
{ "_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}
{ "_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic"}
```

4 – MongoDB – DDL / DML

■ Deleting Documents

- MongoDB's **remove()** method is used to remove documents from a collection. The `remove()` method accepts two parameters. One is the deletion criteria and the second is the optional `justOne`-flag.

- Syntax:

`db.col.remove(DELETION_CRITERIA)` (delete multiple docs)

or

`db.col.remove(DELETION_CRITERIA, 1)` (delete one doc)

or

`db.col.remove({})` (delete all docs)

4 – MongoDB – DDL / DML

- Querying the database (continued)

- Projections

Basic Syntax: show key1, key2, ... but don't show the ObjectId.

```
db.col.find({SELECTION_CRITERIA},{KEY1:1,...,_id:0})
```

Example

```
{"_id" : ObjectId(5983548781331adf45ec5), "title":"MongoDB Overview"}  
{"_id" : ObjectId(5983548781331adf45ec6), "title":"NoSQL Overview"}  
{"_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point Overview"}
```

```
db.col.find({},{"title":1,_id:0})
```

```
{"title":"New MongoDB Tutorial"}  
{"title":"NoSQL Overview"}  
{"title":"Tutorials Point Point Overview "}
```

4 – MongoDB – DDL / DML

- Querying the database (continued)

- Limitations

Use the `limit()` method to limit the number of shown documents.

```
db.col.find().limit(NUMBER)
```

- Skipping documents

Use the `skip()` method to skip a certain number of documents in the result set.

```
db.col.find().limit(NUMBER).skip(NUMBER2)
```

4 – MongoDB – DDL / DML

- Querying the database (continued)

- Sorting the result set

To sort documents in MongoDB, you need to use **sort()** method.

The **sort()** method accepts a document containing a list of fields along with their sorting order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

```
db.col.find({}, {"title":1, _id:0}).sort({"title":-1})
```

4 – MongoDB – DDL / DML

■ Übung (Teil 2)

1. Ändern Sie einzelne Attribute verschiedener Dokumente.
2. Ändern Sie ein Attribut mehrerer Dokumente (multi=true).
3. Ersetzen Sie ein Dokument durch ein neues Dokument.
4. Verifizieren Sie die Wirkungsweise der verschiedenen Möglichkeiten zum Löschen von Dokumenten.
5. Üben Sie die Projektion verschiedener Attribute.
6. Limitieren Sie die Ausgabe mit der Methode limit().
7. Verifizieren Sie die Funktionsweise der skip()-Methode.
8. Ändern Sie die Sortierreihenfolge eines beliebigen Suchresultats.
9. Konfigurieren Sie explizit einen bzw. mehrere Indizes mit und ohne Indexoptionen.

Literatur

- Sieben Wochen, sieben Datenbanken
Moderne Datenbanken und die NoSQL-Bewegung
E. Redmon & J. R. Wilson, O'Reilly®
ISBN 978-3-86899-791-0
- MongoDB Inc.
<https://www.mongodb.com/>
- MongoDB ORG
<https://www.mongodb.org/>
- MongoDB Tutorial
<http://www.tutorialspoint.com/mongodb/index.htm>
- MongoDB Konfigurationsoptionen
<http://docs.mongodb.org/manual/reference/configuration-options/>



PennState

College of Information
Sciences and Technology

IST MongoDB



MongoDB: Introduction

- JSON-style documents with dynamic schemas offer simplicity and power.
- MongoDB is an open-source document database - developed and supported by 10gen
- Runs on Linux, Windows, Apple
- From the NoSQL family of database systems
- Stores structured data as JSON-like documents with dynamic schemas BSON
- <http://www.mongodb.org/>

MongoDB: Download



- <https://www.mongodb.com/try/download/community>

MongoDB Community Server

MongoDB offers both an Enterprise and Community version of its powerful distributed document database. The community version offers the flexible document model along with ad hoc queries, indexing, and real time aggregation to provide powerful ways to access and analyze your data. As a distributed system you get high availability through built-in replication and failover along with horizontal scalability with native sharding.

The MongoDB Enterprise Server gives you all of this and more. Review the Enterprise Server tab to learn what else is available.

Available Downloads ^

Version	4.4.6 (current) ✓
Platform	Windows ✓
Package	msi ✓

 **Download**

Copy Link

[Current releases & packages](#)

[Development releases](#)

[Archived releases](#)

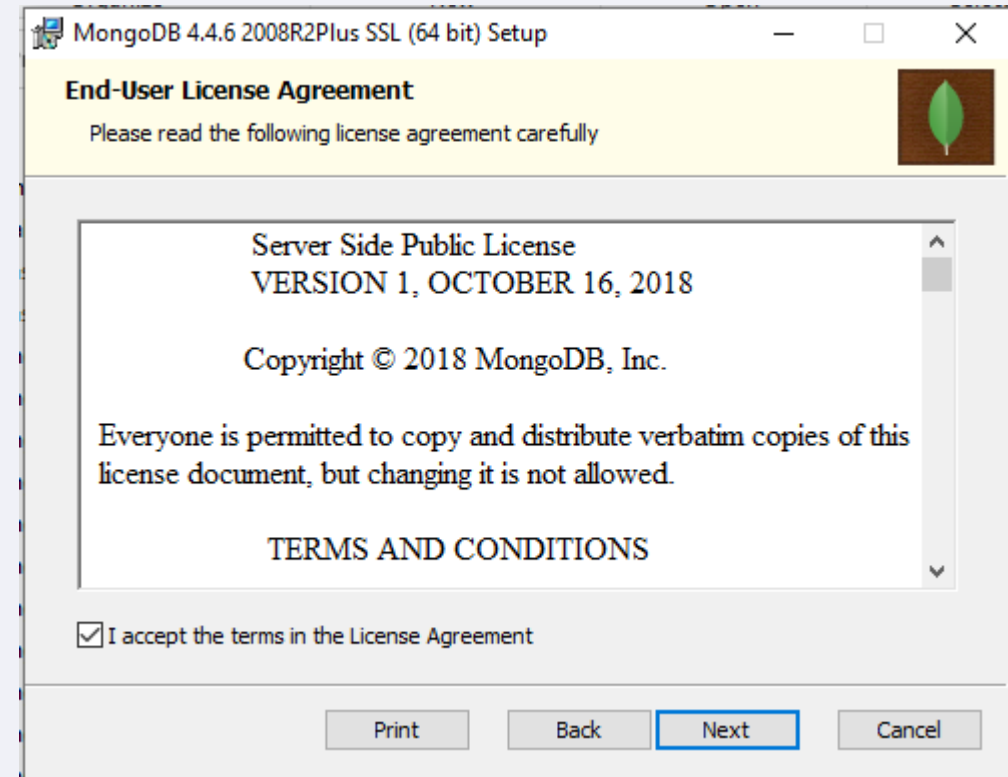
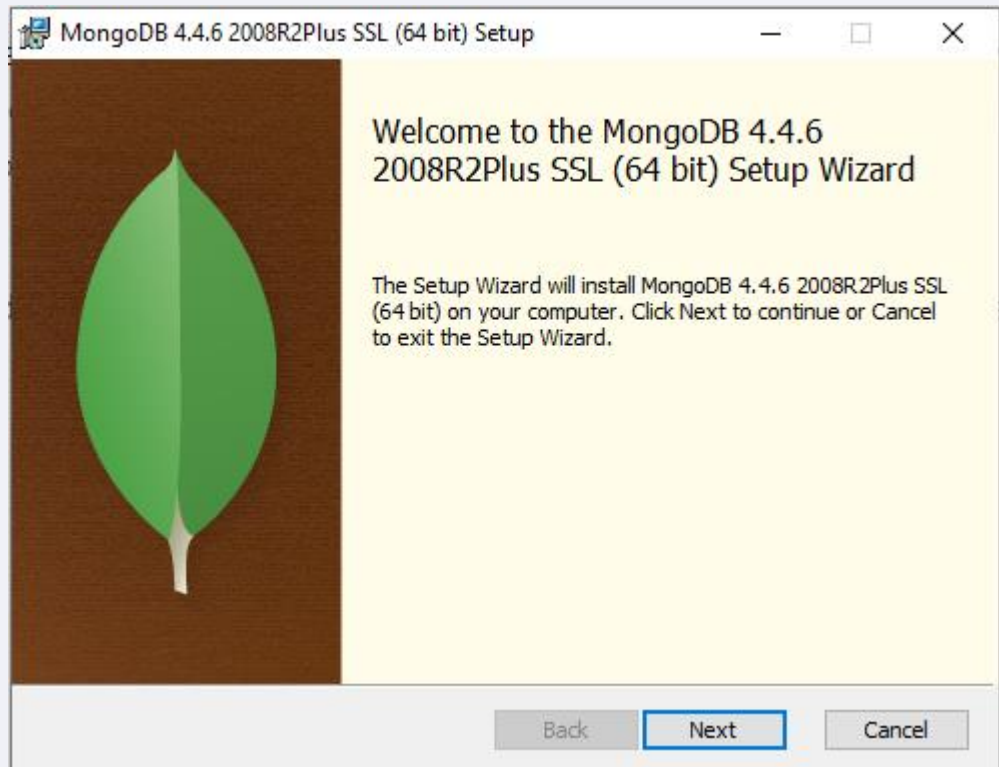
[Changelog](#)

[Release Notes](#)

MongoDB: Install



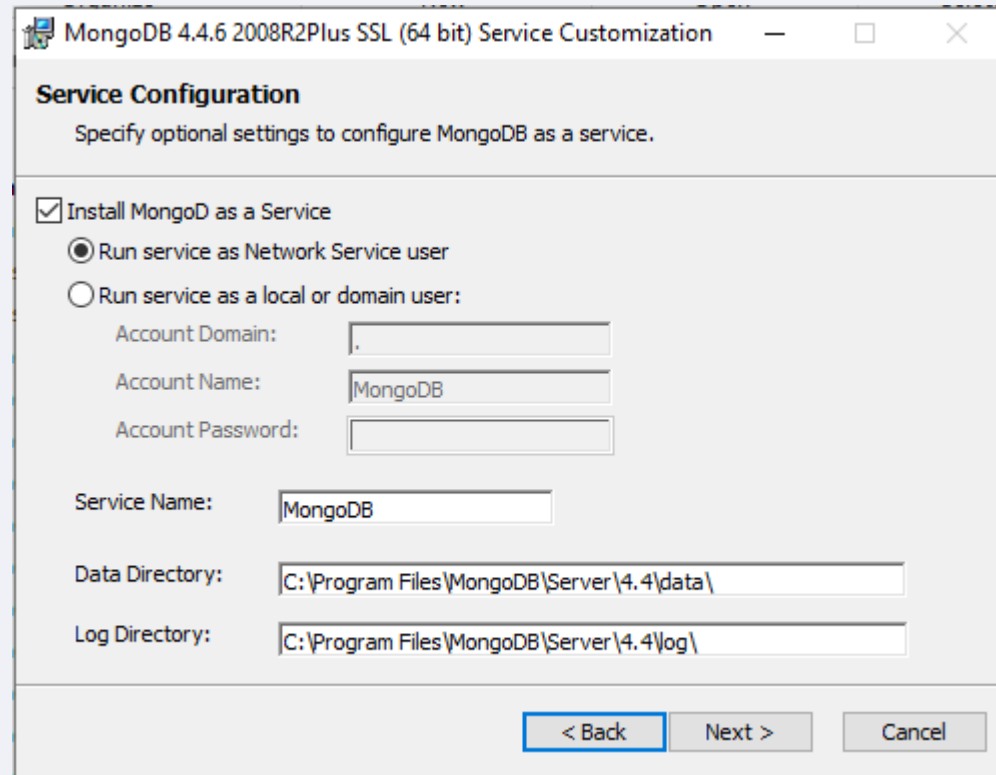
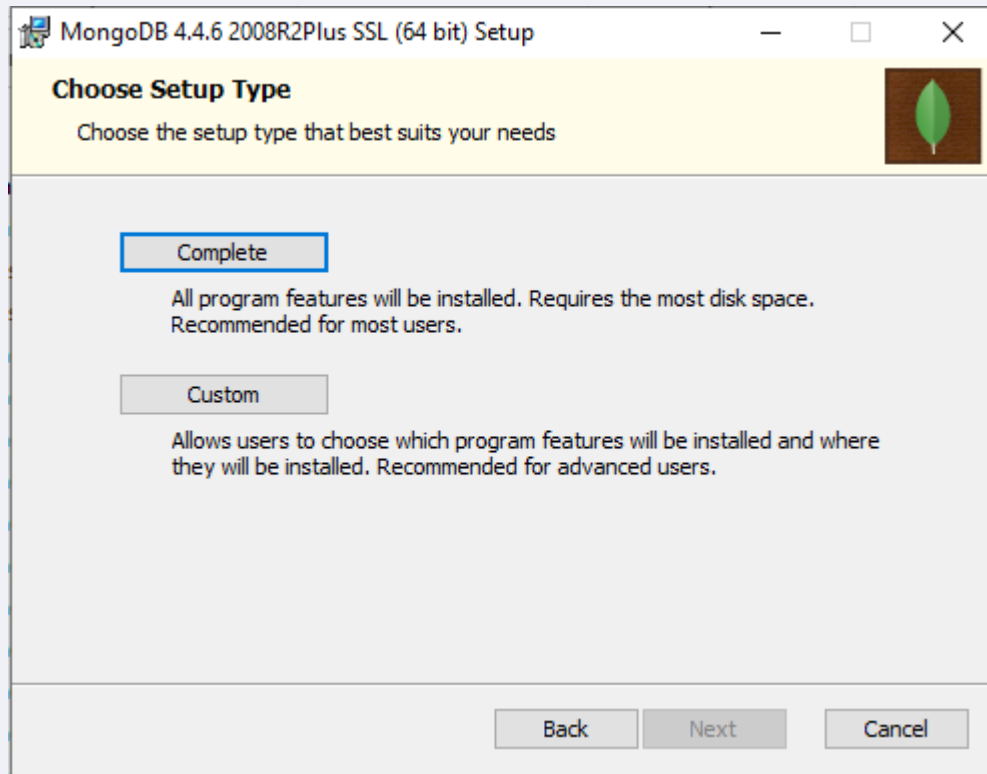
- Step through the installation wizard



MongoDB: Install



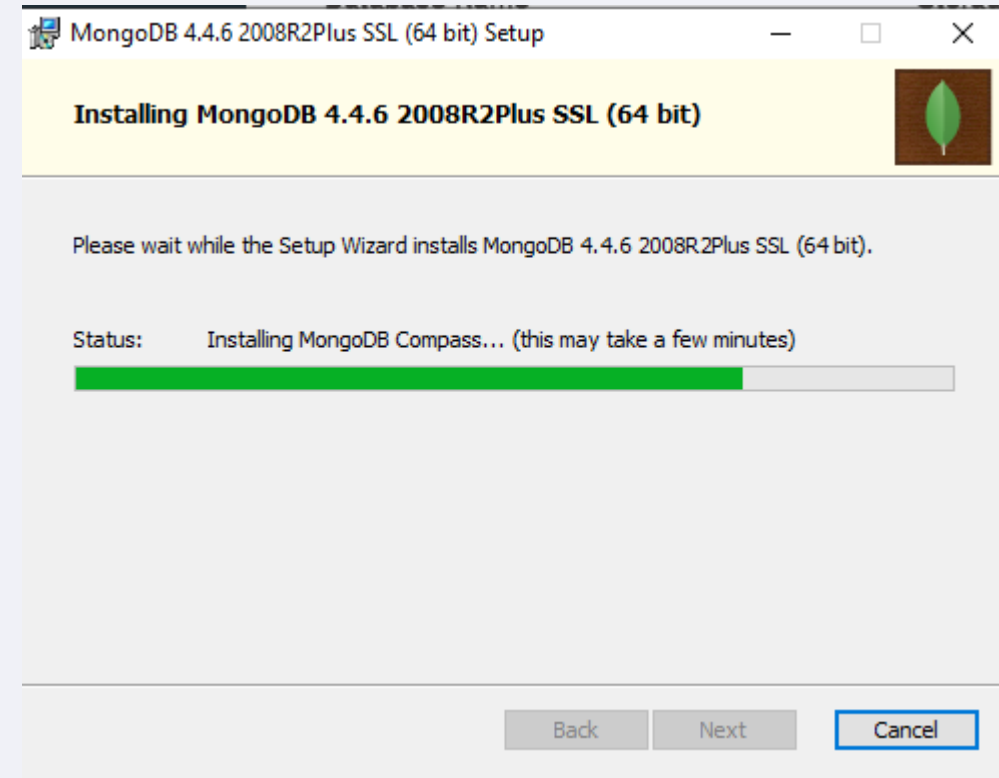
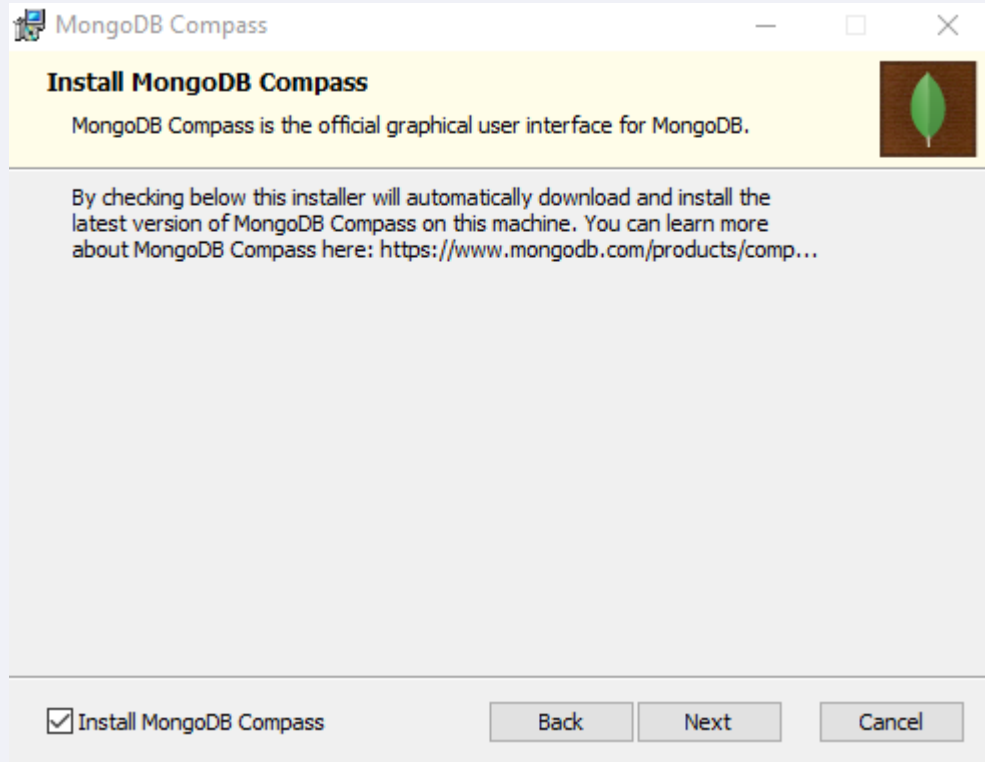
- Step through the installation wizard
- Notice the installation directory



MongoDB: Install



- Make sure to install Compass



MongoDB: Install



- Compass

The screenshot shows the MongoDB Compass interface for creating a new connection. The window title is 'MongoDB Compass - Connect'. On the left, there is a sidebar with 'New Connection', 'Favorites', and 'Recents'. The 'Recents' list shows several entries for 'localhost:27017' with timestamps. The main area is titled 'New Connection' with a '☆ FAVORITE' button. Below the title, it says 'Fill in connection fields individually'. A text input field is labeled 'Paste your connection string (SRV or Standard ⓘ)' and contains the example string 'e.g. mongodb+srv://username:password@cluster0-jtpxd.mongodb.net/admin'. A green 'Connect' button is positioned to the right of the input field. On the right side of the dialog, there is a light green box with the text 'New to Compass and don't have a cluster? If you don't already have a cluster, you can create one for free using [MongoDB Atlas](#). CREATE FREE CLUSTER'. Below this, there are two sections: 'How do I find my connection string in Atlas?' with the text 'If you have an Atlas cluster, go to the Cluster view. Click the 'Connect' button for the cluster to which you wish to connect.' and a link 'See example'; and 'How do I format my connection string?' with a link 'See example'.



MongoDB: Directory

- Navigate to the MongoDB bin directory

Name	Date modified	Type	Size
InstallCompass.ps1	5/7/2021 4:04 PM	Windows PowerS...	2 KB
mongo.exe	5/7/2021 4:30 PM	Application	21,199 KB
mongod.cfg	6/17/2021 1:19 PM	CFG File	1 KB
mongod.exe	5/7/2021 5:14 PM	Application	37,535 KB
mongod.pdb	5/7/2021 5:14 PM	Program Debug D...	379,900 KB
mongos.exe	5/7/2021 4:49 PM	Application	26,693 KB
mongos.pdb	5/7/2021 4:49 PM	Program Debug D...	255,988 KB

```
C:\Program Files\MongoDB\Server\4.4\bin>
```




MongoDB: Interactive Shell

- Start the interactive shell >mongo

```
Command Prompt - mongo
C:\Program Files\MongoDB\Server\4.4\bin>mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongod
Implicit session: session { "id" : UUID("32dbcd25-fc23-4297-bef2-6d9190f77d54") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
  2021-06-17T13:19:23.271-04:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you and anyone you share the URL with. MongoDB may use this information to make product improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```



MongoDB: Interactive Shell

Interactive Shell

The first thing to notice is that the MongoDB shell is JavaScript-based.

So you can do things like:

```
a = 5;
```

```
a * 10;
```

```
for(i=0; i<10; i++) { print('hello'); };
```

```
> a = 5;
5
> a
5
> a * 10;
50
> for(i=0; i<10; i++) { print('hello'); };
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
9
> |
```



MongoDB: Interactive Shell

Interactive Shell

- The first thing to notice is that the MongoDB shell is JavaScript-based.
- So, you can do things like:

```
a = 5;
```

```
a * 10;
```

```
for(i=0; i<10; i++) { print('hello'); };
```

```
> a = 5;
5
> a
5
> a * 10;
50
> for(i=0; i<10; i++) { print('hello'); };
hello
hello
hello
hello
hello
hello
hello
hello
hello
hello
9
> |
```



MongoDB: Documents

Interactive Shell

- MongoDB is a document database.
- This means that we store data as documents, which are similar to JavaScript objects.
- Here below are a few sample JS objects:

```
var a = {age: 25};
```

```
var n = {name: 'Ed', languages: ['c', 'ruby', 'js']};
```

```
var student = {name: 'Jim', scores: [75, 99, 87.2]};
```

```
> var a = {age: 25};  
> a  
{ "age" : 25 }  
> var n = {name: 'Ed', languages: ['c', 'ruby', 'js']};  
> n  
{ "name" : "Ed", "languages" : [ "c", "ruby", "js" ] }  
> var student = {name: 'Jim', scores: [75, 99, 87.2]};  
> student  
{ "name" : "Jim", "scores" : [ 75, 99, 87.2 ] }
```

Notice the name value pairs separated by the : colon



MongoDB: Saving and Querying

Interactive Shell

- Try adding some documents to the scores collection:

```
for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
```

- Try that, then enter `db.scores.find();`
- To see if the save succeeded. Since the shell only displays 10 results at time, you'll need to enter the 'it' command to iterate over the rest.

```
> for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
9
> db.scores.find();
{ "_id" : ObjectId("5325c0d51cdcaf4ab3d2cd93"), "a" : 99 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9a"), "a" : 0, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1457"), "a" : 1, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1459"), "a" : 3, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9e"), "a" : 4, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c145b"), "a" : 5, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145d"), "a" : 6, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda0"), "a" : 7, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145f"), "a" : 8, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda2"), "a" : 9, "exam" : 5 }
```



MongoDB: Basic Queries

Interactive Shell

- You've already tried a few queries, but let's make them more specific.
- How about finding all documents where $a == 2$:
`db.scores.find({a: 2});`
- Or what about documents where $a > 15$?
`db.scores.find({a: {'$gt': 15}});`

```
db.scores.find({a: 2});
```

```
db.scores.find({a: {'$gt': 15}});
```

```
> for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
9
> db.scores.find();
{ "_id" : ObjectId("5325c0d51cdcaf4ab3d2cd93"), "a" : 99 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9a"), "a" : 0, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1457"), "a" : 1, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1459"), "a" : 3, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9e"), "a" : 4, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c145b"), "a" : 5, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145d"), "a" : 6, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda0"), "a" : 7, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145f"), "a" : 8, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda2"), "a" : 9, "exam" : 5 }
```

```
> db.scores.find({a: 2});
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
> db.scores.find({a: {'$gt': 15}});
{ "_id" : ObjectId("5325c0d51cdcaf4ab3d2cd93"), "a" : 99 }
```



MongoDB: Query Operators

Interactive Shell

- \$gt is one of many special query operators

- Here are few others:

\$lt - '<', \$lte - '<='

\$gte - '>=', \$ne - '!='

\$in - 'is in array', \$nin - '! in array'

```
db.scores.find({a: {'$in': [2, 3, 4]}});
```

```
db.scores.find({a: {'$gte': 2, '$lte': 4}});
```

```
> for(i=0; i<10; i++) { db.scores.save({a: i, exam: 5}) };
9
> db.scores.find();
{ "_id" : ObjectId("5325c0d51cdcaf4ab3d2cd93"), "a" : 99 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9a"), "a" : 0, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1457"), "a" : 1, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1459"), "a" : 3, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9e"), "a" : 4, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c145b"), "a" : 5, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145d"), "a" : 6, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda0"), "a" : 7, "exam" : 5 }
{ "_id" : ObjectId("5325c29e4069473ce92c145f"), "a" : 8, "exam" : 5 }
{ "_id" : ObjectId("5325c29e1cdcaf4ab3d2cda2"), "a" : 9, "exam" : 5 }
```

```
> db.scores.find({a: {'$in': [2, 3, 4]}});
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1459"), "a" : 3, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9e"), "a" : 4, "exam" : 5 }
> db.scores.find({a: {'$gte': 2, '$lte': 4}});
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9c"), "a" : 2, "exam" : 5 }
{ "_id" : ObjectId("5325c29d4069473ce92c1459"), "a" : 3, "exam" : 5 }
{ "_id" : ObjectId("5325c29d1cdcaf4ab3d2cd9e"), "a" : 4, "exam" : 5 }
```



MongoDB: Updates

Interactive Shell

- Now create a couple documents like these for updating:

```
db.users.save({name: 'Johnny', languages: ['ruby', 'c']});
```

```
db.users.save({name: 'Sue', languages: ['scala', 'lisp']});
```

- Make sure they were saved by called `db.users.find()`

- Update the first document like so:

```
db.users.update({name: 'Johnny'}, {name: 'Cash', languages: ['english']});
```

```
> db.users.save({name: 'Johnny', languages: ['ruby', 'c']});
> db.users.save({name: 'Sue', languages: ['scala', 'lisp']});
> db.users.find();
{
  "_id" : ObjectId("5325c53b1cdcaf4ab3d2ce08"),
  "languages" : [
    > "ruby",
    > "c"
  ],
  "name" : "Johnny"
}
{
  "_id" : ObjectId("5325c5424069473ce92c14c3"),
  "languages" : [
    > "scala",
    > "lisp"
  ],
  "name" : "Sue"
}
```

```
> db.users.update({name: 'Johnny'}, {name: 'Cash', languages: ['english']});
> db.users.find({name: 'Cash'});
{
  "_id" : ObjectId("5325c53b1cdcaf4ab3d2ce08"),
  "languages" : [
    > "english"
  ],
  "name" : "Cash"
}
```




MongoDB: Update Operators

Interactive Shell

- The previous update replaced the entire document, but MongoDB also supports partial updates to documents.

For example, you can set a value:

```
db.users.update({name: 'Cash'}, {'$set': {'age': 50}});
```

```
> db.users.find({name: 'Cash'});
{
  "_id" : ObjectId("5325c53b1cdcaf4ab3d2ce08"),
  "languages" : [
    > "english"
  ],
  "name" : "Cash"
}
```

```
> db.users.update({name: 'Cash'}, {'$set': {'age': 50}});

> db.users.find({name: 'Cash'});
{
  "_id" : ObjectId("5325c53b1cdcaf4ab3d2ce08"),
  "languages" : [
    > "english"
  ],
  "age" : 50,
  "name" : "Cash"
}
```



MongoDB: Update Operators

Interactive Shell

You can also push and pull items from arrays:

```
db.users.update({name: 'Sue'}, {'$pull': {'languages': 'scala'}});
```

```
db.users.update({name: 'Sue'}, {'$push': {'languages': 'ruby'}});
```

```
> db.users.find({name: 'Sue'});  
{  
  >   "_id" : ObjectId("5325c5424069473ce92c14c3"),  
  >   "languages" : [  
  >     >   "scala",  
  >     >   "lisp"  
  >   ],  
  >   "name" : "Sue"  
}
```

```
> db.users.update({name: 'Sue'}, {'$pull': {'languages': 'scala'}});  
  
> db.users.find({name: 'Sue'});  
{  
  >   "_id" : ObjectId("5325c5424069473ce92c14c3"),  
  >   "languages" : [  
  >     >   "lisp"  
  >   ],  
  >   "name" : "Sue"  
}
```

```
> db.users.update({name: 'Sue'}, {'$push': {'languages': 'ruby'}});  
> db.users.find({name: 'Sue'});  
{  
  >   "_id" : ObjectId("5325c5424069473ce92c14c3"),  
  >   "languages" : [  
  >     >   "lisp",  
  >     >   "ruby"  
  >   ],  
  >   "name" : "Sue"  
}
```



MongoDB: Deleting Data

Interactive Shell

- To delete matching documents only, add a query selector to the remove method:

```
db.users.remove({name: 'Sue'});
```

- To delete everything from a collection:

```
db.scores.remove();
```

```
> db.users.remove({name: 'Sue'});
> db.users.find();
{
  "_id" : ObjectId("5325c53b1cdcaf4ab3d2ce08"),
  "languages" : [
    > "english"
  ],
  "age" : 50,
  "name" : "Cash"
}
```

```
> db.users.remove();
> db.users.find();
```

Prinzipien des Webdesigns

Layout, Farben, Texturen, Typographie

Prof. Dr. Andrej Bachmann



Nicht jede Webseite braucht ein gutes Webdesign aber jede Webseite kann eins haben

- Ein *gutes Webdesign* bringt Elemente einer Seite in *Beziehung* zu einander und behält gleichzeitig *Balance* zwischen diesen
- Modeerscheinungen kommen und gehen, aber gutes Design ist *zeitlos*
- *Feinschliff* ist für ein gutes Design *unentbehrlich*
- *Künstlerische Begabung* ist hilfreich aber *nicht ausschlaggebend*, um gut aussehende und gleichzeitig funktionale Webseiten zu entwickeln



- Der Kunde ist da
- Was sind die nächsten Schritte?

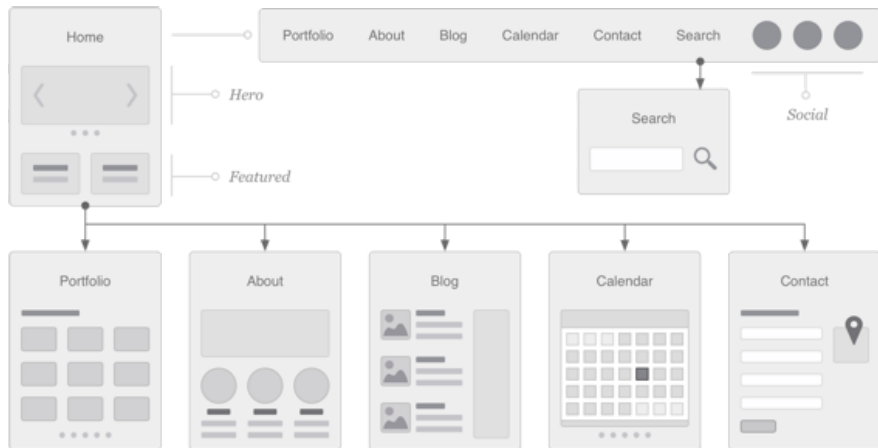
- Die erste Phase ist mit der Analysephase von Software Engineering vergleichbar
- So *vielen Informationen* wie möglich *vorab sammeln*
- Fragen, Fragen, Fragen, ...
 - ▶ Was macht die Firma?
 - ▶ Was ist das Ziel der Webseite?
 - ▶ Welche Informationen sollen bereitgestellt werden?
 - ▶ Wer ist die Zielgruppe?
 - ▶ Haben die Konkurrenten eine Webseite?
 - ▶ Welche Webseiten mag der Kunde und welche nicht?
 - ▶ Welcher Zeitrahmen und welches Budget stehen zur Verfügung?

- *Versetzen* Sie sich in den Besucher der Webseite
- Welche Informationen wünscht sich der Besucher?
- Ist die *Botschaft* klar?
- Wie viele *Klicks* braucht ein Besucher, um die Information zu erreichen?
- In dieser Phase arbeiten Sie noch von dem *Design unabhängig*
- Nützliche Werkzeuge
 - ▶ Große Wand und eine Packung Haftnotizen
 - ▶ Tools, um Wireboards zu erstellen

Denken Sie immer daran:

Form folgt der Funktion

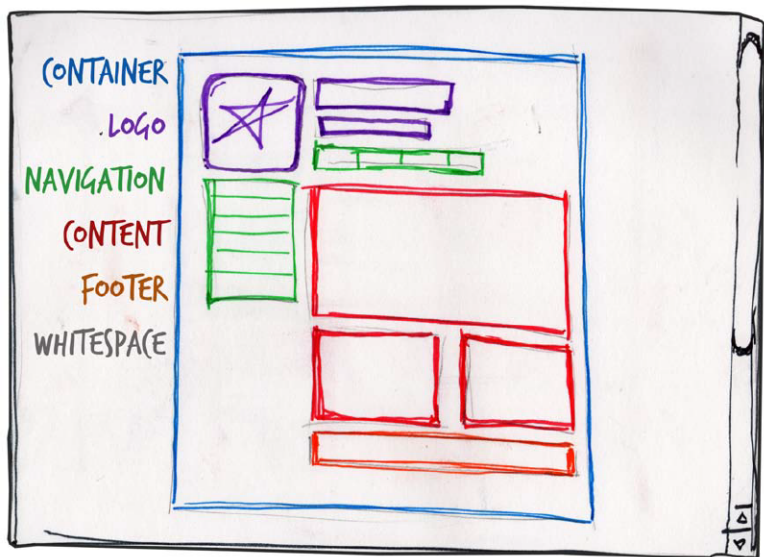
Architektur der Information



- Umsetzung beginnt mit dem *Layout*
- Wichtig ist: Beginnen Sie nicht mit einem HTML/CSS-Prototypen!
- *Papier und Stifte* oder *Grafikprogramme* sind die primären Werkzeuge
- Machen Sie *mehrere Skizzen*
- Erst dann modellieren Sie eine davon in einem Grafik-Programm endgültig
- Das Ergebnis ist ein Bild, an dem Sie *mit dem Kunden diskutieren* können



Anatomie einer Webseite



- Raster erfüllen beim Webdesign *zwei essentielle Funktionen*
 - ▶ Ausrichten von Elementen
 - ▶ Erhalten von Proportionen
- Raster in der Grafik wurden von dem niederländischen Künstler Piet Mondrian eingeführt
- Grundlegende Ideen dafür existieren aber seit Jahrtausenden
 - ▶ Goldener Schnitt (1,618033, 137,5°)
- Und in der Natur von Beginn an



Zwei verbreite Rastersysteme

- Drittel-Regel

- ▶ Eine der *einfachsten* und *effektivsten Regeln* den Raster aufzubauen
- ▶ Man braucht nicht mal Taschenrechner dafür

- 960-Rastersystem

- ▶ Alle modernen Monitore unterstützen mindestens 1024 x 768 Pixel
- ▶ 960 ist teilbar durch 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 20, 24, 30, 32, 40, 48, 60, 64, 80, 96, 120, 160, 192, 240, 320 und 480
- ▶ Dadurch ist das System *extrem flexibel*
- ▶ Viele Webseiten sind in diesem Raster aufgebaut
- ▶ Es stehen ein Generator und eine CSS-Bibliothek zur Verfügung



10K APART

8 DAYS LEFT ENTRIES DUE
8/25/2010 5PM PST

Inspire the web with just 10K.

It's time to get back to basics — back to optimizing every little byte like your life depends on it. Your challenge? Build a web app in less than 10 kilobytes.

[ENTER AN APP](#) [EXPLORE THE APPS](#)

Prizes!

Over \$10K in prizes

ONE GRAND PRIZE
Best Overall App in Contest

- Registration to any AEA Event
- \$3000 Visa Cash Card
- *HTML5 for Web Designers*
(The very first [A Book Apart](#))

THREE RUNNERS UP
Best Design, Best Technical, People's Choice

- Registration to any AEA Event
- \$1000 Visa Cash Card
- *HTML5 for Web Designers*

THE RULES

[FAQS](#)

SIZE — Total file size, including images, scripts & markup, can't be over 10K.

STANDARDS — We encourage HTML5, and apps must work equally well in IE9, Dev Preview, Firefox and a WebKit browser.

LIBRARIES — You can use one of these libraries, and it won't count against your 10K.

BONUS — 9 Honorable Mentions will win a copy of *HTML5 for Web Designers*.

- Balance ist für eine *harmonische Darstellung* einer Webseite entscheidend
- Es gibt zwei Formen
 - ▶ Symmetrisch
 - ▶ Asymmetrisch
- Typisch für die Webseiten ist die vertikale Symmetrie
- Im Logo-Design und beim Druck kommen außerdem noch bilaterale und kreisförmige symmetrische Balance vor
- Asymmetrische Balance ist *abstrakter* aber auch *spannender*
 - ▶ Z.B. Ein großes Objekt auf einer Seite wird durch mehrere kleinere Objekte auf der anderen Seite ausgeglichen
- Nimmt man ein Element einer balancierten Seite weg, geht die harmonische Wirkung verloren
 - ▶ Man hat das Gefühl es würde *etwas fehlen*

Beispiele für Balance

Comment Trifecta

Let's face it, comments on blog posts aren't what they used to be. There's just too many ways to consume information. With [Refresh Columbia](#) in particular, meetup announcements start as blog posts, get automatically syndicated as a MailChimp email campaign, posted as a note on the Facebook fan page wall, read in a bunch of other RSS readers and announced on Twitter. Many of those steps (Specifically Facebook and Twitter) provide people external channels to communicate about the content here. I was trying to figure out a way to tie some of those conversations together and decided to give the [WPBook WordPress plugin](#) a try. According to ...

Bonjour, my name is Jason!

I like to climb on things. Can I have a banana?
Eep. Eep. I'm not really a monkey, I just play one on the internet. In real life, I'm a web designer/developer. Here on my personal site, I tend to sidestep all things professional for more trivial faire. When I do write or speak about what I *actually* do for a living, I tend to do so [elsewhere](#). Feel free to [drop me a line](#) if you have questions, or use that frighteningly descriptive hyperlink below to learn more.



ABOUT STEINWAY

Steinway is dedicated to making the finest pianos in the world

Steinway & Sons was founded in 1853 by German immigrant Henry Engelhard Steinway in a Manhattan loft on Varick Street. Over the next thirty years, Henry and his sons, C. F. Theodore, Charles, Henry Jr., William, and Albert, developed the modern piano. They built their pianos one at a time, applying skills that were handed down from master to apprentice, generation after generation.

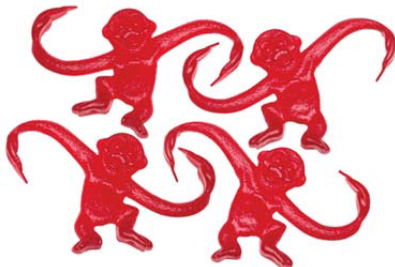
Each Steinway grand piano, for example, takes nearly a year to create. Nothing is hurried.

Today, we still build our pianos that way. Each Steinway grand piano, for example, takes nearly a year to create. Nothing is hurried. Even the carefully selected woods employed in the rims, tops, soundboards, and actions cure for months in our yard, kilns and conditioning rooms, until they stabilize at a rigidly specified moisture content.



Grundsätze der Design-Theorie

Eine Übersicht



- Unity-Theory: Webseite als eine Einheit
 - ▶ Nähe: Bildung von Gruppen durch Nahes
 - ▶ Wiederholung: Bildung von Gruppen durch ein gemeinsames Erkennungsmerkmal

Grundsätze der Design-Theorie

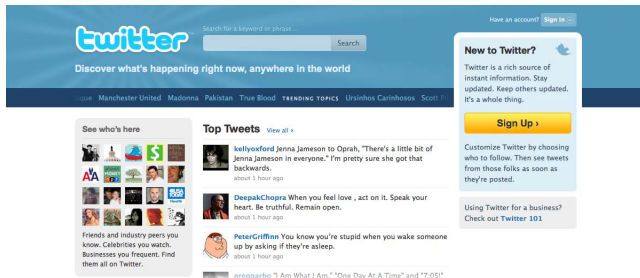
Eine Übersicht



- Betonung oder Dominanz: Ist ein anders wichtiges Konzept bei der Gestaltung von Webseiten
 - ▶ Platzierung: *Zentrum einer Komposition* ist das Element, das der Besucher als erstes anschaut
 - ▶ Kontinuität: Unsere *Augen sind träge*: Wenn sie einer Richtung folgen, verlassen sie diese nur, wenn ein anderes dominierendes Element kommt

Grundsätze der Design Theorie

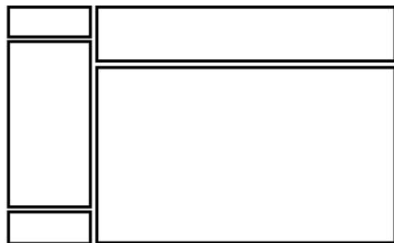
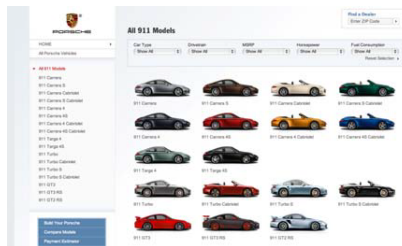
Eine sehr oberflächliche Übersicht



- Isolation: Ein Element, dass abgetrennt von den anderen steht, zieht die Aufmerksamkeit an
- Kontrast: Je größer der Kontrast eines Elements zu der Umgebung ist, desto stärker rutscht es in der Vordergrund
- Proportion: Stimmt die Proportion nicht, erweckt dass die Aufmerksamkeit

Brot und Butter Layouts

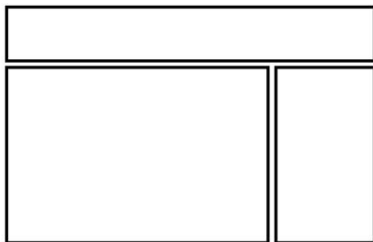
Linke Navigation



- Hat bis jetzt *alle Zeiten überstanden*
- Damit sind sehr viele kreative Entwürfe möglich
- Gleichzeitig wurde damit schon so vieles umgesetzt, dass jeder neuer Entwurf *vertraut* vorkommt
- Momentan scheint nicht mehr attraktiv für innovative Webseiten zu sein
- Dafür aber quasi Standard für Unternehmenswebseiten

Brot und Butter Layouts

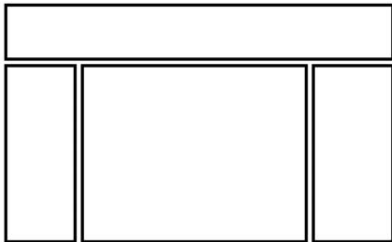
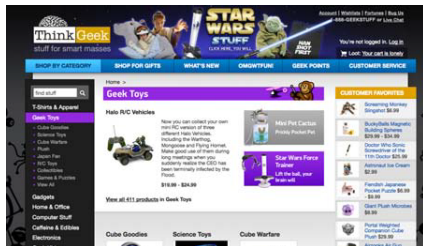
Rechte Navigation



- Diese Variante betont den Inhalt
- Sie ist zur Zeit sehr populär und wird von vielen Social Platforms verwendet
- Sie ist *ergonomischer für die Rechtshändler*

Brot und Butter Layouts

Drei Spalten



- Ist vor allen bei Online-Shops populär
- Erlaubt, *mehrere Arten von verschiedenen Informationen gleichzeitig* zu präsentieren

Wenn die Inspiration auf sich warten lässt ;-)

- Es gibt eine Reihe von verschiedenen Webseiten, die Entwicklungen in dem Webdesign verfolgen und protokollieren
- Schwerpunkte sind unterschiedlich
 - ▶ Design Pattern
 - ▶ Galerien
- Nützlich sind sie allemal, um eigene Ideen zu entwickeln
 - ▶ <http://unmatchedstyle.com/>
 - ▶ <http://cssdrive.com/>
 - ▶ <http://patterntap.com/>
 - ▶ <http://developer.yahoo.com/ypatterns/>

- Farben spielen im Webdesign eine wichtige Rolle
- Eine richtige Farbwahl ist nicht einfach
- Viele Faktoren spielen eine Rolle
 - ▶ Ästhetik
 - ▶ Identität
 - ▶ Benutzbarkeit
- Beachten Sie, dass die Wirkung einer Farbe individuell ist und von der kulturellen Gegebenheiten stark abhängig
- Was würden Sie mit den folgenden Farben assoziieren?
 - ▶ rot, orange, gelb, grün, blau, purpur, weiß, schwarz



- Leidenschaft
- Dramatik
- Spannung
- Aufregung



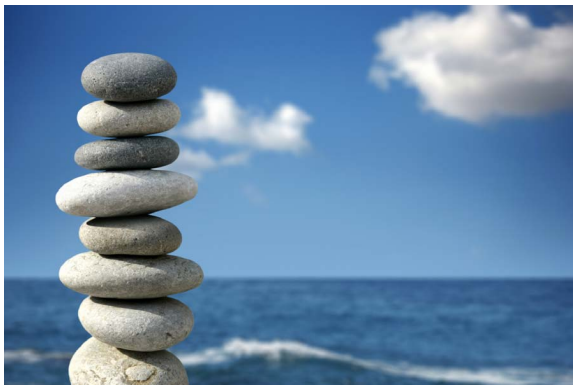
- Frische
- Energie
- Enthusiasmus



- Glück
- Sonnenschein

```
Autonagical loader: Initializing FVC Framework...  
Hello, Jason!  
We're pre-loading some files  
to make MailChimp run faster.  
=====  
OPTIONS:  
quit pre-loading = [ Q ]  
toggle strongbad = [ T ]  
pre-load AGAIN = [ A ]  
81> .....  
82> .....  
83> .....  
84> .....  
85> .....  
86> .....  
87> .....  
88> .....  
89> .....  
90> .....  
91> .....  
92> .....  
93> .....  
94> .....  
95> .....  
96> .....  
97> .....  
98> .....  
99> .....  
100> .....  
101> .....  
102> .....  
103> .....  
104> .....  
105> .....  
106> .....  
107> .....  
108> .....  
109> .....  
110> .....  
111> .....  
112> .....  
113> .....  
114> .....  
115> .....  
116> .....  
117> .....  
118> .....  
119> .....  
120> .....  
121> .....  
122> .....  
123> .....  
124> .....  
125> .....  
LOADING COMPLETE.
```

- Wachstum
- Natur
- Stabilität



- Melancholie
- Klarheit
- Glaube



- Königlich
- Kraft
- Extravaganz



- Perfektion
- Licht
- Reinheit



- Tod
- Böse
- Eleganz
- Kraft

Farbmodelle

- Die Geschichte der Farbmodelle beginnt in dem 3. Jahrhundert v. Chr.
- Seit dem haben viele bekannte Philosophen, Künstler, Schriftsteller dazu beigetragen
 - ▶ Isaac Newton
 - ▶ Johann Wolfgang von Goethe
 - ▶ Johannes Itten



- Vielmehr als allgemeine Farbmodelle spielen *Farbschemas* in dem Webdesign eine wichtige Rolle
- Folgende Varianten werden häufig benutzt
 - ▶ Monochromatisch
 - ▶ Analog
 - ▶ Komplementär
 - ▶ Gesplittet Komplementär
 - ▶ Triade
 - ▶ Tetratisch (Doppelt-komplementär)
- Es gibt sehr nützliche Tools im Web mit denen eine Passende Farbauswahl getroffen werden kann
 - ▶ <http://colorshemedesigner.com/>
 - ▶ <http://kuler.adobe.com/>



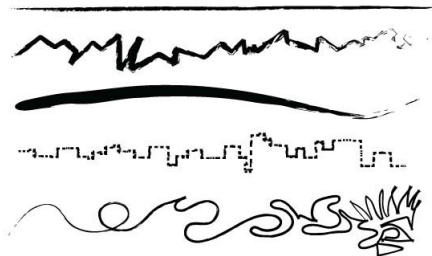




- Grundsätzliche Bausteine sind
 - ▶ Pixel
 - ▶ Linien
 - ▶ Formen
 - ★ geometrisch
 - ★ frei
- Texturen bilden die *"Oberfläche"* einer Webseite
- Sie bestimmen, *welche Wirkung* eine Seite erzeugt
- Texturen helfen *Volumen und Tiefe* darzustellen
- Bzw. Muster durch *Wiederholung* zu erzeugen

Linien

- Linien werden vielfältig im Webdesign angewendet
- So kann eine diagonal verlaufende Linie für Spannung bzw. Aufmerksamkeit sorgen
- Eine zackige Linie - Gefahr oder Hektik
- Eine abgerundete kurvige Linie - Entspannung und Geschmeidigkeit



- Boxmodell ist prädestiniert, mit geometrischen insbesondere rechteckigen Formen zu arbeiten
 - ▶ im einfachsten Fall reicht es, die *Umrandung zu aktivieren*
- Gleichzeitig ist das ein Grund, wieso beim Webdesign freie Formen oft übersehen werden
- Dabei profitiert eine Webseite davon, wenn man den *Benutzer davon ablenkt*, dass sie rechteckig aufgebaut ist
 - ▶ Eine der typischen Vorgehensweisen um das zu erreichen, ist die Nutzung von Hintergrundbildern
 - ★ die eine freie Form enthalten
 - ★ die sich über die Box-Grenzen zusammensetzen
 - ▶ Auch *abgerundete Ecken* einer Umrandung sind ein hilfreiches Mittel, um dies zu erreichen

Abgerundete Ecken

- Abgerundete Ecken war ein lang ersehntes Feature
- Wegen Browser-Inkompatibilitäten und einer relativ späten Einführung sind *viele Notlösungen* entstanden
 - ▶ Kombination aus zusätzlichen HTML-Elementen mit CSS
 - ▶ JavaScript
- Sie sollen *nicht mehr verwendet* werden
- Die `border-radius`-Regel wird inzwischen von allen modernen Browsern verstanden

```
border-top-left-radius: 2px;  
border-top-right-radius: 2px;  
border-bottom-right-radius: 2px;  
border-bottom-left-radius: 2px;
```

```
<!-- Bzw. auf einen Schlag: -->  
border-radius: 2px;
```

- Rotierte Figuren und Elemente erzeugen eine vergleichbare Wirkung wie die diagonalen Linien
 - ▶ In diesem Fall ist die Wirkung in der Regel *stärker ausgeprägt*
- Mit der CSS-Eigenschaft `transform` steht dafür ein *mächtiges Werkzeug* zur Verfügung
 - ▶ Nur unter Safari ist eine *herstellerspezifische Angabe* erforderlich

```
-webkit-transform: rotate(7.5deg); /* Safari 3.1+ */  
-ms-transform: rotate(7.5deg); /* IE 9 */  
transform: rotate(7.5deg); /* Firefox 16+, IE 10+, Opera */
```

CSS transform-Eigenschaft

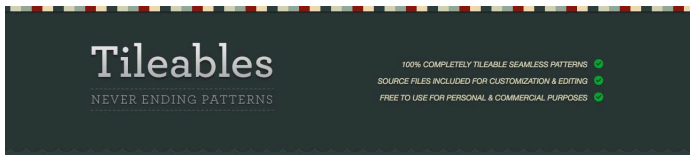
- Neben der Rotation werden noch *viele weitere Effekte* unterstützt
 - ▶ Translation: `translate`
 - ▶ Skalieren: `scale`
 - ▶ Strecken: `skew`
- Noch *flexibler* lassen sich diese Einstellungen mit den Matrizen umsetzen: `matrix`
- Die Transformationen funktionieren sowohl in 2D als auch 3D

```
transform: translate(50px,100px);  
transform: scale(2,4);  
transform: skew(30deg,20deg);  
transform: matrix(0.866,0.5,-0.5,0.866,0,0);
```

```
<!-- 3D Beispiel -->  
perspective: 300px;  
transform: rotateY(180deg);  
transform-style: preserve-3d;
```

- Weitere Gestaltungselemente sind Schatten und Beleuchtung eines Objekts
- Früher nur mit Hintergrundbildern realisierbar
- Inzwischen stellt CSS Eigenschaften `box-shadow`, `text-shadow` und die Einstellung `linear/radial-gradient` zur Verfügung

```
box-shadow: 10px 5px 15px #888888;  
text-shadow: 5px 5px 10px black;  
background: radial-gradient(circle, red, yellow, green);  
background: linear-gradient(to right, red, orange, yellow);
```



- Hintergrundbilder sind immer noch Gestaltungselement Nummer 1

```
background-color: #FF9900;  
background-image: url('animalcracker.png');  
background-repeat: repeat-x;  
background-position: left bottom;  
background-attachment: fixed;  
  
background: #FF9900 url('animalcracker.png') repeat-x left  
bottom fixed;
```

- Primäre Aufgabe einer Webseite ist die *Informationsübermittlung*
- Text spielt in diesem Kontext eine wichtige Rolle
- Es ist mehr als nur Ausrichtung, Größe und Farbe einer Schrift



- Der Browser kann nur dann eine Schriftart korrekt anzeigen, wenn sie auf dem Rechner installiert ist
- Inzwischen kann man davon ausgehen, dass die rechts dargestellten Schriftarten auf jedem Rechner vorhanden sind
- Eine dieser Schriftarten soll immer als eine *Rückgrifflösung* definiert werden

Arial

Arial Black

Comic Sans MS

Courier New

Georgia

Impact

Times New Roman

Trebuchet MS

Verdana

Ausgesuchte Schriftarten benutzen

- Eine spezielle Schriftart kann mit `font-face` einem Browser zugänglich gemacht werden
- Diese Möglichkeit soll sparsam nur für die wichtigen Elemente benutzt werden
- Die Seiten wie <http://typekit.com> bieten sehr schöne professionelle Schriftarten an
- Eine komplett kostenfreie Auswahl stellt <http://code.google.com/webfonts> zur Verfügung

```
@font-face {  
  font-family: "League Gothic";  
  src: url("/type/league_gothic.otf") format("opentype");  
}  
h1 {  
  font-family: "League Gothic", Arial, sans-serif;  
}
```


Weitere Textformatierungen

- Es gibt *zahlreiche Einstellungen* mit denen Textfluss beeinflusst werden kann

```
text-align: justify;
text-decoration: underline;
text-transform: uppercase;
text-indent: 50px; /* Einzug in der ersten Zeile */
letter-spacing: 2px;
word-spacing: 30px;
line-height: 90%;
white-space: nowrap;
vertical-align: text-top;
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt laoreet dolore magna aliequam volutpat. Ut wisi enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis a aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit vulputate velit esse.

Justified text and narrow columns, particularly narrow columns with longer words do not play well together either.

Zum größtem Teil baut diese Vorlesung auf dem folgenden Buch auf:



Jason Beaird: The Principles of Beautiful Web Design, ISBN:
978-0-9805768-9-4, SitePoint Pty. Ltd. 2010

WebDev 1

Bootstrap

Prof. Dr. Jürgen Heym

Hochschule Hof

1 – Bootstrap Get Started

- *Bootstrap is a free front-end framework for faster and easier web development.*
- *Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins.*
- *Bootstrap also gives you the ability to easily create responsive designs.*
- Latest stable releases are 5.1.3, and 4.6.0, and 3.4.1, and ...

Source: W3School's Website on Bootstrap: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

1 – Bootstrap Get Started

- Sources / Material
 - Official Bootstrap Website
<http://getbootstrap.com/>
 - W3School's Bootstrap Tutorial
<https://www.w3schools.com/bootstrap/>
 - Wikipedia's Page about Bootstrap
[https://en.wikipedia.org/wiki/Bootstrap_\(front-end_framework\)](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework))

Most of the material provided in these slides is published by W3School and is copyrighted by Refsnes Data, see

https://www.w3schools.com/about/about_copyright.asp

1 – Bootstrap Get Started

- **Why Use Bootstrap?**
 - **Easy to use:** Anybody with just basic knowledge of HTML and CSS can start using Bootstrap.
 - **Responsive features:** Bootstrap's responsive CSS adjusts to phones, tablets, and desktops.
 - **Mobile-first approach:** In Bootstrap 3, mobile-first styles are part of the core framework.
 - **Browser compatibility:** Bootstrap is compatible with all modern browsers.

Source: W3School's Website on Bootstrap: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

1 – Bootstrap Get Started

- **Why should you use Bootstrap Content Delivery Network (CDN)?**
 - Many users already have downloaded Bootstrap from MaxCDN when visiting another site. As a result, it will be loaded from cache when they visit your site, which leads to faster loading time.
 - Also, most CDN's will make sure that once a user requests a file from it, it will be served from the server closest to them, which also leads to faster loading time.

Source: W3School's Website on Bootstrap: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

1 – Bootstrap Get Started

- **How can we enable Bootstrap 3 mobile-first via CDN's?**
 - Introduce the following lines of code in your HTML header and don't forget jquery!

```
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"
integrity="sha384..." crossorigin="anonymous">

<!-- Optional theme -->
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap-
theme.min.css" integrity="sha384..." crossorigin="anonymous">

<!-- Latest compiled and minified JavaScript -->
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"
integrity="sha384..." crossorigin="anonymous"></script>
```

Source: W3School's Website on Bootstrap: https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

2 – Two Basic Bootstrap Pages

- Responsive fixed width container

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  --- Snip-Snap Here go the Bootstrap & jQuery scripts and links ---
</head>
<body>

<div class="container">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>

</body>
</html>
```

Try it Yourself: First Example

https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

2 – Two Basic Bootstrap Pages

- Responsive full width container

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Bootstrap Example</title>
  --- Snip-Snap Here go the Bootstrap & jQuery scripts and links ---
</head>
<body>

<div class="container-fluid">
  <h1>My First Bootstrap Page</h1>
  <p>This is some text.</p>
</div>

</body>
</html>
```

Try it Yourself: Second Example

https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

3 – Bootstrap Grids

- **Grid Features**
 - 12 columns
 - columns may be grouped
 - responsive

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4			
span 4				span 8							
span 6						span 6					
span 12											

Source: W3School's Website on Bootstrap: https://www.w3schools.com/bootstrap/bootstrap_grid_basic.asp

3 – Bootstrap Grids

■ Grid Classes

- xs (for phones)
- sm (for tablets)
- md (for desktops)
- lg (for larger desktops)

- Classes may be combined!
- Numbers should add up to 12 for each row.

- Example (two unequal columns ratio 1:2)

```
<div class="row">  
  <div class="col-sm-4">.col-sm-4</div>  
  <div class="col-sm-8">.col-sm-8</div>  
</div>
```

Try it Yourself: Last Example

https://www.w3schools.com/bootstrap/bootstrap_grid_basic.asp

4 – Bootstrap Text/Typography

■ Defaults

- Font-size: 14px
- Line-height: 1.428

■ Contextual Colors and Backgrounds

- Meaning through colors

- Classes for text colors

- ✓ `.text-muted,`
- ✓ `.text-primary,`
- ✓ `.text-success,`
- ✓ `.text-info,`
- ✓ `.text-warning,`
- ✓ `.text-danger`

- Classes for background colors

- ✓ `.bg-primary,`
- ✓ `.bg-success,`
- ✓ `.bg-info,`
- ✓ `.bg-warning,`
- ✓ `.bg-danger`

More: https://www.w3schools.com/bootstrap/bootstrap_typography.asp

4 – Bootstrap Text/Typography

- **Bootstrap CSS Typography Reference**

- For a complete reference of all typography elements/classes, see the following source

https://www.w3schools.com/bootstrap/bootstrap_ref_css_text.asp

- **Bootstrap Helper Classes Reference**

- For more information about contextual classes, see

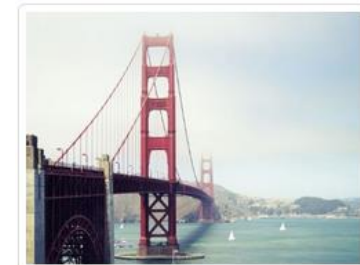
https://www.w3schools.com/bootstrap/bootstrap_ref_css_helpers.asp

5 – Bootstrap Tables

- **Table classes**
 - Basic table *.table*
 - Stripped rows *.table-striped*
 - Bordered table *.table-bordered*
 - Hover rows *.table-hover*
 - Condensed table *.table-condensed*
 - Responsive table *.table-responsive*
- **Contextual classes for table rows (<tr>) and cells (<td>)**
 - *.active, .success, .info, .warning, .danger*
- **Exercises & Try it Yourself**
 - https://www.w3schools.com/bootstrap/bootstrap_tables.asp

6 – Bootstrap Images

- **Rounded Corners**
 - Class `.img-rounded` adds rounded corners
 - IE8 does not support rounded corners
- **Circle**
 - Class `.img-circle` class shapes the image to a circle
- **Thumbnail**
 - Class `.img-thumbnail` class shapes the image to a thumbnail



6 – Bootstrap Images

- **Responsive Images**

- Responsive images automatically adjust to fit the size of the screen.
- Class `.img-responsive`
- The image will then scale nicely to the parent element.
- Is equal to: `display: block; max-width: 100%; height: auto;`

- **Image Gallery**

- Grid System + thumbnail images

- **Responsive Embeds**

- Let videos or slideshows scale properly on any device.
- Class `.embed-responsive-item`

- **Exercises & Try it Yourself**

- https://www.w3schools.com/bootstrap/bootstrap_images.asp

7 – Jumbotron & Page Header

■ Jumbotron

- Big box for calling extra attention to some special content or information
- Displayed as a grey box with rounded corners
- Enlarges the font sizes of the text inside it
- Inside a jumbotron you can put nearly any valid HTML, including other Bootstrap elements/classes.
- Class `.jumbotron`
- If you want the jumbotron to NOT extend to the edge of the screen, place it inside a `<div class="container">`.
- If you want the jumbotron to extend to the screen edges, place the jumbotron outside the `<div class="container">`.

7 – Jumbotron & Page Header

- Jumbotron Example

```
<div class="container">  
  
  <div class="jumbotron">  
    <h1>Bootstrap Tutorial</h1>  
    <p>This is a little paragraph ...</p>  
  </div>  
  
  <p>This is some text.</p>  
  
</div>
```

- Exercises & Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_jumbotron

7 – Jumbotron & Page Header

- **Page Header**

- Section divider
- Adds a horizontal line under the heading
- Adds some extra space around the element
- Class `.page-header`

- **Try it Yourself**

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_page-header

7 – Jumbotron & Page Header

- Page Header Example

```
<div class="container">  
  
  <div class="page-header">  
    <h1>Bootstrap Tutorial</h1>  
    <p>This is a little paragraph ...</p>  
  </div>  
  
  <p>This is some text.</p>  
  
</div>
```

- Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_page-header

8 – Basic Well

■ Class `.well`

- The `.well` class adds a rounded border around an element with a gray background color and some padding.
- Change the size of the well by adding the `.well-sm` class for small wells or `.well-lg` class for large wells.
- Example

```
<div class="container">  
  <div class="well well-sm">Basic Well</div>  
</div>
```

The `.well` class adds a rounded border around an element with a gray background color and some padding.

■ Try it Yourself

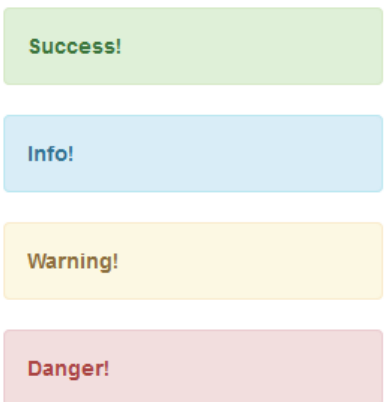
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_well

9 – Alerts

■ Alert messages

- Alerts are created with the `.alert` class, followed by one of the four contextual classes `.alert-success`, `.alert-info`, `.alert-warning` or `.alert-danger`.
- Example

```
<div class="container">  
  <div class="alert alert-success">Success!</div>  
  <div class="alert alert-info">Info!</div>  
  <div class="alert alert-warning">Warning!</div>  
  <div class="alert alert-danger">Danger!  
  </div>  
</div>
```



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_alerts

9 – Alerts

■ Alert Links

- Add the alert-link class to any links inside the alert box to create "matching colored links".
- Example

```
<div class="alert alert-success">  
  <strong>Success!</strong>  
  <a href="#" class="alert-link"> You should read this!</a>  
</div>
```

Success! You should read this message.

Info! You should read this message.

Warning! You should read this message.

Danger! You should read this message.

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_alerts

9 – Alerts

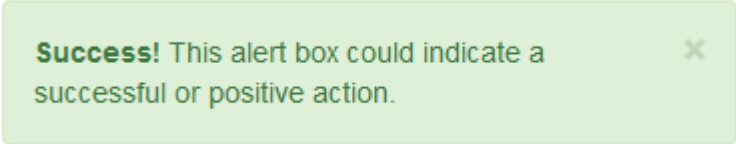
■ Alert Links & Closing Alerts

- To close the alert message, add a `.alert-dismissible` class to the `alert` container.

Then add `class="close"` and `data-dismiss="alert"` to a link or a button element.

- Example

```
<div class="alert alert-success alert-dismissible">
  <a href="#" class="close"
    data-dismiss="alert"
    aria-label="close" >
    &times;
  </a>
  Success text!
</div>
```



Success! This alert box could indicate a successful or positive action.

■ Try it Yourself

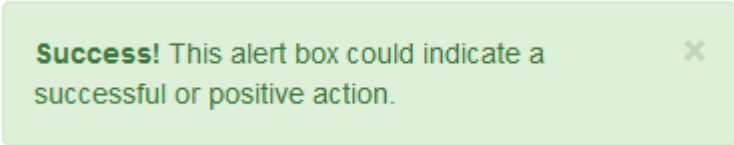
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_alerts_dismissible

9 – Alerts

■ Animated Alerts

- The `.fade` and `.in` classes adds a fading effect when closing the alert message.
- Example

```
<div class="alert alert-success alert-dismissible fade in">  
  <a href="#" class="close"  
    data-dismiss="alert"  
    aria-label="close" >  
    &times;  
  </a>  
  Success text!  
</div>
```



Success! This alert box could indicate a successful or positive action.

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_alerts_fade

10 – Buttons

■ Button Styles

- Seven button styles are predefined by bootstrap:
.btn, .btn-default, .btn-primary, .btn-success, .btn-info, .btn-warning, .btn-danger, .btn-link
- Button styles may be used with <a> and <button> tags and input type “submit” and “button”.



- Examples

```
<button type="button" class="btn btn-primary">Primary</button>  
<a href="#" class="btn btn-info" role="button">Link Button</a>  
<input type="button" class="btn btn-info" value="Input Button">
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_styles

10 – Buttons

■ Button Sizes

- Four different button sizes are predefined in bootstrap: .btn-lg, .btn-md, .btn-sm and .btn-xs



- Block level buttons span the entire width of their parent element: .btn-block
- Examples

```
<button type="button" class="btn btn-primary btn-lg">Primary</button>  
<button type="button" class="btn btn-primary btn-block">Primary</button>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_sizes
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_block

10 – Buttons

■ Active & Disabled Buttons

- Buttons can be set to be active (clickable) or a disabled state (unclickable): `.active` and `.disabled` are the classes.



- Examples

```
<button type="button" class="btn btn-primary active">  
  Active Primary</button>  
<button type="button" class="btn btn-primary disabled">  
  Disabled Primary</button>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_active

10 – Buttons

■ Button Groups

- Button may be grouped together without space between them.
- For horizontal / vertical button groups use a <DIV>-element with class .btn-group / .btn-vertical:



- Justified button groups take the width of their parent element and get class .btn-group-justified.
- Example

```
<div class="btn-group">  
  <button type="button" class="btn btn-primary">Apple</button>  
  <button type="button" class="btn btn-primary">Samsung</button>  
</div>
```

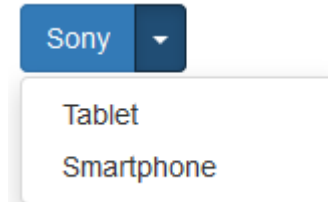
■ Try it Yourself

- https://www.w3schools.com/bootstrap/bootstrap_button_groups.asp

10 – Buttons

■ Button Dropdown Menus

- Example



```
<div class="btn-group">  
  <button type="button"  
    class="btn btn-primary dropdown-toggle"  
    data-toggle="dropdown">  
    Sony <span class="caret"></span>  
  </button>  
  <ul class="dropdown-menu" role="menu">  
    <li><a href="#">Tablet</a></li>  
    <li><a href="#">Smartphone</a></li>  
  </ul>  
</div>
```

■ Try it Yourself

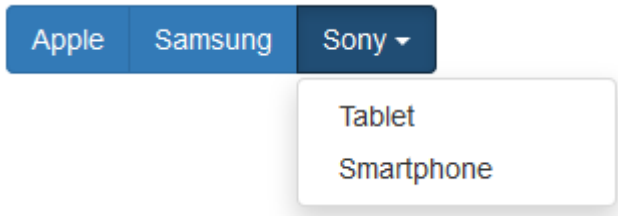
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_group_dropdown

10 – Buttons

■ Nesting Button Groups & Dropdown Menus

– Example

```
<div class="btn-group">
  <button type="button" class="btn btn-primary">Apple</button>
  <button type="button" class="btn btn-primary">Samsung</button>
  <div class="btn-group">
    <button type="button"
      class="btn btn-primary dropdown-toggle"
      data-toggle="dropdown">
      Sony <span class="caret"></span>
    </button>
    <ul class="dropdown-menu" role="menu">
      <li><a href="#">Tablet</a></li>
      <li><a href="#">Smartphone</a></li>
    </ul>
  </div>
</div>
```



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_button_group_dropdown

11 – Glyphicons

■ Glyphicons Halflings

- Bootstrap provides 260 glyphicons from the [Glyphicons Halflings](#) set.
- Glyphicons can be used in text, buttons, toolbars, navigation, forms, etc.
- The *name* part in the syntax above must be replaced with the proper name of the glyphicon.

```
<span class="glyphicon glyphicon-name"></span>
```

■ Try it Yourself

- <http://glyphicons.com/>
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_glyphs

12 – Badges & Labels

- **Badges associated with links**

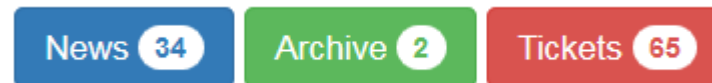
- Badges are numerical indicators and can be used with links.
- Use badges with class `.badge` within a ``-tag.
- Badges associated with links:

```
<a href="#">News
    <span class="badge">6</span>
</a>
```



- Badges inside buttons:

```
<button type="button" class="btn btn-primary">News
    <span class="badge">34</span>
</button>
```



- **Try it Yourself**


- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_badges
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_badges2

12 – Badges & Labels

■ Labels

- Provide additional information about something using labels.
- Use a span element with class label and contextual classes for coloring labels: .label-default, .label-primary, .label-success, .label-info, .label-warning or .label-danger

```
<div class="container">
  <span class="label label-default">Default</span>
  <span class="label label-primary">Primary</span>
  <span class="label label-success">Success</span>
  <span class="label label-info">Info</span>
  <span class="label label-warning">Warning</span>
  <span class="label label-danger">Danger</span>
</div>
```



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_labels

13 – Progress Bars

■ Basic Progress Bars with and without Labels

- A basic progress bar has no label. The progress is set by a width style attribute.

```
<div class="progress">  
  <div class="progress-bar"  
    role="progressbar"  
    style="width:70%"></div>  
</div>
```

```
<div class="progress">  
  <div class="progress-bar"  
    role="progressbar"  
    style="width:50%">50%</div>  
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar1
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar2

13 – Progress Bars



■ Colored Progress Bars

- Use contextual classes for coloring progress bars: `.progress-bar-success`, `.progress-bar-info`, `.progress-bar-warning` or `.progress-bar-danger`.
- Add stripes to the progress bars with class `.progress-bar-striped`.
- Animate your progress bar with class `.active`.

```
<div class="progress">  
  <div class="progress-bar progress-bar-warning"  
    role="progressbar"  
    style="width:60%">60% (warning)</div>  
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar3
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar4
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar5

13 – Progress Bars

■ Stacked Progress Bars

- Stacked progress bars are created placing multiple progress bars within the same <div>-element:

```
<div class="progress">  
  <div class="progress-bar ..." style="width:20%">Free</div>  
  <div class="progress-bar ..." style="width:30%">Used</div>  
  <div class="progress-bar ..." style="width:20%">Damaged</div>  
</div>
```



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_progressbar6

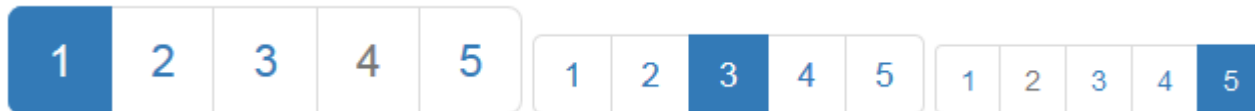
14 – Pagination

■ Basic & Advanced Pagination

- You want to show several result pages.
- Use a ``-element with class `.pagination`:

```
<ul class="pagination">  
    <li><a href="#">1</a></li> ...  
</ul>
```

- Indicate the active page with class `.active`.
- Indicate disabled pages with class `.disabled`.
- Use different sizes with classes `.pagination-lg` and `.pagination-sm`



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_pagination

14 – Pagination

■ Breadcrumbs

- Breadcrumbs are just another type of pagination.
- Use a ``-element with class `.breadcrumb`:

```
<ul class="breadcrumb">
  <li><a href="#">Home</a></li>
  ...
  <li class="active">Informatics</li>
</ul>
```

- Indicate the active element with class `.active`.



Home / Students / Module Descriptions / Informatics

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_breadcrumbs

14 – Pagination

■ Pager

- Pager is another type of pagination.
- Pager provide navigation buttons.
- Align navigation buttons with class `.previous` to the left and class `.next` to the right.

```
<ul class="pager">  
  <li class="previous"><a href="#">First</a></li>  
  <li class="previous"><a href="#">First</a></li>  
  <li class="next"><a href="#">First</a></li>  
  <li class="next"><a href="#">Last</a></li>  
</ul>
```



■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_pager_prev

15 – List Groups

■ Basic List Groups

- Unordered lists with items.
- List groups may have badges, linked items, active or disabled states and contextual classes.
- Example with badges:

```
<ul class="list-group">  
  <li class="list-group-item">  
    New Elements <span class="badge">12</span></li>  
  <li class="list-group-item">  
    Deleted Elements <span class="badge">5</span></li>  
</ul>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_list_group_badge

15 – List Groups

■ Basic List Groups

- Use classes `.active` and `.disabled` to show the active state.
- Color list items with the following contextual classes:
`.list-group-item-success`, `list-group-item-info`, `list-group-item-warning`, and `.list-group-item-danger`
- Example with linked active and disabled items:

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    First link item (active) </a>
  <a href="#" class="list-group-item disabled">
    Second link item (disabled) </a>
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_list_group_active
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_list_group_disabled
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_list_group_context

15 – List Groups

■ Custom List Groups

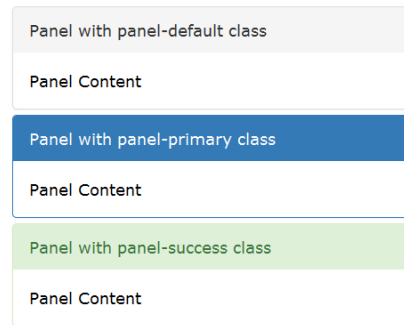
- Use nearly any HTML element inside a list group.
- For headings and text use classes `.list-group-item-heading` and `.list-group-item-text`.

```
<div class="list-group">
  <a href="#" class="list-group-item active">
    <h4 class="list-group-item-heading">
      First List Group Item Heading</h4>
    <p class="list-group-item-text">List Group Item Text</p>
  </a>
  <a href="#" class="list-group-item">
    <h4 class="list-group-item-heading">
      Second List Group Item Heading</h4>
    <p class="list-group-item-text">List Group Item Text</p>
  </a>
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_list_group_custom

16 – Panels



■ Grouped Panels

- Use class `.panel` and `.panel-default` for default colors and `.panel-body` for panel content.
- Add a panel header with class `.panel-heading` and a footer with class `.panel-footer`.
- Several panels may be grouped together with class `.panel-group`.
- Introduce colors with contextual classes as usual.

```
<div class="panel-group">
  <div class="panel panel-default">
    <div class="panel-heading">Panel Heading</div>
    <div class="panel-body">A Basic Panel</div>
    <div class="panel-footer">A Basic Panel</div>
  </div>
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/bootstrap_panels.asp

17 – Dropdowns

■ Basic Dropdown

- Use class `.dropdown` within a `div`-element.
- Define a button with class `dropdown-toggle` and link the data via a `data-toggle`-attribute.
- Tag the `ul`-element with class `.dropdown-menu`.

```
<div class="dropdown">
  <button class="btn btn-primary dropdown-toggle"
    type="button"
    data-toggle="dropdown">Dropdown Example
    <span class="caret"></span>
  </button>
  <ul class="dropdown-menu">
    <li><a href="#">HTML</a></li>
    <li><a href="#">CSS</a></li>
  </ul>
</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_dropdown-menu

17 – Dropdowns

■ Advanced Dropdown

- Introduce dividers within dropdown menus to separate groups.

```
<li class="divider"></li>
```

- Class dropdown-header gives us the ability to use headers inside a dropdown menu.

```
<li class="dropdown-header">My Header 1</li>
```

- Tag active and disabled menu items with classes .active and .disabled.
- Use class .dropup for dropup menus.

■ Try it Yourself

- https://www.w3schools.com/bootstrap/bootstrap_dropdowns.asp

18 – Collapsibles

■ Hide some DIVs

- Define a button or an a-tag that is able to toggle the visibility of a div-element.
- By default the div-element is hidden. Use class `.in` to invert the behavior.

```
<button data-toggle="collapse"  
        data-target="#demo">Collapsible</button>
```

```
<div id="demo" class="collapse">Bla bla text....</div>
```

or alternatively

```
<a href="#demo" data-toggle="collapse">Collapsible</a  
<div id="demo" class="collapse in">Bla bla text....</div>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/bootstrap_collapse.asp

18 – Collapsibles

- **Hide Panels**
- **Hide List Groups**
- **Hide Accordions**

- **Try it Yourself**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_collapsible_panel
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_collapsible_listgroup
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_collapsible_accordion

19 – Menus, Tabs & Pills

■ Menus

- Create inline menus as unordered lists.

Start Contact Login

```
<ul class="list-inline">
  <li><a href="#">Start</a></li>
  <li><a href="#">Contact</a></li>
  <li><a href="#">Login</a></li>
</ul>
```

■ Menus with Tabs

- Create inline menus with tabs as unordered lists.

Start

Contact

Login

```
<ul class="nav nav-tabs">
  <li><a href="#">Start</a></li>
  <li class="active"><a href="#">Contact</a></li>
  <li><a href="#">Login</a></li>
</ul>
```

■ Try it Yourself

- https://www.w3schools.com/bootstrap/bootstrap_tabs_pills.asp

19 – Menus, Tabs & Pills

- **Menus with Tabs & Dropdowns**

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_tabs_dropdown

- **Menus with Pills**

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_pills
- Pills are created with class nav-pills.
- Use class .active to tag the active pill.



```
<ul class="nav nav-pills">  
  <li><a href="#">Start</a></li>  
  <li class="active"><a href="#">Contact</a></li>  
  <li><a href="#">Login</a></li>  
</ul>
```

- Use class .nav-stacked for vertical pills.
- Use class .nav-justified for centered strings in tabs or pills.
- Try out pills with dropdown menus.

19 – Menus, Tabs & Pills

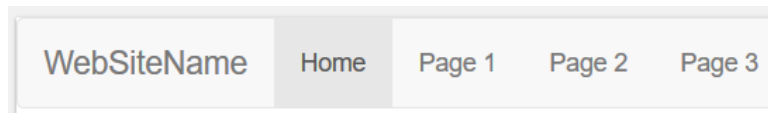
- **Dynamic Tabs & Pills**

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_tabs_dynamic
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_pills_dynamic

20 – Navigation Bars

■ Basic Navigation Bars

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar&stacked=h



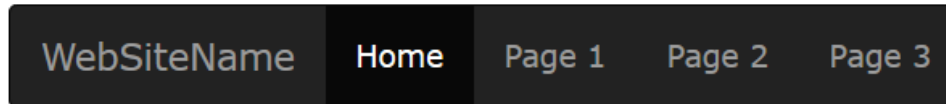
- Example: Add a basic navigation bar to the top of your page.

```
<nav class="navbar navbar-default">
  <div class="container-fluid">
    <div class="navbar-header">
      <a class="navbar-brand" href="#">Your Company</a>
    </div>
    <ul class="nav navbar-nav">
      <li class="active"><a href="#">Start</a></li>
      <li><a href="#">Contact</a></li>
      <li><a href="#">Login</a></li>
    </ul>
  </div>
</nav>
```

20 – Navigation Bars

■ Advanced Navigation Bars

- Inverted Navigation Bars: Change `.navbar-default` to `.navbar-inverted`.

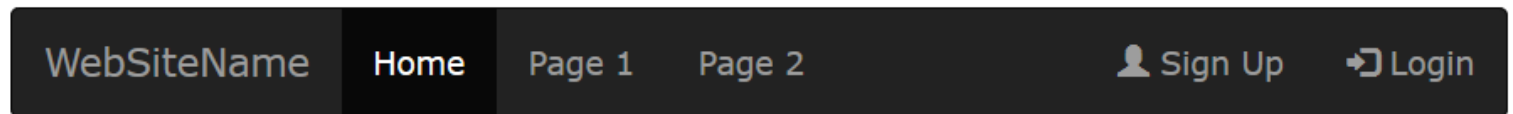


- Navigation bars may hold dropdown menus:

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_dropdown&stacked=h

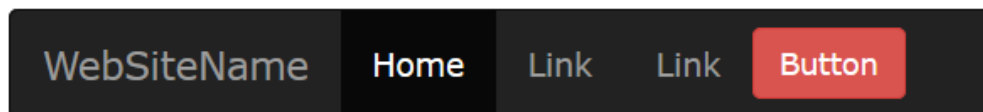
- Right-Aligned Navigation:

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_right&stacked=h



- Buttons within your Navigation:

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_btn&stacked=h



20 – Navigation Bars

■ Advanced Navigation Bars

- Forms within Navigation Bars.

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_form&stacked=h

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_form_addon&stacked=h



- Text within Navigation Bars is possible.

- Fixed Navigation Bar:

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_fixed&stacked=h

- Collapsible Bar === Hamburger Navigation

https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_navbar_collapse&stacked=h

21 – Forms

- **Form Layouts**

- Vertical Form (default)
- Horizontal Form
- Inline Form

- **Form rules**

- For optimum spacing, wrap labels and form controls in `<div class="form-group">`.
- Add class `.form-control` to all textual `<input>`, `<textarea>`, and `<select>` elements.

- **Try it out!**

- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_basic&stacked=h

Email:

Password:

Remember me

21 – Forms

- Example

```
<div class="container">  
<form>  
  <div class="form-group">  
    <label for="email">Email:</label>  
    <input type="email" class="form-control"  
      id="email" placeholder="Enter email">  
  </div>  
  
  <div class="form-group">  
    <label for="pwd">Password:</label>  
    <input type="password" class="form-control"  
      id="pwd" placeholder="Enter password">  
  </div>  
  
  <div class="checkbox">  
    <label><input type="checkbox"> Remember me</label>  
  </div>  
  
  <button type="submit" class="btn btn-default">Submit</button>  
</form>  
</div>
```

The screenshot shows a web form with the following elements: an "Email:" label above a text input field with the placeholder "Enter email"; a "Password:" label above a password input field with the placeholder "Enter password"; a checkbox labeled "Remember me"; and a "Submit" button. The form is styled with a clean, modern look, and the labels are in a bold, dark font.

22 – Input

- **Bootstrap Input Form Controls**
 - input
 - ✓ type text, password, datetime, datetime-local, date, month, time, week, number, email, url, search, tel, and color
 - textarea
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_textarea
 - checkbox
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_checkbox
 - radio
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_radio
 - select
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_select
- **Try it out!**
 - https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp

22 – Input

- Example: Inline Checkbox

Audi BMW

```
<form>
  <label class="checkbox-inline">
    <input type="checkbox" value="Audi">Audi
  </label>
  <label class="checkbox-inline">
    <input type="checkbox" value="BMW">BMW
  </label>
</form>
```

- Example: Radio Buttons

Audi

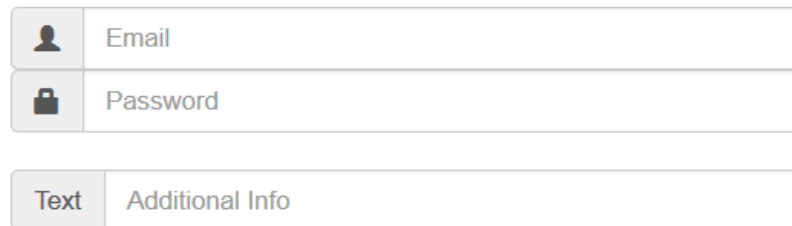
BMW



```
<form>
  <div class="radio">
    <label><input type="radio" name="optradio">Audi</label>
  </div>
  <div class="radio">
    <label><input type="radio" name="optradio">BMW</label>
  </div>
</form>
```

22 – Input

■ More Input Controls

- Inputs will NOT be fully styled if their type is not properly declared!
- With class `.form-control-static` on a `<p>` element you can display plain text next to a form label within a horizontal form.
- Input Groups
 - ✓ The `.input-group` class is a container to enhance an input by adding an icon, text or a button in front or behind it as a "help text".
 - ✓ The `.input-group-addon` class attaches an icon or help text next to the input field.



	Email
	Password
Text	Additional Info

- ✓ https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_input_group

■ Try it out!

- https://www.w3schools.com/bootstrap/bootstrap_forms_inputs.asp

22 – Input

■ Input Group Button

- Important feature for search input fields is a search button next to the input field. Use the `.input-group-btn` for the implementation.
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_input_group_btn
- Example

```
<form>
  <div class="input-group">
    <input type="text" class="form-control" placeholder="Search">
    <div class="input-group-btn">
      <button class="btn btn-default" type="submit">
        <i class="glyphicon glyphicon-search"></i>
      </button>
    </div>
  </div>
</form>
```

22 – Input

■ Form Control States

- *INPUT FOCUS* - The outline of the input is removed and a box-shadow is applied on focus
- *DISABLED INPUTS* - Add a disabled attribute to disable an input field
- *DISABLED FIELDSETS* - Add a disabled attribute to a fieldset to disable all controls within
- *READONLY INPUTS* - Add a readonly attribute to an input to prevent user input
- *VALIDATION STATES* - Bootstrap includes validation styles for error, warning, and success messages. To use, add `.has-warning`, `.has-error`, or `.has-success` to the parent element
- *ICONS* - You can add feedback icons with the `.has-feedback` class and an icon
- *HIDDEN LABELS* - Add a `.sr-only` class on non-visible labels
- https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_form_horizontal_all

Focused

Disabled

Disabled input and select list (Fieldset disabled)

Input with success and glyphicon

Input with warning and glyphicon

Input with error and glyphicon

23 – Media Objects

■ Basic Media Objects

- There is an easy way to align media objects (like images or videos) to the left or to the right of some content with bootstrap.
- https://www.w3schools.com/bootstrap/bootstrap_media_objects.asp
- Example

```
<div class="media">  
  <div class="media-left">  
      
  </div>  
  <div class="media-body">  
    <h4 class="media-heading">John Doe</h4>  
    <p>Lorem ipsum...</p>  
  </div>  
</div>
```



Left-aligned

Lorem ipsum dolor sit amet, consectetur
dolore magna aliqua.

24 – Plugins

■ **Carousel Plugin**

- The Carousel plugin is a component for cycling through elements, like a carousel (slideshow).
- https://www.w3schools.com/bootstrap/bootstrap_carousel.asp

■ **Modal Plugin**

- The Modal plugin is a dialog box/popup window that is displayed on top of the current page.
- https://www.w3schools.com/bootstrap/bootstrap_modal.asp

■ **Tooltip Plugin**

- The Tooltip plugin is small pop-up box that appears when the user moves the mouse pointer over an element.
- https://www.w3schools.com/bootstrap/bootstrap_tooltip.asp

24 – Plugins

■ Popover Plugin

- The Popover plugin is a pop-up box that appears when the user clicks on an element. The popover can contain much more content than a tooltip.
- https://www.w3schools.com/bootstrap/bootstrap_popover.asp

■ Scrollspy Plugin

- The Scrollspy plugin is used to automatically update links in a navigation list based on scroll position.
- https://www.w3schools.com/bootstrap/bootstrap_scrollspy.asp

■ Affix Plugin

- The Affix plugin allows an element to become affixed (locked) to an area on the page. This is often used with navigation menus or social icon buttons, to make them "stick" at a specific area while scrolling up and down the page.
- https://www.w3schools.com/bootstrap/bootstrap_affix.asp

25 – W3School's Bootstrap Templates

- **Blog**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_blog
- **Portfolio**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_portfolio
- **WebPage**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_webpage
- **Social**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_social
- **Analytics**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_analytics
- **Online Store**
 - https://www.w3schools.com/bootstrap/tryit.asp?filename=trybs_temp_store

Gestaltungsrichtlinien

HelfRecht

Unternehmerische Planungsmethoden AG

Stand 12. 2007

Basis

**Das HelfRecht Corporate Design:
Dauerhaft –
und in lebendigem Wandel**

Das HelfRecht Corporate Design ist als ein lebendiges System angelegt. Es bietet Spielraum zur kontinuierlichen Weiterentwicklung und stellt damit sicher, dass seine Attraktivität auf längere Sicht gewahrt bleibt.

Das hier vorliegende Manual ist deshalb in die zwei Bereiche: „Basis“ und „Medien-Umsetzungsregeln“ gegliedert.

Blick auf das Wesentliche

Der Bereich „Basis“ präsentiert die Architektur des HelfRecht Corporate Designs. Hier geht es vor allem darum, die Prinzipien des Design-Systems zu klären. Es werden nicht mehr Details als notwendig behandelt, um den Blick auf das Wesentliche zu erleichtern. Zudem sind alle hier beschriebenen Regelungen auch insofern echte Prinzipien, als sie dauerhaft gültig bleiben. Änderungen sind in Intervallen von mindestens fünf Jahren vorgesehen.

Details zu den einzelnen Medien

Der Bereich „Umsetzungsregeln“ beschreibt dort, wo akuter Bedarf auftritt, die Gestaltung einzelner Medien im Detail. Über die Design-Prinzipien hinaus, werden hier weitere grafische Einzelheiten definiert. Die Umsetzungsregeln bleiben solange gültig, bis praktische Erfahrungen mit der Regelanwendung Anlass und Substanz zur Detail-Erneuerung geben. Dies kann in kürzeren Takten erfolgen.

Durch die Gliederung in fundamentale Prinzipien mit dauerhafter Gültigkeit und Detailregeln mit begrenzter Geltungsdauer wird sichergestellt, dass das HelfRecht Design einerseits konsistent und unverwechselbar ist, zugleich wird aber auch eine hohe Lebendigkeit durch gezielten Wandel erreicht. Nicht zuletzt lässt sich das Design-System an neu auftretende Anforderungen anpassen – auf eine kostengünstige, wirtschaftlich verantwortungsvolle Art und Weise.

Eine übliche Herausforderung bei der Anwendung von Design-Systemen ist, dass die Anwender das ganze Regelwerk im Überblick haben müssen – die Kenntnis nur einzelner Regeln hilft nicht weiter. Um es den Anwendern zu erleichtern, den Überblick zu gewinnen, wurde darauf geachtet, das Regelwerk so kompakt wie nur irgend möglich zu halten.

Dennoch können immer wieder Unklarheiten und Fragen auftreten. Die persönliche Rückfrage hilft hier am schnellsten weiter. In solchen Fällen können Sie mich gerne ansprechen: Tel. 089/33 88 04 oder senden Sie ein Mail an: so@gestaltungsbuero-schultes.de

Ich wünsche Ihnen viel Erfolg und Freude mit der Anwendung des HelfRecht Design-Systems.

Sonja Schultes
Gestaltungsbüro Schultes, München
November 2007

Das HelfRecht Logo setzt sich aus zwei Elementen zusammen:
Die Wortmarke HelfRecht und das Bildzeichen



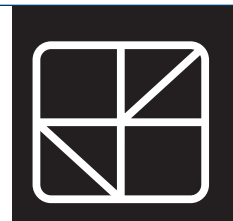
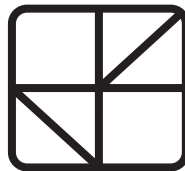
Wortmarke

Die Wortmarke von HelfRecht steht in Schwarz bzw. Weiß/negativ auf dunklem Grund.



Bildzeichen

Das Bildzeichen von HelfRecht steht immer in der Farbe der Wortmarke: in der Regel in Schwarz bzw. Weiß/negativ auf dunklem Grund, in Ausnahmefällen in Rot



In Ausnahmen kann die Wort- und Bildmarke auch in Rot (primäre Identitätsfarbe HKS 16) eingesetzt werden: wenn auf der Seite, auf der das Logo plaziert wird, weder das Rot noch farbige Winkel und Fotos verwendet werden.



Größenverhältnis Bildzeichen zu Wortmarke auf Titelseiten

Das Bildzeichen von HelfRecht steht auf Titelseiten (wenn losgelöst von der Adresse) immer in einer fixen Größe zur Wortmarke.



Logo und Adresse

In Verbindung mit der Adresse ist die Breite des Logos ein 1/5 der Wortmarke



Der Abstand des Bildzeichens zur Firmierung (Adresse) beträgt die halbe Breite des Bildzeichens. Anwendung bei Absenderangaben.

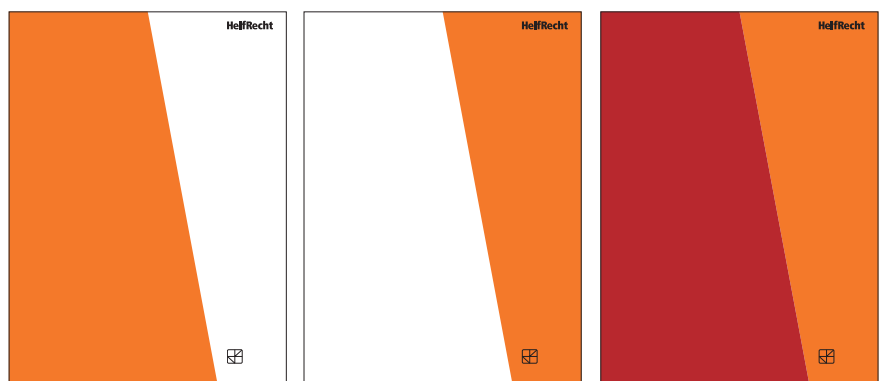


Abstand Wortmarke zur Adresse = minimum Höhe des Bildzeichens (h).

Die Wortmarke kann jedoch nach oben wandern.

Telefon 0049 (0) 92 32 / 601 - 0
Fax 0049 (0) 92 32 / 601 - 280
info@helfrecht.de
www.helfrecht.de

Ausgehend von der „Planer-Ecke“ wird das Erkennungszeichen der HelfRecht Planer auch für die Printmedien und digitalen Medien, in modifizierter Form aufgegriffen.



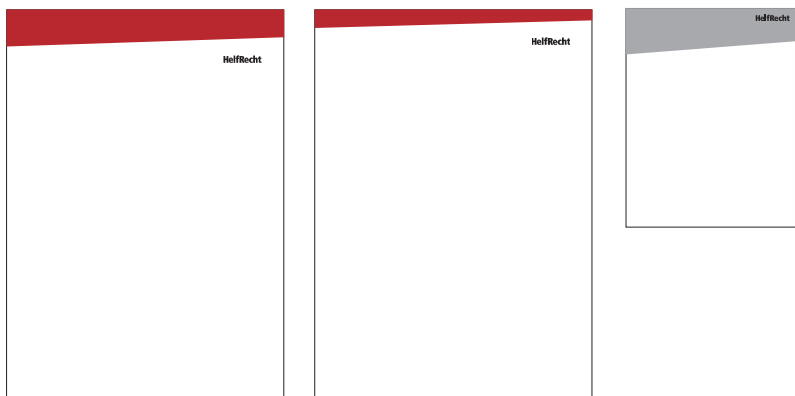
1.
Die Winkel-Flächen sind Träger für Farbflächen, Text-Informationen, Bildmotive grafische Strukturen und Illustrationen.
Hier am Beispiel mit einem Winkel von 79°



Auf der Rückseite kann der Winkel nochmals aufgegriffen werden. Die Bildmotive von Titel und Rücktitel korrespondieren.



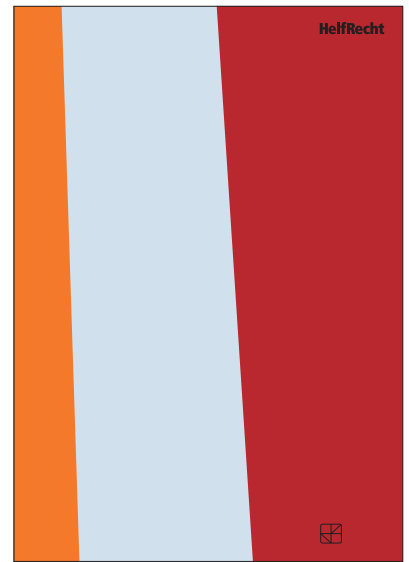
2.
Die Winkel-Flächen werden in vertikaler und in horizontaler Anordnung eingesetzt: Vertikal werden sie auf Titel- und Rück- und Innenseiten verwendet. Horizontal kommen sie auf einblättrigen Medien mit umfangreichen Texten zum Einsatz.
Zum Beispiel: Briefpapier, Mailingbögen, Newsletter, Anzeigen, etc..



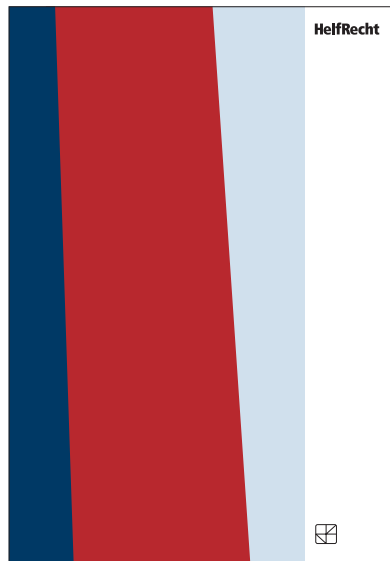
3.
Die vertikalen Flächen teilen Titelseiten in Zonen auf.
Bis zu maximal vier Zonenflächen können geschaffen werden.
Bei vier Zonenflächen ist von zwei dominierenden Flächen auszugehen, die jeweils über eine untergeordnete Fläche verfügen können.



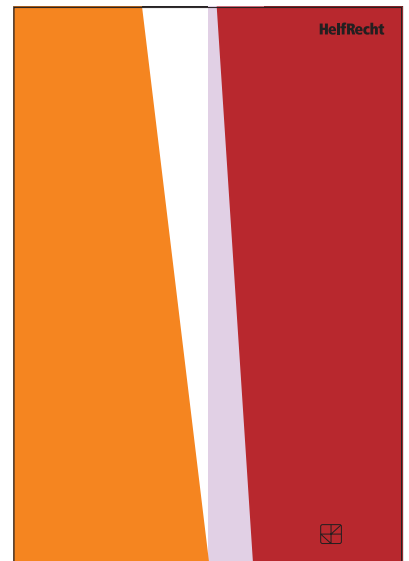
2 Zonen



3 Zonen mit 2 dominierenden Winkeln



4 Zonen mit 2 dominierenden Winkeln



4 Zonen mit 2 dominierenden Winkeln

5.
 Der Neigungswinkel der dominierenden Fläche liegt zwischen 75° und 88° .
 Messpunkt an der Basislinie unten.
 Alle Winkelungen (Stufen) zwischen 75° und 88° sind möglich (76° , 77° , 78° , 79° usw.)
 Winkel unter 75° und über 88° werden nicht verwendet.

Die Proportionen der Aufteilung der Seite durch die Flächen werden so gewählt, dass die Text- und Bildelemente überzeugend zur Geltung kommen: Die Winkel-Flächen sollen auf eine attraktive Weise „tragen“, aber nicht selbst als Blickfänger in den Vordergrund treten.

Strikt zu beachten ist, dass eine Flächenteilung niemals durch die Wort- und Bildmarke verlaufen darf.

Horizontale Winkel

Die horizontalen Flächen werden am oberen Blattrand als eine Kopffläche positioniert. Der untere Rand der Kopffläche steigt von links nach rechts an. Die Steigung (Winkel) ergibt sich aus einem festgelegten Abstand von der Blattoberkante.

In die breite Kopffläche können markante Texte oder Bildmotive integriert werden.

Angaben in mm

Format DIN A4 und DIN A5 breiter Winkel

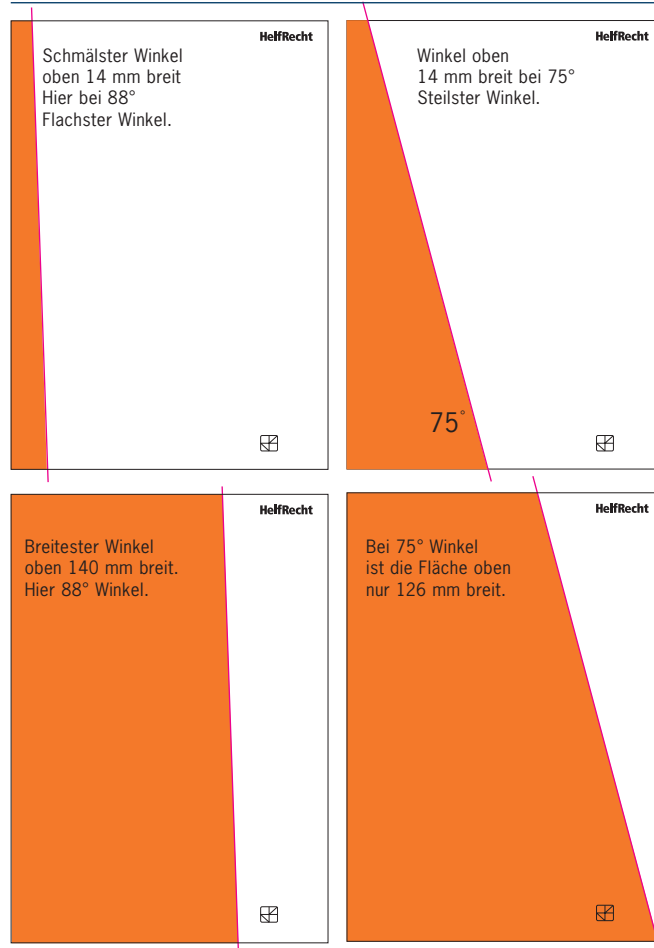
Format DIN A4 und DIN A5 schmaler Winkel

Minimale und maximale Breite der Winkelfläche oben.

Die obere Fläche hat eine minimum und eine maximum Breite bezogen auf die Breite des Formates.
 Die Strecke oben sollte idealerweise durch 7 teilbar sein.

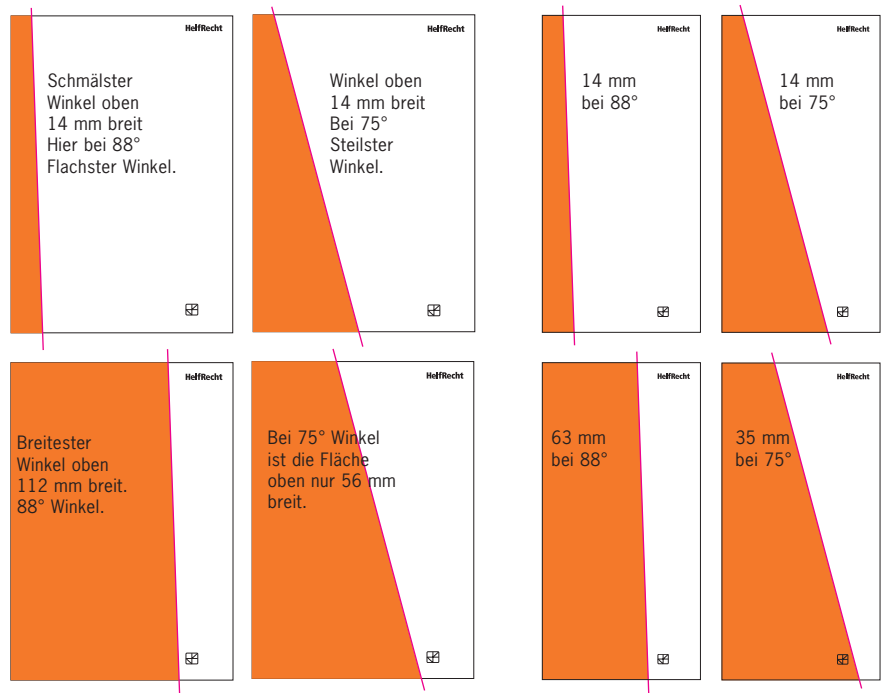
Die Winkel hier zeigen die extremsten Neigungen auf, sie bewegen sich zwischen 75° und 88°.

Format DIN A4



Format DIN A5 148 x 210

Format DIN lang 105 x 210



**Titelseite
mit 3 und 4 Flächen**

Bei zwei Winkeln (= 3 Flächen)
können beide Flächen auch
den gleichen Neigungsgrad haben.
Eine Fläche soll jedoch dabei
dominieren.

Bei der Einteilung der Seite
in 4 Flächen ist der 3. Winkel
immer ein 90 Grad Winkel.
Die Flächen sollen in einem
spannungsreichen Verhältnis
zueinander stehen.



Für alle Printprodukte von HelfRecht wird die Schrift **Syntax und Syntax Serif** (von Hans Eduard Meier) im Originalschnitt von Linotyp verwendet.

Die Verwendung unterschiedlicher Schriftschnitte und Schriftgrößen ergibt differenziertes, klares und leichtlesbares Schriftbild.

Für die Bearbeitung von Drucksachen am PC werden folgende Schriftschnitte eingesetzt: Syntax roman, bold und black Syntax Serif roman und bold

Die Syntax „Kursiv stellen“ sollte nur für den internen Hausgebrauch angewendet werden. Kursiv stellen entstellt die Schrift, sie entspricht nicht dem Originalschnitt.

Syntax light
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax roman
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax medium
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax bold
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax black
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax italic
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

Syntax Serif

Die Syntax Serif dient als ergänzende Schrift für Editorial, Epilog, Zitate sowie im Schriftwechsel als Korrespondenzschrift. Sie wird nicht für Titel, Zwischentitel, Lauftext und Bildunterschriften verwendet.

SyntaxSerif regular
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

SyntaxSerif bold
abcdefghijklmnopqr
stuvwxyz
ABCDEFGHIJK
LMNOPQRSTU
VWXYZ
0123456789

0123456789
Schrift

Headline

Die Schriftgröße der Headlines richtet sich nach dem Inhalt bzw. nach der längsten Headline innerhalb eines Mediums.

Sie gibt die Headlinegröße vor. Alle Headlines und Subheadlines innerhalb eines Mediums sind immer gleich groß.

Von 70 bis 42 Punkt können beide Schriftschnitte verwendet werden. Damit die Headline markant und deutlich wirkt, kommt der Schriftschnitt roman nur dann zum Einsatz wenn genügend Weißraum/Abstand zum Fließtext zur Verfügung steht.

Die Syntax roman oder bold 70 Punkt, ZAB 74 Punkt, LW 0

Head 1
Head 1b

Die Syntax roman oder bold 63 Punkt, ZAB 67 Punkt, LW 0

Head 2
Head 2b

Die Syntax roman oder bold 56 Punkt, ZAB 60 Punkt, LW 0

Head 3
Head 3b

Die Syntax roman oder bold 49 Punkt, ZAB 53 Punkt, LW 0

Head 4
Head 4b

Die Syntax bold 42 Punkt, ZAB 46 Punkt, LW 0

Head 5
Head 5b

Headline 2

als kleinste Headline wird die
Syntax bold in 14 Punkt verwendet

Headlines
sind immer in gemischter
Schreibweise.
Nie in VERSALIEN.

Die Syntax bold 35 Punkt, ZAB 39 Punkt, LW 0

Head 6b
zweite Zeile

Die Syntax bold 28 Punkt, ZAB 32 Punkt, LW 0

Head 7b
zweite Zeile

Die Syntax bold 21 Punkt, ZAB 25 Punkt, LW 0

Head 8b
zweite Zeile

Die Syntax bold 14 Punkt, ZAB 18 Punkt, LW 0

Head 9b
zweite Zeile

Die Syntax bold 12 Punkt, ZAB 12 Punkt, LW 0, zum Beispiel für Stellenanzeigen, Mailing

Head 10b
zweite Zeile

Topline

Die Toplines für die Schriftgrößen: 6, 7 und 8 sind aus der Syntax bold Schriftgröße identisch mit der Fließtextgröße.

Zeilenabstand siehe Headlinegrößen.
Die Topline ist immer in Schwarz

Die Syntax bold 35 Punkt, ZAB 39 Punkt, LW 0

Kleine Topline zur Überschrift

Headline 6B

Kleine Topline zur Überschrift

Hier steht Headline 7B

Kleine Topline zur Überschrift

Hier steht Headline 8B

Themenüberschriften

Eine weitere Möglichkeit um Headlines zu ergänzen sind Themenüberschriften. Sie werden in der gleichen Größe wie die Headlines gesetzt und durch Rasterung (56%) zurückgenommen.

Übergeordnete Themen in der gleichen Größe und in 56% Schwarz

Thema 8B 56%

Hier steht Headline 8B

Schlagwörter

Einzelne Schlagwörter können in der Syntax black plus Akzentfarbe hervorgehoben werden.

Schlagwörter in Farbe

Neu

Aktuell

35%

Mengensatz, Fließtexte

Der linksbündige Flattersatz ist Standard für alle Texte.

Mehr als 3 Trennungen nacheinander sollten vermieden werden. Der Fließtext wird in der Syntax roman gesetzt.

Fließtext-Größen für PC

9 Punkt, Zeilenabstand von 11 Punkt alternativ 10 Punkt mit Zeilenabstand von 13 Punkt

Fließtext-Größen für professionellen Satz

9 Punkt, Zeilenabstand von 4 mm, Laufweite +4
alternativ 9,5 Punkt oder 10,25 Punkt mit Zeilenabstand von 4,5 mm

Bei kurzen Texten wie zum Beispiel beim Katalog oder bei kleineren Formaten (105 x 210) wird ein Zeilenabstand von 3,5 mm für 9 Punkt Schriftgröße verwendet.

Der Blocksatz ist dann anwendbar wenn ein erheblicher Anteil der Seiten Text dominant ist. Wie zum Beispiel bei Büchern oder Periodika. Wird in einem Medium der Fließtext als Blocksatz angelegt ist das im ganzen Medium durchzuhalten.

Professioneller Blocksatz

Blocksatz in der Stilvorlage bearbeiten. Kein erzwungener Blocksatz anwenden. Erzwungener Blocksatz ergibt unschöne „löchrige“ und schwer lesbare Textfelder.

Ausnahme:

Der Blocksatz wird durch Flattersatzpassagen unterbrochen werden, wenn mit diesem Wechsel inhaltliche Unterschiede hervorgehoben werden sollen, zum Beispiel bei Aufzählungen.

Linksbündiger Flattersatz

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild HelfRecht praxisgetreu vorzuführen. Dieser Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild HelfRecht praxisgetreu vorzu

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild HelfRecht praxisgetreu

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild HelfRecht praxisgetreu

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Hier 3,5 mm Zeilenabstand.

Blocksatz mit Textpassagen für Aufzählungen in Flattersatz

Dieser Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout, er dient dazu, die Typografie, das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Dieser Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem Layout, er dient die Typo.

Zwischentitel steht hier in bold gleiche Größe mit einer Zeile Abstand zum Vortext

1. Er hat keine Beziehung zu dem Layout
2. Typografie und das Erscheinungsbild praxis getreu
3. vorzuführen. Dieser Text steht anstelle des
4. Inhaltes. Er hat keine Beziehung zum Blindtext vorliegenden Layout, sondern dient dazu.

Dieser Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout, er dient dazu, die Typografie, das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Der Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem .

Vorspanntexte

Der Vorspann ist immer linksbündiger Flattersatz und wird durch den Schriftschnitt bold hervorgehoben. Die Schriftgröße ist identisch mit Lauftext.

Professioneller Satz

Die Laufweite ist in den Bold-Schnitten immer eine Einheit plus: Bei 9 Punkt also LW +5.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Dieser Text steht anstelle des Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout, sondern

Zwischentitel und Leerzeilen

Zwischentitel sind in der gleichen Schriftgröße wie der Lauftext und wie der Vorspann in Syntax bold. Sie stehen mit einer Leerzeile zur letzten Zeile. Zwischenüberschriften sind immer linksbündig (auch wenn der Text, der folgt Blocksatz ist). Die Trennung der Textblöcke erfolgt immer durch ganze Leerzeilen, eine oder auch zwei. Bei halben Zeilen oder X-Abständen halten die Textblöcke nicht mehr Register und es entsteht ein unruhiges und unstrukturiertes Lesebild

Der Zwischentitel kann auch in der dominierenden Akzentfarbe des Mediums stehen.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Dies ist ein Zwischentitel

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient ist dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Zwischentitel in der dominierenden Farbe des Mediums

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das ErscheiDieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das ErscheiDieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erschei

Die Bildunterschrift ist 1 Punkt kleiner als der Fließtext, aber im gleichen Zeilenabstand wie der Fließtext. Sie kann in der dominierenden Akzentfarbe des Mediums stehen oder in Schwarz. Die Spalte ist schmaler als die Fließtextspalte (sie liegt innerhalb des Rasters, siehe Satzspiegel.

Marginaltexte, Legenden, Fußzeilen sind in 7 oder 6 Punkt

Bildunterschriften, Marginaltexte (Quellenangabe), Fußzeilen, Grafiken, Stellenanzeigen

Dieser Text steht anstelle des eigentlichen Textes. Stellvertretend für die Bildunterschrift Dieser Text anstelle des eigentlichen Text 8 Punkt mit Zeilenabstand 4 mm. Für PC: 11 Punkt.

Für kleine Stellenanzeigen wird die Syntax roman und Syntax bold in 8 Punkt mit Zeilenabstand 9 Punkt eingesetzt. Wenn möglich mit Laufweite 4

Dieser Text steht anstelle des eigentlichen Textes. Stellvertretend für Fußzeilen, Grafiken einen Hinweis (Querverweis oder Quellenangabe) Marginaltexte. Legenden und Beschreibungen im Katalog. **Bei Hervorhebungen wird die Syntax bold verwendet. Marginaltexte. Legenden und Beschreibungen im Katalog.**

Schriftgröße hier 7 Punkt mit Zeilenabstand 3,0625 mm Laufweite + 4

Zeilenabstand PC 8 Punkt

Dieser Text steht anstelle des eigentlichen Textes. Stellvertretend für Fußzeilen, Grafiken einen Hinweis (Querverweis oder Quellenangabe) Marginaltexte. Dieser Text steht anstelle des eigentlichen Textes. Stellvertretend für Fußzeilen und einen Hinweis (Querverweis oder Quellenangabe) Marginaltexte oder bei Bedarf auch für komplexe Grafiken.

Schriftgröße hier 6 Punkt mit Zeilenabstand 2,4 mm Laufweite + 4

Zeilenabstand PC 7 Punkt

Zitate

Zitate werden in der Syntax Serif italic gesetzt.

Die Schriftgröße, Zeilenabstand und Laufweite ist identisch mit der Fließtextgröße.

Der Name des Zitierten steht in Syntax roman 2 Punkt kleiner als das Zitat.

Das Zitat kann auch in der dominierenden Akzentfarbe des Mediums stehen. Der Zitierte steht jedoch in Schwarz.

Syntax Serif italic (kursiv)

„Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.“

Vorname Nachname, Position, Jahr

„Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.“

Vorname Nachname, Position, Jahr

Syntax Serif roman und bold

Die Syntax Serif dient ergänzend als Schrifttyp für Editorial, Epilog sowie als Korrespondenzschrift. Auszeichnungen in der Syntax Serif bold.

Fließtext-Größen für PC

9 Punkt, Zeilenabstand von 11 Punkt alternativ 10 Punkt mit Zeilenabstand von 13 Punkt verwendet.

Fließtext-Größen für professionellen Satz

9 Punkt, Zeilenabstand von 4 mm, Laufweite +5
alternativ 9,5 Punkt mit Zeilenabstand von 4,5 mm, Laufweite +5

Die Syntax Serif wird nicht für Fließtexte, Headline, Legenden und Produkttexte verwendet.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Dieser Text steht anstelle des eigentlichen Inhaltes

Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Zeichen für Aufzählung, Stichpunkte, Wahlmöglichkeiten, Hinweise

Listen

Die Quadrate für die Aufzählung sind aus der Schrift EuropeanPi 3 Die Größe der Quadrate ist 3 Punkt kleiner als der Fließtext. Bei einer Schriftgröße von 10 Punkt sind die Aufzählungsquadrate 7 Punkt (Beispieltext) Der Abstand der Quadrate zum Text ist ein Wortabstand (Leerzeichen) Bei einer Schriftgröße von 10 Punkt ist das 3,5 mm.

Quadrate zum Ankreuzen sind in Outline (ohne Schatten), aus der Schrift EuropeanPi 3 und in der gleichen Größe wie der Fließtext. Der Abstand zum Text ist ein Wortabstand

Pfeile streng und zweckmäßig. Schrift: Zapf Dingbats, Größe identisch mit Fließtext.

Zeichen um z.B. auf Informationen hinzuweisen. In der Syntax bold. Gleiche Schriftgröße wie der Text.

HelfRecht Produktbeschreibungen sind tabellarisch gesetzt. Trennlinien (Stärke 0,25 mm) zwischen den Textzeilen erleichtern das Zuordnen der Information. Schriftgröße (für Katalog): 8 Punkt mit Zeilenabstand 3,5 mm und 7 Punkt mit Zeilenabstand 3,0625 mm

- Aufzählung, Listen immer flächige Quadrate, keine Kontur, Schatten oder Kreise.
- Abstand zum Aufzählungspunkt ist 1 Wortabstand (Leerzeichen)
- Größe des Aufzählungspunktes ist 3 Punkt kleiner als die Schriftgröße des Textes. Das Quadrat ist aus der Schrift EuropeanPi 3
- Sie sind entweder in schwarz oder in der Akzentfarbe des Mediums.
- Aufzählungen können mit ganzen Leerzeilen getrennt werden oder wenn viele Positionen auch ohne Leerzeilen. Halbe Zeilen sollten vermieden werden. Die Texte halten nicht mehr Register (Versatz) und wirken unordentlich.
- Die Unterpunkte einer Aufzählung werden in 30% gerastert.
- Aufzählungen sind immer im Flattersatz, da die einzelnen Punkte bei Aufzählungen immer unterschiedlich lang sind, wird der Blocksatz notgedrungen löchrig und nicht lesefreundlich.

- bitte hier ankreuzen
- bitte kreuzen sie hier an
- kann auch in 2. Farbe des Medium stehen.

- siehe Seite 100
- siehe Seite 100

- > Zeichen weist auf wichtige Inhalte hin
- > Lassen Sie Ihrem Unterbewusstsein Zeit.

Neuheiten 9 Punkt

	Lederringbuch „Classic Gold“
8503	schwarz mit Goldprägung (Logo)
7503	bordeaux mit Goldprägung (Logo)
9145	Planereinlagen: FrühjahrsSet
5907	Multifunktionsstift: Diplomat Visa Data 2

Lederringbuch »Private«

(ohne Inhalt)	
naturgeschrumpftes Rindleder	
8-mm-Ringmechanik	
Außenmaße 168 x 106 mm	
zwei Lederinnentaschen, zweiseitig offen	
auswechselbare Stiftschlaufe	
(Ø 8 mm, wahlweise Ø 6 mm) → Seite 30	
Initialen- und Namensprägung möglich	
integrierbar in die Office-Mappen	

Artikel-Nr. 2255	schwarz	39,50
------------------	---------	-------

HelfRecht

Typografie: So nicht.

Diese Satzformen und Schriftbilder kommen bei HelfRecht nicht zur Anwendung.

Kein zentrierter Satz (Mittelachse)

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.

Formsatz dient nicht dem Inhalt im Sinne von HelfRecht.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen. Formsatz dient nicht dem Inhalt.

Keine Sperrungen bei Textzeilen. Auch keine einzelnen Worte innerhalb eines Textes sperren.

Dieser Text steht anstelle des eigentlichen Inhaltes. Dieser Text steht anstelle Inhaltes. Er hat keine Beziehung zu de

Keine Unterstreichungen weder im Fließtext noch bei anderen Texttypen.

Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zu dem vorliegenden Layout,

Keine Versalien zur Auszeichnung. Texte in Großbuchstaben sind schwer lesbar.

DIESER TEXT STEHT ANSTELLE DES EIGENTLICHEN INHALTES. DIESER TEXT STEHT ANSTELLE DES IST EIGENTLICHEN INHALTES. DIESER TEXT STEHT UND ANSTELLE DES EIGENTLICHEN INHALTES.

Outlineschriften, Schatten, Verzerrungen in alle Richtungen.

Dieser Text steht anstelle

Dieser Text steht anstelle

Dieser Text steht **anstelle**



Formate Abb. in 30%

Die HelfRecht Druckformate sind DIN Formate.
Ausnahme die Hauszeitschrift „Methodik“ sie ist im Format: 210 x 285 mm.

54 x 84 mm

Kleinstes Format
z.B. für
Visitenkarten

70 x 105 mm

Taschen-Format
z.B. für
schnelle Info

105 x 148 mm

Postkarten-Format
z.B. für
Karten, Mailings,
Aktionen, etc..

oder als
Querformat (148 x 105)

105 x 105 mm

Quadratisches
Format z.B. für
Karten, Mailingaktionen,
Sonderthemen.

105 x 210 mm

DIN-Lang Format
z.B. für
Faltblätter, Flyer,
Einladungen, Grußkarten
etc..

oder als
Querformat (148 x 105)
für
Kurzmitteilungen,
Compliment-Cards...

148 x 210 mm

Format z.B. für
Katalog,
HelfRecht Bibliothek

210 x 297 mm

Format für Broschüren

Logogröße auf Titelseiten

Wortmarke und Bildzeichen auf Titelseiten

Das Logo von HelfRecht besteht aus der Wortmarke und dem Bildzeichen. Die Wortmarke kann, wenn die Umstände Einschränkungen verlangen, ohne Bildzeichen eingesetzt werden.

Der Einsatz des Bildzeichens ohne Wortmarke ist problematisch, da das Zeichen – anders als beispielsweise der Stern von Mercedes – vielen potenziellen Kunden nicht geläufig ist. Die isolierte Anwendung ist deshalb nur dann möglich, wenn:

entweder das Umfeld klar „HelfRecht“ signalisiert; oder das mit dem Zeichen versehene Objekt nur für Personen gedacht ist, die das Bildzeichen mit HelfRecht assoziieren.

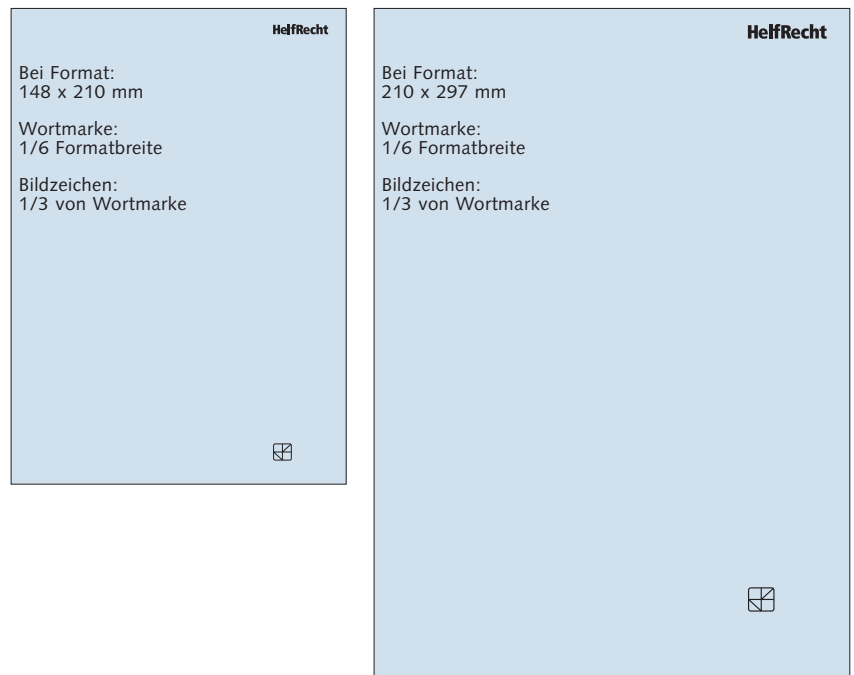
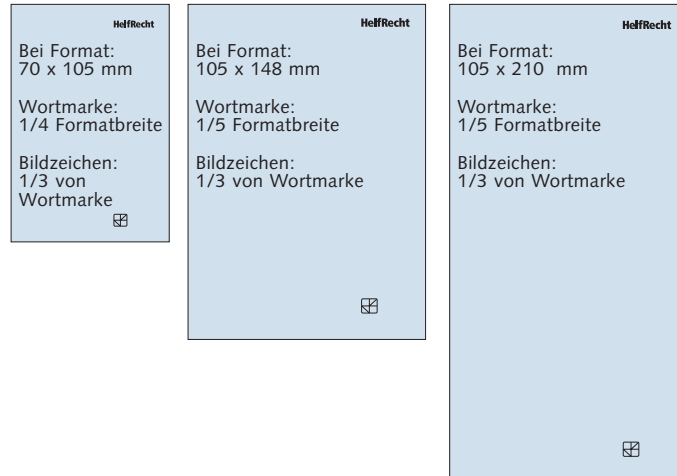
Wortmarke zu Bildzeichen

Die Breite des Bildzeichens beträgt ein Drittel der Länge der Wortmarke.

Die Bildmarke wird unterhalb der Wortmarke und linksbündig auf Achse angeordnet (Details siehe Satzspiegel).

Wortmarke zum Flächenformat

Die Größe der HelfRecht Wortmarke ist abhängig von der Breite der Fläche, auf der sie plziert wird – bei schmalen Formaten ist die Wortmarke proportional breiter. Dadurch wird für eine prägnante Erkennbarkeit / Lesbarkeit gesorgt.



Satzspiegel für Seitenformat 105 x 210 mm, Titelseite, Abb. in 100%

Die Gestaltung der Printmedien von HelfRecht basiert auf einem Raster-System.

Die Maße der Ränder: Kopfsteg, Fußsteg, Randsteg und Bundsteg beziehen sich wenn möglich auf die Zahl 7:

Sie haben das Maß von 7 mm, dessen Mehrfaches (z.B. 2 x 7 mm = 14 mm) bzw. dessen Hälfte (3,5 mm) oder dessen Viertel (1,75 mm).

Die Einteilung des Rasters erfolgt horizontal und vertikal. Entsprechend der Breite des Formates wird, der Raster in 3, 6 (siehe Abb.), 12 oder 13 horizontale Einheiten geteilt (bei viel- und kleinteiligen Information, wie z.B. beim Katalog).

Der Zeilenabstand richtet sich nach der Schriftgröße. Zum Beispiel beträgt er hier 3,5 mm bei einer Schriftgröße bis max. 9 Punkt.


Die Wortmarke steht auf Titelseiten oben – rechtsbündig – im Raster. Die Bildmarke steht linksbündig in der horizontalen Achse zur Wortmarke auf der Basislinie des Rasters.

Die Größe des Logos

Beim Papierformat 105 x 210 mm ist die Wortmarke 21 mm breit. Die Breite des Bildzeichens beträgt 7mm.

Die Größe des Logos

Die Bildformate nehmen ebenfalls das Rastersystem auf.

							HelfRecht		7
1	2	3	4	5	6				
							Zeilenabstand bei einer Schriftgröße von max. 9 Punkt ist 3,5mm		
							Die Schrift steht auf Zeile und hält Register.		
							Hier steht ein Zwischentitel auf Zeile		
							hier steht der Lauffexthier steht der Lauffexthier steht der Lauffexthier steht der Lauffext		
schmalste Einheit	Bildformate richten sich nach dem Zeilenabstand und sind mit minimum einer Leerzeile getrennt								
							usw.		
							3,5 mm		
							3,5 mm		
									
							Achse		14
7	12,25	3,5	und so weiter						7

HelfRecht Raster, Satzspiegel DIN A5

Bei Rubriken etc., die oben plaziert sind, ist der obere Abstand zum Papierrand größer als der Abstand unten zum Satzspiegelrand. Entsprechend steht die Wortmarke außerhalb des Satzspiegels.


Satzspiegel für Seitenformat 148 x 210 mm, Abb. in 100%, rechte Seite

10,5			Neuheiten													
		<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 10%;">1</td><td style="width: 10%;">2</td><td style="width: 10%;">3</td><td style="width: 10%;">4</td><td style="width: 10%;">5</td><td style="width: 10%;">6</td><td style="width: 10%;">7</td><td style="width: 10%;">8</td><td style="width: 10%;">9</td><td style="width: 10%;">10</td><td style="width: 10%;">11</td><td style="width: 10%;">12</td><td style="width: 10%;">13</td> </tr> </table>	1	2	3	4	5	6	7	8	9	10	11	12	13	
1	2	3	4	5	6	7	8	9	10	11	12	13				
		<p>HelfRecht hat den Planer zwar nicht völlig neu erfunden – ein bisschen aber schon! Denn das neue Planerkonzept »7·24« bietet eine einzigartige Innovation: die Ringmechanik zum Wechseln.</p> <p>Der besondere Vorteil: Sie können den kompletten Planerinhalt inklusive der Ringmechanik mit einem Handgriff in eine andere Planerhülle einsetzen, können also Ihrem Planer ruck, zuck ein neues »Outfit« verpassen.</p> <p>→ Seiten 14 bis 19</p> <p>Zeilenabstand bei 9 Punkt 3,5 mm</p>	<p>Nutzen Sie von Anfang an die Wechselmöglichkeit und sparen Sie 10 Prozent.</p> <p>Damit Sie den besonderen Vorteil des Planersystems »7·24« von Anfang an optimal nutzen können, empfehlen wir Ihnen die Kombination von Business- und Outdoor-Planerhülle inklusive</p> <p>Zeilenabstand bei 7 Punkt jede 5. Zeile steht im Raster Zeilenabstand 3,0625 mm (=8,681 Punkt)</p>													
<p>1,75 1,75 1,75 1,75 1,75 1,75 1,75 1,75</p>	<p>8</p> <p>7</p> <p>3,5</p> <p>7</p> <p>3,5</p>	<p>und so weiter</p>	<p>4</p> <p>7</p>													

Seitenformat 210 x 297 mm, Titelseite mit Headline im Winkel Abb. in 70%



Doppelseite offenes Format 420 x 297 mm, Abb. in 40%



Sehr geehrte Damen und Herren,
 Unsere praxisnahe Führung hat viele Facetten aber ganz klare praktische Grundlagen. Diese Basis erarbeiten Sie bei unseren Planungstagen für Unternehmensführung.

In diesen Planungstagen treffen Sie als Unternehmer auf Unternehmer. Wir reden nämlich nicht – wir es wird tun – von der Theorie, sondern führen tatsächlich wie ein Unternehmen. Aus dieser Praxiserfahrung haben wir konkrete Methoden und Systeme zur Unternehmensführung ab, um die höchsten Unternehmenswerte zu machen. Und das bereits seit über 30 Jahren.

Im HelfRecht-Managementsystem finden Sie pragmatische Instrumente, die bei jeder Phase der kontinuierlichen Innovation, des permanenten Qualitätsmanagements oder des wirkungsvollen, durchgängigen Controlling ermöglichen. Solche Führungsinstrumente haben wir in eigener Regie entwickelt und in ein ganzheitliches System integriert – lange bevor sie mit irgendwelchen Begriffen als Sensation durch die Wirtschaftskreise gingen.

Wir glauben aber nicht an die Wundervirkung einzelner Werkzeuge. Einen höchlichster Nutzen entfalten diese doch nur, wenn sie eingebunden sind in ein unternehmerisches Konzept, das auf den Menschen abgestimmt ist. Denn der Mensch ist der Bewegende aller wirtschaftlichen Prozesse.

Dieses unternehmerische Konzept entwickeln Sie während der Planungstage – eine individuelle Strategie für Ihr Unternehmen sowie ganz konkrete Hilfe für Problemstellungen und Dilemmata. Was Sie ebenfalls mit nach Hause nehmen ist unser Versprechen, künftig jederzeit für Sie da zu sein, wenn Sie auf ihrem Erfolgsweg praktischen Rat zur methodischen Umsetzung brauchen.

HelfRecht und seine Kunden

Ihr
 Werner Razer
 Vorstand HelfRecht AG

HelfRecht ist der Partner in der Fortentwicklung und Vermittlung moderner Führungsinstrumente für Unternehmer.

- Seit über 30 Jahren sind wir in diesem Bereich etabliert.
- Mehr als 20.000 Führungspersonalitäten haben bereits an unseren Planungstagen teilgenommen.
- Die meisten kommen wieder.
- Empfehlung begeisteter Kunden zu uns.
- Viele besuchen die Planungstage immer wieder.

Ihre HelfRecht-Planungstage für Unternehmensführung

Systematisch zu 8B hoher Führungssicherheit

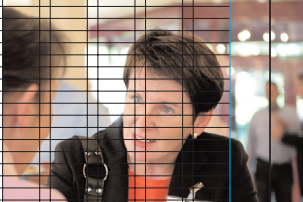
Unternehmer: Die Welt bewegen
 Sie wollen etwas bewegen, statt zu warten, bis das Glück auf Sie zukommt. Sie haben Bedenken vor. Sie wollen Ihr Unternehmen für die Zukunft ausrichten, wollen Wachstum erreichen oder eine Krise meistern.

Willkommen bei HelfRecht! Was Sie beschäftigt hat uns verblüht. Für solche Aufgaben sind wir mit den Planungstagen für Unternehmensführung ein befragter Partner mit großer Erfahrung.

Vier konzentrierte Tage. Ein guter Start.
 Das Tagesgeschäft lässt Ihnen nur selten Zeit und Ruhe für kreative, konzentrierte Zukunftsgestaltung. Permanenz werden Sie gebracht und beibringt, möchte jemand Ihre Entscheidung, Ihren Kommentar, Ihre Fallnahmen.

Systematisch entscheidende Fortschritte erzielen
 Die HelfRecht-Planungstage bringen Ihrem Unternehmen in vielfältiger Weise bedeutende Fortschritte, in welchen Bereichen sehen Sie aktuell den größten Bedarf an positiven Veränderungen?

Das Prinzip
 1. Ich will das gesamte Unternehmen flexibler und besser steuerbar machen, damit wie Ziele schneller entstehen und Marktreaktionen effizienter umgesetzt werden können.
 2. Ich will meine Mitarbeiter motivieren und fördern und ihnen die besten Arbeitsbedingungen schaffen, damit sie noch produktiver, doch eigenverantwortlicher und auch initiativere sind.
 3. Ich will die Effektivität der Strukturen und Prozesse im Unternehmen bestmöglich verbessern und damit die Qualität und den Nutzen für unsere Kunden steigern.
 4. Ich will die Leistungen von Marketing und Vertrieb optimieren sowie die Produktivität erheblich steigern.
 5. Damit mein Unternehmen möglichst im Markt beherrschend beziehungsweise neue Märkte erschließt.
 6. Ich will die Solidität der Finanzierung erhöhen.
 7. Ich will die Rentabilität nachhaltig steigern, die Liquidität verbessern und somit die Kreditwürdigkeit verbessern.
 8. Ich will mein Unternehmen und die Produkte besonders vorbildlich auf besonderen Kundenwünschen ausrichten.
 9. Ich will die Innovationsfähigkeit und die Innovationsaktivitäten des Unternehmens erheblich erhöhen.
 10. Ich will unseren Kunden langfristige, persönliche Beziehungen und dabei nach besonders hoher Loyalität bei Mitarbeitern, Geschäftspartnern und Zulieferern.
 11. Ich will mein Unternehmen und die Produkte besonders individuell und eigenständig machen, so dass sie aus Sicht unserer Kunden unverwundbar und nicht austauschbar sind.
 12. Ich will für mich ausreichende Freizeitmöglichkeiten für unternehmerische Entscheidungen schaffen.



Seitenformat 210 x 280 mm, Schrift im Raster, Abb. in 70%

<p>Hier steht die Topline in 9 Punkt bold</p>	
<p>Hier steht die Headline 8B in 28 Punkt, ZAB 34 Punkt</p>	
<p>Zwischentitel in 9 Punkt bold Sie wollen etwas bewegen, statt zu warten, bis das Glück auf Sie zukommt. Sie haben Bedeutendes vor: Sie wollen Ihr Unternehmen für die Zukunft ausrichten, wollen Wachstum erreichen oder eine Krise meistern.</p>	<p>Bilder stehen stehen auf Zeile. Wenn Bilder neben Text stehen, bestimmt die Versalhöhe der Schrift die obere Kante</p>
<p>Willkommen bei HelfRecht! Was Sie beschäftigt, ist uns vertraut. Für solche Aufgaben sind wir mit den Planungstagen für Unternehmensführung ein gefragter Partner mit großer Erfahrung.</p>	
<p>Schrift steht auf Zeile Das Tagesgeschäft lässt Ihnen nur selten Zeit und Ruhe für kreative, konzentrierte Zukunftsgestaltung. Permanent werden Sie gebraucht und gefordert, möchte jemand Ihre Entscheidung, Ihren Kommentar, Ihre Einflussnahme. Wenn Sie die treibende Kraft und nicht der Getriebene sein wollen, hilft nur eines: Klinken Sie sich aus – um die wichtigsten unternehmerischen Fragen zu überdenken und alle Schalter auf „Erfolg“ zu stellen.</p>	<p>Zwischentitel in 9 Punkt bold Sie wollen etwas bewegen, statt zu warten, bis das Glück auf Sie zukommt. Sie haben Bedeutendes vor: Sie wollen Ihr Unternehmen für die Zukunft ausrichten, wollen erreichen oder eine Krise meistern. Dies ist ein Blindtext, ersticht für den eigentlichen text. Die Spalten halten Register in den Zeilen.</p>
<p>Wohl kaum etwas anderes ist hierfür besser geeignet als die Planungstage bei HelfRecht. Wir versprechen Ihnen, dass Sie der Rahmen beflügeln wird, unerwartet große Lösungspakete für Ihr Unternehmen zu schnüren. Die Dichte Ihrer Leistung in diesen vier kreativen Tagen wird Sie selbst überraschen – und Ihr Unternehmen machtvoll bewegen. Vier Tage. Ein erstaunlicher Anfang für eine packende Entwicklung.</p>	<p>Hier steht eine Legende Jede 4. Zeile steht im Zeilenraster Hier steht eine Legende Jede 4. Zeile steht im Zeilenraster Hier steht eine Legende Jede 4. Zeile steht im Zeilenraster Hier steht eine Legende Jede 4. Zeile steht im Zeilenraster Hier steht eine Legende Jede 4. Zeile steht im Zeilenraster</p>

Die Pagina steht außerhalb des Satzspiegels, rechtsbündig mit mindestens einer Zeile Abstand.

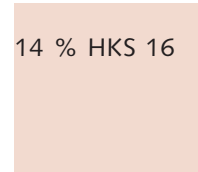
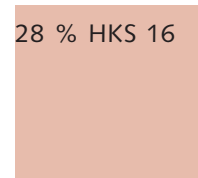
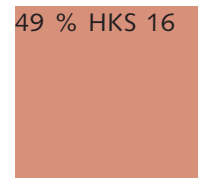
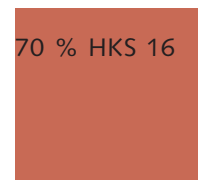
Das Farbsystem von HelfRecht besteht aus zwei Identitätsfarben und einer Akzentfarben-Palette.

Die Identitätsfarben

Die Identitätsfarben tragen maßgeblich zur visuellen Individualität und zur spontanen Wiedererkennbarkeit von HelfRecht bei.

Wenn der Einsatz von Farben eingeschränkt ist, hat die Primärfarbe Vorrang. Die Sekundärfarbe hat ihre eigentliche Funktion in der Erzeugung eines Kontrasts, der die Ausstrahlung der Primärfarbe eminent erhöht.

Primäre Identitätsfarbe



Primäre Identitätsfarbe von HelfRecht ist ein dunkles, warmes Rot

Gestrichenes Papier:

Schmuckfarbe: HKS 16 K

Prozessfarbe:

30% cyan, 100% magenta, 100% gelb

Ungestrichenes Papier

Schmuckfarbe: HKS 16 N

Prozessfarbe:

10% cyan, 100% magenta, 100% gelb

Kunststoffe, Lacke, etc..

RAL 3003 Rubinrot

Bildschirm

R 158, G 23, B 26

80% R, 0% G, 20% B

Hex # (Web): 9E171A

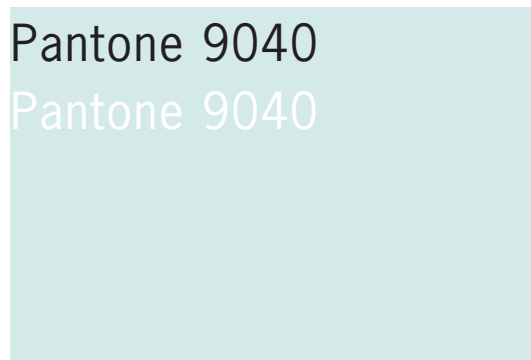
Die Rastertöne sind zu verwenden bei Flächen und Diagrammen.

Sekundäre Identitätsfarbe

Die sekundäre Identitätsfarbe

Pantone 9040 wird hauptsächlich in der Geschäftsausstattung in den horizontalen Winkelflächen eingesetzt.

Größere Flächen (z.B. innerhalb einer Broschüre) können ebenfalls mit diesem Ton gestaltet werden.



Gestrichenes Papier:

Schmuckfarbe: Pantone 9040 U

Prozessfarbe:

10% cyan, 5% gelb

Ungestrichenes Papier

Schmuckfarbe: Pantone 9040 U

Prozessfarbe:

8% cyan, 4% gelb

Bildschirm

R 228, G 239, B 238

85% R, 93% G, 88% B

Hex (Web): E4EFEE

Die hier abgebildeten Farben sind in der Bezeichnung verbindlich, jedoch nicht von der Bildschirmdarstellung und dem Farbausdruck.

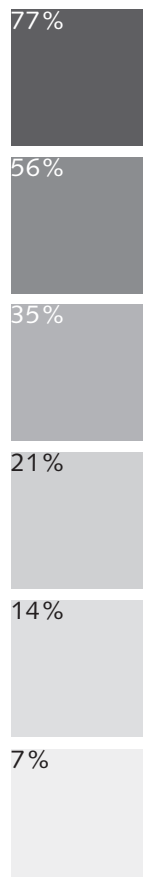
Die Akzentfarben dienen als Mittel zur ästhetischen Aufwertung sowie als Strukturgeber und Zuordnungshilfen (z.B. Kapitel im Katalog).

Da sie für die Identität von HelfRecht weniger relevant sind, können Veränderungen der Palette in kürzeren Intervallen vorgenommen werden.

Akzentfarben, die CMYK Werte sind abgestimmt für gestrichenes Papier

Pantone 541 CMYK: 100_58_9_50	Pantone 525 CMYK: 74_100_0_0	Pantone 582 CMYK: 25_9_100_44	Pantone 5473 CMYK: 86_20_32_53	Pantone 249 CMYK: 44_95_7_32	Pantone 158 CMYK: 0_65_95_0
Pantone 279 CMYK: 68_34_0_0	Pantone 528 CMYK: 43_58_0_0	Pantone 397 CMYK: 14_2_100_16	Pantone 5483 CMYK: 65_11_25_27	Pantone 246 CMYK: 34_88_0_0	Pantone 144 CMYK: 0_58_100_0
Pantone 536 CMYK: 35_17_2_7	Pantone 530 CMYK: 18_33_0_0	Pantone 396 CMYK: 10_0_95_0	Pantone 5493 CMYK: 48_4_16_15	Pantone 244 CMYK: 9_45_0_0	Pantone 143 CMYK: 0_36_87_0
Pantone 538 CMYK: 14_4_1_3	Pantone 263 CMYK: 10_18_0_0	Pantone 586 CMYK: 10_0_59_0	Pantone 5523 CMYK: 22_1_9_2	Pantone 250 CMYK: 7_28_0_0	Pantone 135 CMYK: 0_23_76_0

mögliche Schwarz-Rasterstufen



Die hier abgebildeten Farben sind in der Bezeichnung verbindlich, jedoch nicht von der Bildschirmdarstellung und dem Farbausdruck.

Bildsprache

Bilder von HelfRecht sind natürlich und realistisch, kraftvoll, warm und aus dem Leben gegriffen, von diskret heiterer Grundstimmung – aber nie sentimental.

Die Bildmotive sind dicht und erzählerisch. Ein unaufdringlicher Symbolismus ist möglich, insofern natürliche Analogien zwischen der Aussage und dem Bildmotiv bestehen. Beispiel: Aussage „Teamleistung“ mit „Achter mit Steuermann“ illustrieren.

Zu beachten ist allerdings, dass der Schritt von einem charmanten Symbolismus zu einem einfallslosen Klischee klein ist („Palmenstrand“ als Zeichen für Entspannung, „Montblanc-Füller“ als Symbol für gewichtige Vereinbarung, etc..).

Auf Verzerrungen, extreme und unrealistische Perspektiven, manieristische Stilmittel (z.B. betonte Weichzeichner) und Verfremdung jeder Art wird verzichtet:

HelfRecht „manipuliert“ grundsätzlich nicht. Was immer man von HelfRecht sieht, erscheint „wahrheitsgetreu“ und wirkt vertrauenserweckend.

Die Fotografie ist bevorzugt farbig, nicht schwarz-weiß. Kräftige Kontraste sind möglich, insofern sie dem Bildthema angemessen sind, grelle – effekthascherische – Farben werden jedoch vermieden.



Bildsprache Produktaufnahmen

Die HelfRecht Produktfotografie zeigt das Produkt so, wie es tatsächlich ist – wohl „von seiner besten Seite“, aber nie mit überhöhenden optischen Tricks versehen.

Die Bilder machen die Produkte verstehbar, indem sie das ganze Objekt in seinen echten Dimensionen und Proportionen erkennbar machen und Details, die bemerkt werden sollen, durch ergänzende Aufnahmen hervorheben.

Das Produkt wird gewissermaßen vor der Kamera gedreht und gewendet, von ihr sinnlich ertastet und in seinen Funktionen nachvollzogen.

In diesem Sinn ist die Produktfotografie von HelfRecht sachlich und informativ, sympathisch und verlässlich.

„Systematik“ ist eine für HelfRecht besonders wichtige Kompetenz. Deshalb werden auch bei der Produktfotografie durchgängige formale Regeln angewandt. Die Draufsicht auf die Produkte soll, wo immer möglich, in einem Winkel von entweder ca. 60° oder ca. 90° (volle Draufsicht) erfolgen. Die Hauptachse des Produkts soll in einem Winkel von ca. 30°, 60° oder 90° zur Grundlinie des Bildformats stehen.

Die Produkte sind im üblichen Fall auf neutralem Hintergrund darzustellen; Freisteller sind zu vermeiden. Für Aufmacherfotos oder um der besseren Verständlichkeit willen, kann das Produkt in seinem Anwendungsumfeld dargestellt werden.

Die Fotografien werden ohne Rahmen abgebildet. Ecken werden grundsätzlich nicht abgerundet.



HelfRecht Anwendungsbeispiele

Mailingbögen Vorderseite Typ A

Die Mailingbögen von HelfRecht sind in zwei Typen unterteilt:

- Typ A ist die Version mit breitem Winkel
- Typ B ist die Version mit dem schmalen Winkel.

Die Winkelfarbe auf der Vorderseite ist immer HKS 16.

Der breite Winkel kann Träger von Information sein in Form von Text oder/und Bild bzw. ein angehängtes Bildmotiv.

Typ A_ DIN A4 Mailing-Bogen Vorderseite mit breitem Winkel_ Anmeldung und Information



Weitere Beispiele und detaillierte Beschriftungsmuster und Vermaßung liegen vor.

Typ A_ DIN A4 Mailing-Briefbogen, Vorderseite, Text mit Unterlängen im breitem Winkel

**Mailingbögen Vorderseite
Typ A**

Wenn der Text im Winkel Unterlängen hat, wird die Schrift in 10% HKS 16N gerastert.

Beflügeln Sie sich...

Anrede
Kundenname

Musterwohn Straße 40/1
80863 München

HelfRecht

 **HelfRecht Unternehmersche
Planungsmethoden AG**
Markgrafenstraße 32
D-95680 Bad Alexandersbad

Telefon +49 (0) 92 32/601 - 0
Fax +49 (0) 92 32/601 - 280
info@helfrecht.de
www.helfrecht.de

Vorstand: Werner Bayer
Vorsitzender des Aufsichtsrats:
Siegfried Stocker
Sitz der Gesellschaft:
Bad Alexandersbad
Amtsgericht Hof, HRB 3205

HelfRecht-Katalog 2008

Sehr geehrter «PAN»,

„Beflügeln Sie sich und Ihr Unternehmen!“ Wir helfen Ihnen dabei!
Mit tollen Neuheiten und mit vieltausendfach bewährten Planungs- und Arbeitsmitteln. Schauen Sie doch gleich mal rein in unseren neuen Katalog 2007/08!

HelfRecht-Software „TarGo“: Unternehmenssteuerung per USB-Stick
Besonders ans Herz legen möchte ich Ihnen „TarGo“, eine neu entwickelte Software zum Planen und kontrollierten Umsetzen von Unternehmenszielen. Sie läuft komplett auf einem USB-Stick, also unabhängig von Ihrem Bürorechner oder Laptop. Auf dem Stick haben Sie alles dabei – und können so auch unterwegs mit jedem Rechner auf Ihre Zielpläne zugreifen. Einfach am USB-Anschluss einstecken und loslegen! Noch eine wahrlich beflügelnde Funktion: Per Mausclick übertragen Sie Ihre Daten als Aufgaben oder Termine in Ihr Outlook. Schnell und bequem. → Seiten 4/5

Seit Jahrzehnten beflügelnd: Planungstage bei HelfRecht
Sie wollen Ihr Unternehmen in die Proflika führen? Ihre Führungskräfte weiterentwickeln? Ihre Mitarbeiter zu eigenverantwortlichem Handeln motivieren? Sie wollen Ihre Selbstorganisation verbessern? Ihr Zeitmanagement optimieren? Ihren Führungsstil verfeinern?

Dann kommen Sie zu unseren Planungstagen: Ganz systematisch entwickeln Sie hier klare Ziele und sehr konkrete, maßgeschneiderte Lösungen für Ihre individuelle Situation. → Seiten 65 bis 69, 72

Gemeinsam mit meinem ganzen Team würde ich mich sehr freuen, wenn wir Sie dabei unterstützen dürfen, sich und Ihr Unternehmen in jeder Hinsicht zu beflügeln!

Sehr herzliche Grüße,
Ihr

Werner Bayer

Datum
11. September 2007
Ihre Servicenummer
«KUNDNR»



**HelfRecht
TarGo**



Unser Geschenk an Sie:
Gerade in der heutigen Zeit ist es unverzichtbar, langfristige Ziele zu haben. Gerne schicken wir Ihnen per E-Mail Anregungen für Ihren persönlichen und unternehmerischen Lebenszielplan zu. Interessiert? Dann senden Sie eine E-Mail (Stichwort Lebenszielplan) an: info@helfrecht.de oder rufen Sie uns an: 00 49 (0) 92 32/ 601 - 251

HelfRecht Anwendungsbeispiele

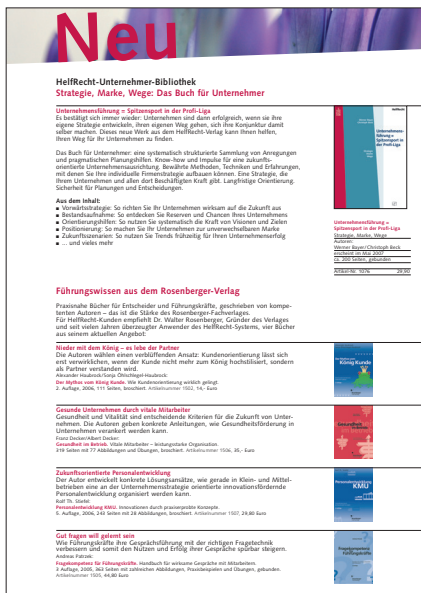
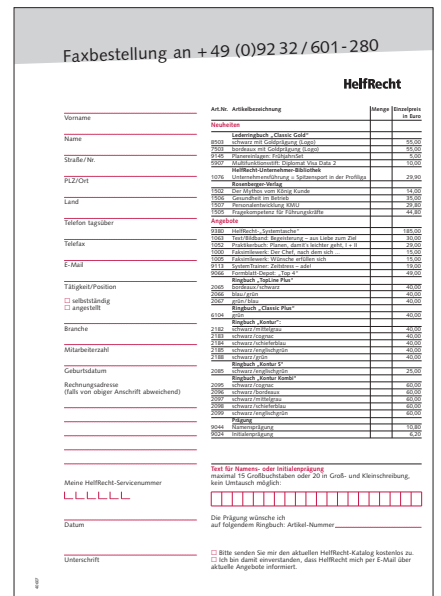
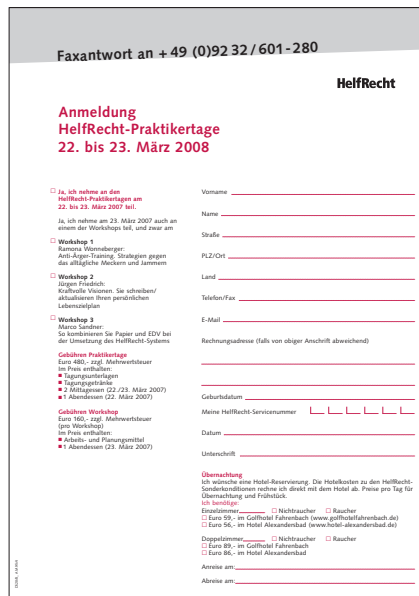
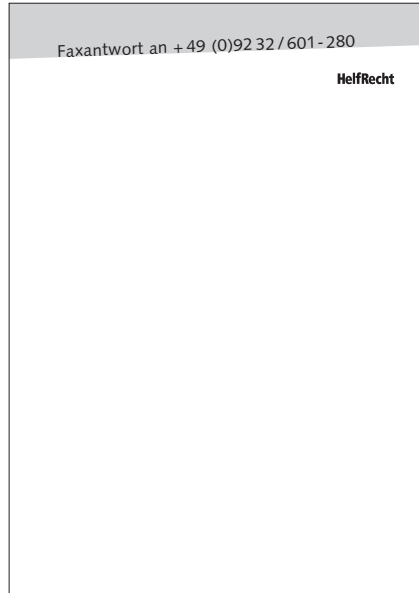
Typ A_ DIN A4 Mailing-Bogen Rückseite mit breitem Winkel in 21% Schwarz_Anmeldung, Bestellung

Mailingbögen Rückseite Typ A

Die Winkelfarbe auf der Rückseite ist für die Faxversion in 21% Schwarz.
Der graue Winkel tritt nur in Kombination mit der Faxnummer auf.

Auf der Rückseite steht der Winkel nie in der HelfRecht Primärfarbe HKS 16.

Der breite Winkel auf der Rückseite kann jedoch für Mailingaktionen (z.B. Frühlingsaktion) mit einem Bildmotiv plus Information eingesetzt werden. Die Schrift im Winkel kann auf der Rückseite in HKS 16N stehen, muss aber nicht.

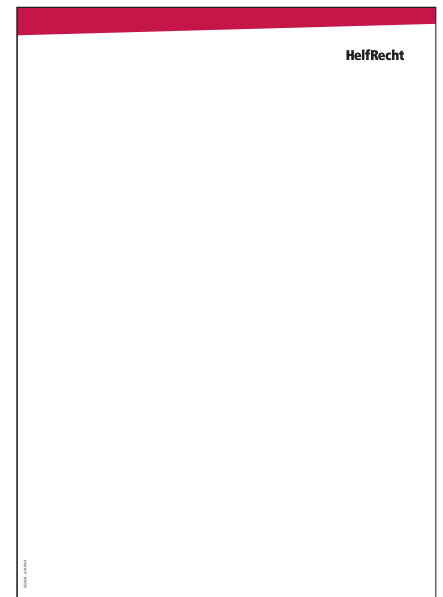
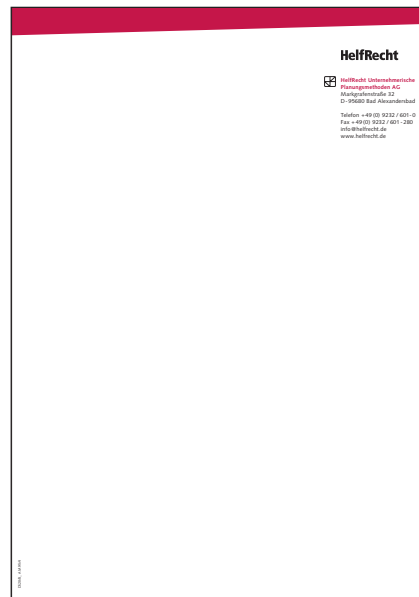


Mailingbögen
Typ B

Die schmalen Winkel der Mailingbögen sind ohne Information. Sie sind immer in der HelfRecht Primärfarbe HKS 16N. Die Mailingbögen mit dem schmalen Winkel werden nicht auf der Rückseite bedruckt.

Für umfangreichere Informationen steht ein 2. Blatt zur Verfügung. Die Wortmarke auf dem 2. Blatt ist idealerweise identisch in Größe und Stand mit der 1. Seite.

Typ B_ DIN A4 Mailing-Bogen mit schmalen Winkel in HKS 16_ mit Beispiel Beschriftung Programm



Weitere Beispiele und detaillierte Beschriftungsmuster und Vermaßung liegen vor.

Stellenanzeige

Stellenanzeige hier am Beispiel von 90 x 119 mm

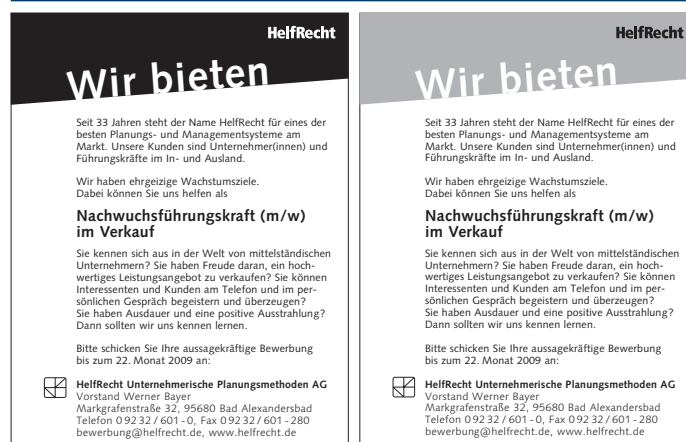


Stellenanzeige schwarz-weiß, Abb. in 50%

Die HelfRecht Stellenanzeigen können sowohl 2-farbig: Schwarz und HKS 16 (bzw. 4fbg, das Rot wird aus CMYk aufgebaut, siehe Farben) als auch schwarz-weiß erscheinen. In diesem Fall kann der horizontale Winkel in schwarz oder in Grau (Schwarz-Raster) angelegt werden.

Die Höhe der 2-spaltigen Anzeige richtet sich nach der Textlänge. Der linke Rand (17,5 mm) bleibt immer erhalten, damit das Bildzeichen genügend Raum bekommt und nicht an den Rand gedrängt wird.

Ein Rahmen mit 0,5 Punkt in schwarz begrenzt die Anzeige.

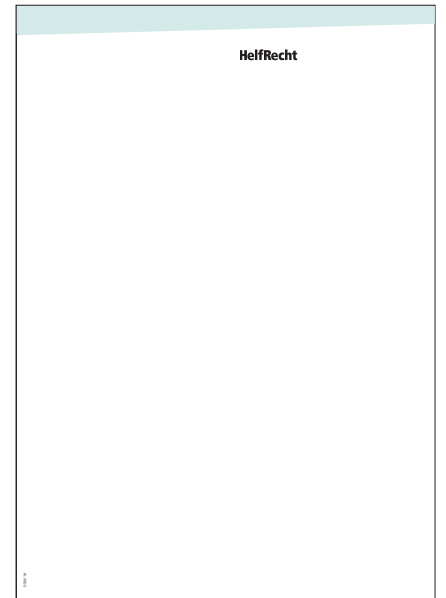
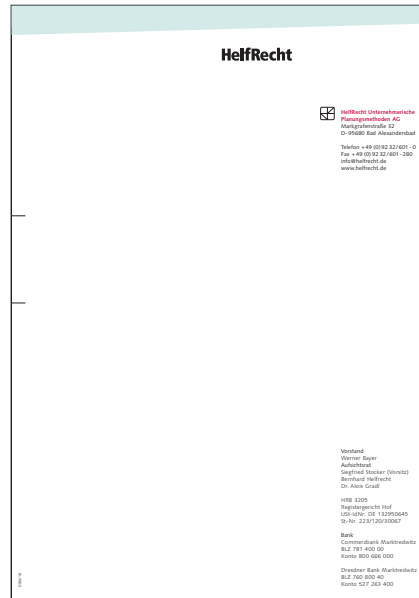


Geschäftsausstattung

Die Geschäftsausstattung von HelfRecht verwendet ausschließlich die sekundäre Identitätsfarbe Pantone 9040 in der Winkelfläche. Die Firmierung wird auf dem Briefbogen in der primären Identitätsfarbe HKS 16 gestellt. Angaben zum Unternehmen in Schwarz.

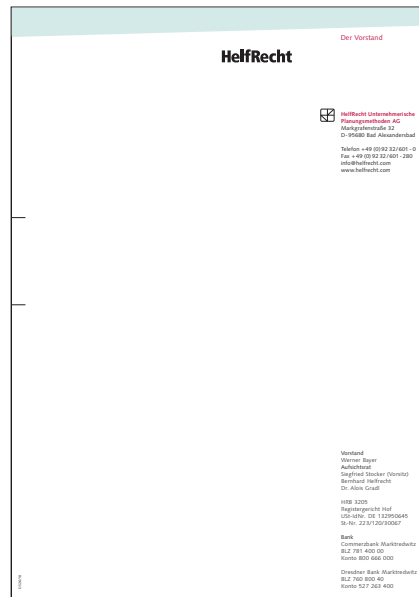
Briefbogen 1. Seite, Abb. 25%

Briefbogen 2. Seite

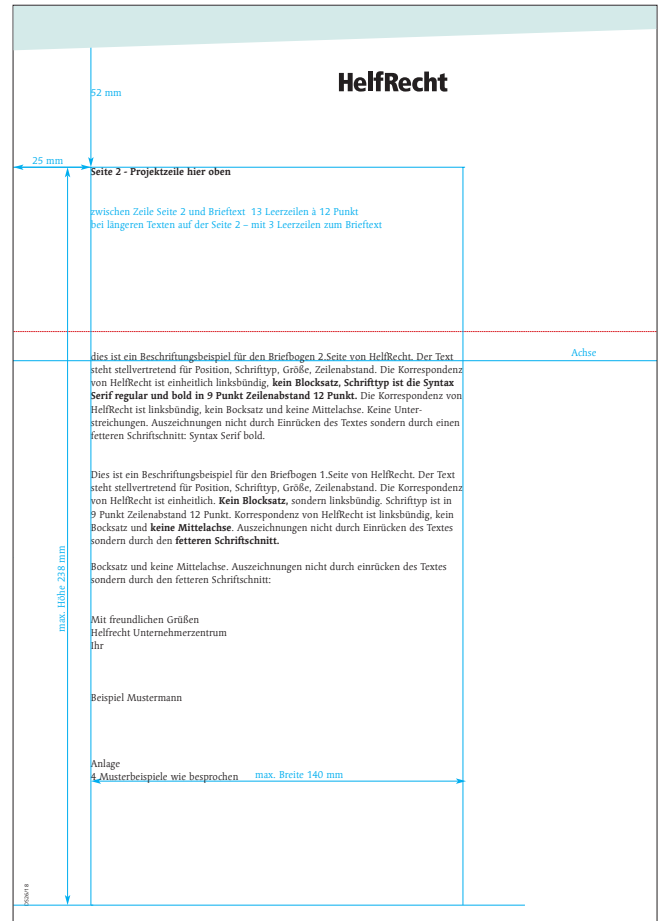
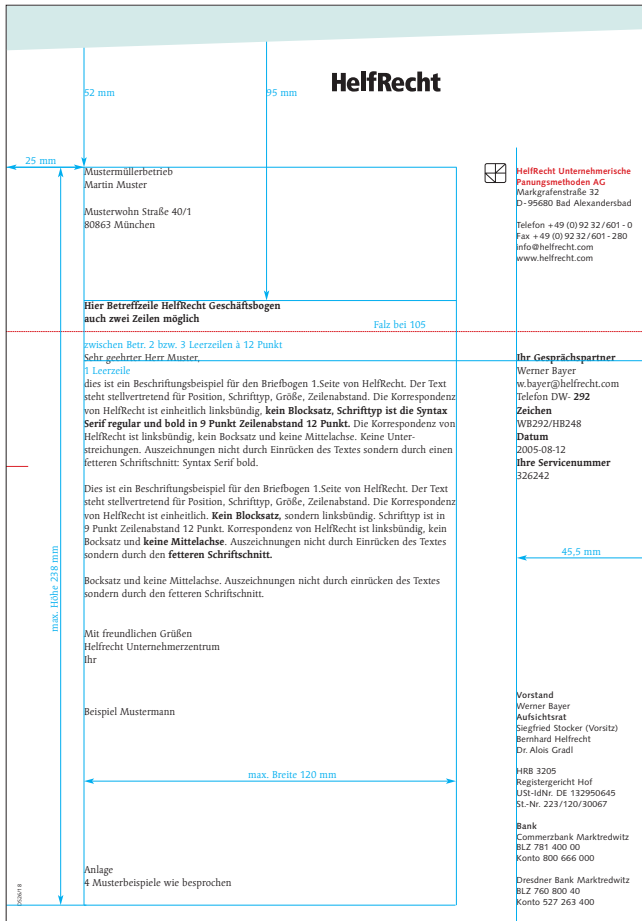


Briefbogen Vorstand 1. Seite

Kurzbrief



Beschriftung Briefbogen 1. und 2. Seite



Beschriftung Briefbogen

Jedes Schriftstück übermittelt nicht nur den Inhalt.

Durch die einheitliche Beschriftung der HelfRecht Briefbögen entsteht ein geschlossenes, leicht erkennbares Bild nach außen. Nur ein richtig beschrifteter Briefbogen bildet mit dem Vordruck die gewünschte Einheit.

Größere Abbildung siehe nächste Seite (Rückseite)

HelfRecht

52 mm

95 mm

25 mm

Mustermüllerbetrieb
Martin Muster

Musterwohn Straße 40/1
80863 München

Hier Betreffzeile HelfRecht Geschäftsbogen
auch zwei Zeilen möglich

Falz bei 105

zwischen Betr. 2 bzw. 3 Leerzeilen à 12 Punkt

Sehr geehrter Herr Muster,
1 Leerzeile

dies ist ein Beschriftungsbeispiel für den Briefbogen 1.Seite von HelfRecht. Der Text steht stellvertretend für Position, Schrifttyp, Größe, Zeilenabstand. Die Korrespondenz von HelfRecht ist einheitlich linksbündig, **kein Blocksatz, Schrifttyp ist die Syntax Serif regular und bold in 9 Punkt Zeilenabstand 12 Punkt.** Die Korrespondenz von HelfRecht ist linksbündig, kein Bocksatz und keine Mittelachse. Keine Unterstreichungen. Auszeichnungen nicht durch Einrücken des Textes sondern durch einen fetteren Schriftschnitt: Syntax Serif bold.

Dies ist ein Beschriftungsbeispiel für den Briefbogen 1.Seite von HelfRecht. Der Text steht stellvertretend für Position, Schrifttyp, Größe, Zeilenabstand. Die Korrespondenz von HelfRecht ist einheitlich. **Kein Blocksatz,** sondern linksbündig. Schrifttyp ist in 9 Punkt Zeilenabstand 12 Punkt. Korrespondenz von HelfRecht ist linksbündig, kein Bocksatz und **keine Mittelachse.** Auszeichnungen nicht durch Einrücken des Textes sondern durch den **fetteren Schriftschnitt.**

Bocksatz und keine Mittelachse. Auszeichnungen nicht durch einrücken des Textes sondern durch den fetteren Schriftschnitt.

Mit freundlichen Grüßen
Helfrecht Unternehmerzentrum
Ihr

Beispiel Mustermann

max. Breite 120 mm

Anlage
4 Musterbeispiele wie besprochen



**HelfRecht Unternehmerrische
Panungsmethoden AG**
Markgrafenstraße 32
D-95680 Bad Alexandersbad

Telefon +49 (0) 92 32/601 - 0
Fax +49 (0) 92 32/601 - 280
info@helfrecht.com
www.helfrecht.com

Ihr Gesprächspartner
Werner Bayer
w.bayer@helfrecht.com
Telefon DW- 292
Zeichen
WB292/HB248
Datum
2005-08-12
Ihre Servicenummer
326242

45,5 mm

max. Höhe 238 mm

Vorstand
Werner Bayer
Aufsichtsrat
Siegfried Stocker (Vorsitz)
Bernhard Helfrecht
Dr. Alois Gradl

HRB 3205
Registergericht Hof
USt-IdNr. DE 132950645
St.-Nr. 223/120/30067

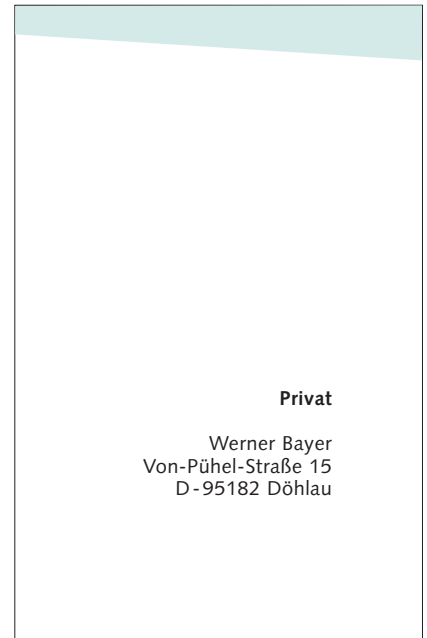
Bank
Commerzbank Marktredwitz
BLZ 781 400 00
Konto 800 666 000

Dresdner Bank Marktredwitz
BLZ 760 800 40
Konto 527 263 400

Visitenkarten 54 x 84 mm, mit Rückseite Abb. 100%

Visitenkarten

Die HelfRecht Visitenkarten sind im Hochformat angelegt. Die Wortmarke bekommt dadurch Raum zur Wirkung. Der Name steht dominant in der primären Identitätsfarbe HKS 16 gestellt. Die Rückseite ist für die Privatadresse vorbehalten.



Kuvert DIN lang

Für den Geschäftsbriefverkehr wird das Kuvert mit dem Winkel in der sekundären Identitätsfarbe Pantone 9040 verwendet.
Die C4 Kuverts für Geschäftsdrucksachen sind nicht mit einem Slogan oder einem Motto bedruckt.

Briefkuvert DIN lang, für Geschäftsbriefe, Abb. 50%



Für Mailing-Aktionen wird der Winkel in der primären Hausfarbe HKS 16 verwendet. Zusätzlich kann, passend zum Inhalt des Kuverts, auf die Vorderseite das Aktionsmotto gestellt werden.

Aktionskuvert DIN lang, für Mailingaktionen, Abb. 50%



Aktionskuvert für DIN A4, für Mailingaktionen, Abb. 30%

Kuvert DIN C4

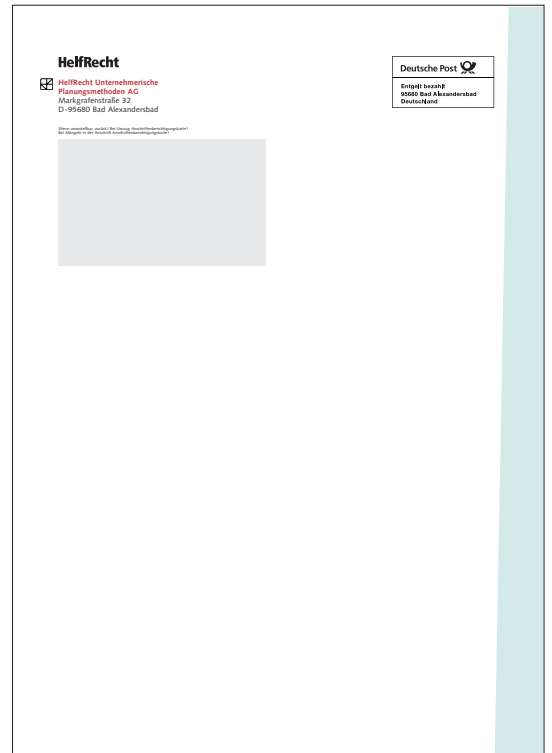
Ausnahme:

Bei großen Kuverts wird der Winkel rechts positioniert.

Bei Mailingaktionen ist der Winkel in der primären Hausfarbe HKS 16.
Zusätzlich kann, passend zum Inhalt des Kuverts, auf die Vorderseite das Aktionsmotto gestellt werden.

Bei Geschäftsdrucksachen (z.B. umfangreiche Schriftstücke, Angebote etc.) die im C4 versendet werden ist der Winkel in der sekundären Identitätsfarbe Pantone 9040. Die C4 Kuverts für Geschäftsdrucksachen sind nicht mit einem Slogan oder einem Motto bedruckt.

Die Winkelfarbe vom Kuvert entspricht der Winkelfarbe der Inhaltsseiten.





HelfRecht Internetauftritt

Layoutentwurf und -beschreibung



Unser Tagesleitmotto

Es ist nicht zu wenig Zeit, die wir haben, sondern es ist zuviel Zeit, die wir nicht nutzen

HelfRecht

Beflügeln Sie Ihr Unternehmen

HelfRecht Planungstage



- für mehr Lebenserfolg
- für mehr Unternehmenserfolg
- Firmeninterne Trainings
- Referenz
- Termine

HelfRecht Produkte - für mehr Erfolg



- Planer
- Bücher
- Zeitschrift
- Mobilsystem
- Planungshilfen
- Kunden werben Kunden

Kundenmeinung



„Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.“

Startseite

Unternehmen

Chefbrief

Know How

Termine

Presse

Impressum

Suche

News



Mit dem Baustein Tagesplanung organisieren Sie souverän Ihren individuellen Tagesablauf. Ziele und Projekte gliedern Sie in einzelne Schritte, die Sie nacheinander abarbeiten. So erreichen Sie die geplanten Ergebnisse, ohne wesentliche Aktivitäten zu vergessen oder zu vernachlässigen. [\[mehr \]](#)

Planungstage



Erfolg lässt sich planen. Dies gilt für unternehmerische berufliche Herausforderungen ebenso wie für ganz persönliche Aufgaben und Ziele.

Das hierfür nötige Know-how sowie geeignete Umsetzungswerkzeuge lernen Sie in den Planungstagen bei HelfRecht kennen. [\[mehr \]](#)

Strategie & Planung - beflügeln Sie Ihr Unternehmen



Regionale Unternehmertreffs: Wir kommen zu Ihnen!

HelfRecht-System jetzt auch in Polen

HelfRecht-Katalog 2008 mit allen Neuheiten

Sparen mit dem Bildungsscheck!

Chefbrief von Helfrecht

Seitenlayout

- 990 Pixel Breite, zentriert
 - Feste Seitenbreite
 - Optimal für Bildschirmauflösungen ab 1024x768 Pixel
 - Geringere Auflösungen kaum noch in Gebrauch
- Dynamische Höhe
 - Anpassung der Seitenhöhe an den Inhalt
 - Seitenfuß immer am unteren Rand des Viewports (Browserfenster), auch bei wenig Inhalt



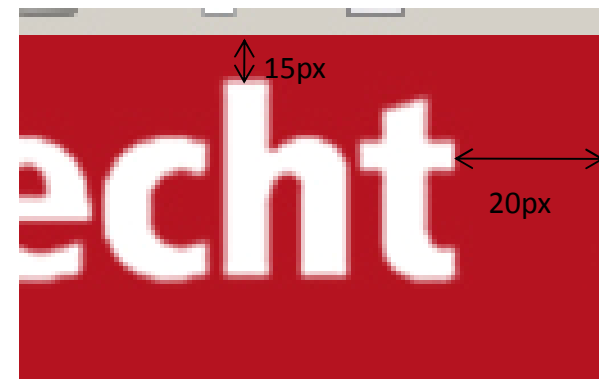
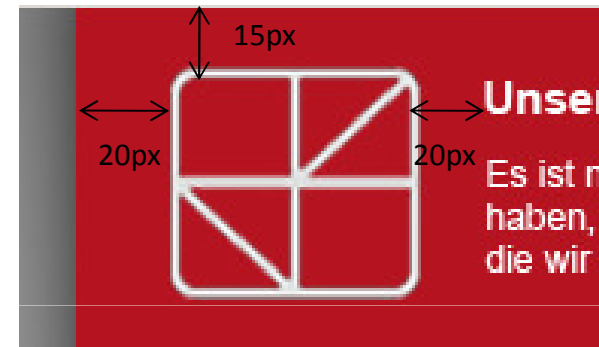
990px



- Aufgreifen der Winkelfläche im Verhältnis 4 : 3
- Höhe 150 Pixel
- Wortmarke und Claim auf der rechten Seite
- Bildzeichen und Tagesleitmotto auf der linken Seite

Position Wortmarke und Bildzeichen

- Bildzeichen 15 Pixel vom oberen Rand und 20 Pixel vom linken Rand entfernt
- Wortmarke ebenfalls 15 Pixel vom oberen Rand und 20 Pixel vom linken Rand entfernt



Navigation



- Hauptnavigation aufgeteilt in drei Punkte: Planungstage, Produkte und Kundenreferenzen (mit aktueller Referenz)
- Darunter die wichtigsten Unterpunkte
- Bei Aktivierung eines Hauptmenüpunktes Untermenü in Leiste darunter

Inhaltsbereiche

- Jeder Inhaltsbereich (z.B. News, Artikel etc.) in separater Box
- Wiederaufgreifen der Schräge hinter der Inhaltsüberschrift
- Kurze Zusammenfassung des Inhalts, mit Klick auf „mehr“ zur Detailansicht
- Bild in fester Breite zu jedem Bereich, Link zur Detailansicht

News



Mit dem Baustein Tagesplanung organisieren Sie Ihren Tagesablauf. Ziele und Projekte gliedern Sie nacheinander abarbeiten. So erreichen Sie die Aktivitäten zu vergessen oder zu vernachlässigen.

Planungstage

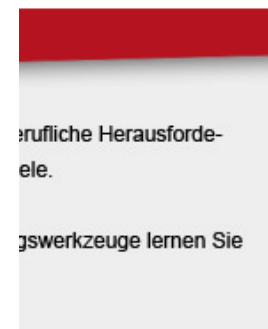
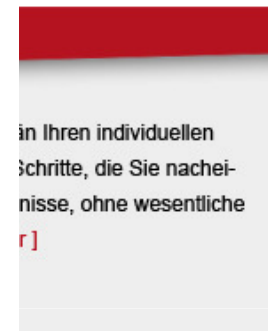


Erfolg lässt sich planen. Dies gilt für Unternehmen ebenso wie für ganz persönliche Projekte.

Das hierfür nötige Know-how sowie geeignete Tools sind im HelfRecht-Portal zu finden.

Info-Bereich

- Werbung für Produkte oder allgemeine Informationen im Info-Bereich
- Breite: 180 Pixel
- Dynamische Höhe (inhaltsabhängig)
- Abstand zum Inhaltsbereich: 100 Pixel
- Rote Schräge des Inhaltsbereich setzt sich fort



Strategie & Planung - beflügeln Sie Ihr Unternehmen

Werner Epper | Christoph Beck

STRATEGIE UND PLANUNG

28 Erfolgsrezepte für eine zukunftsorientierte Unternehmensführung

Regionale Unternehmertreffs:
Wir kommen zu Ihnen!

HelfRecht-System jetzt auch in Polen

HelfRecht-Katalog 2008
mit allen Neuheiten

Sparen mit dem **Bildungsscheck!**

Schriftarten und Schriftgrößen

- Tagesmotto
 - Überschrift 15pt Arial Bold #FFFFFF
 - Text 12pt Arial #FFFFFF
- Navigation
 - Hauptpunkte 15pt Arial Bold #FFFFFF
 - Unterpunkte Arial 11pt #FFFFFF
- Kundenreferenz
 - 11pt Times Italic #FFFFFF

Schriftarten und Schriftgrößen

- Inhalts- und Infobereich
 - Hauptüberschriften 23pt Arial #FFFFFF
 - Unterüberschriften 18pt Arial #000000
 - Fließtext 12pt Arial #000000
- Links
 - 12pt Arial #B51521



Unser Tagesleitmotto

Es ist nicht zu wenig Zeit, die wir haben, sondern es ist zuviel Zeit, die wir nicht nutzen

HelfRecht

Beflügeln Sie Ihr Unternehmen

HelfRecht Planungstage



- für mehr Lebenserfolg
- für mehr Unternehmenserfolg
- Firmeninterne Trainings
- Referenz
- Termine

HelfRecht Produkte - für mehr Erfolg



- Planer
- Bücher
- Zeitschrift
- Mobilsystem
- Planungshilfen
- Kunden werben Kunden

Kundenmeinung



„Dieser Text steht anstelle des eigentlichen Inhaltes. Er hat keine Beziehung zum vorliegenden Layout, sondern dient dazu, die Typografie und das Erscheinungsbild von HelfRecht praxisgetreu vorzuführen.“

[Startseite](#)

[Unternehmen](#)

[Chefbrief](#)

[Know How](#)

[Termine](#)

[Presse](#)

[Impressum](#)

[Suche](#)

News



Mit dem Baustein Tagesplanung organisieren Sie souverän Ihren individuellen Tagesablauf. Ziele und Projekte gliedern Sie in einzelne Schritte, die Sie nacheinander abarbeiten. So erreichen Sie die geplanten Ergebnisse, ohne wesentliche Aktivitäten zu vergessen oder zu vernachlässigen. [\[mehr \]](#)

Planungstage



Erfolg lässt sich planen. Dies gilt für unternehmerische berufliche Herausforderungen ebenso wie für ganz persönliche Aufgaben und Ziele.

Das hierfür nötige Know-how sowie geeignete Umsetzungswerkzeuge lernen Sie in den Planungstagen bei HelfRecht kennen. [\[mehr \]](#)

Strategie & Planung - beflügeln Sie Ihr Unternehmen



Regionale Unternehmertreffs:
Wir kommen zu Ihnen!

HelfRecht-System jetzt auch in Polen

HelfRecht-Katalog 2008
mit allen Neuheiten

Sparen mit dem **Bildungsscheck!**

Chefbrief von Helfrecht

Vielen Dank für die Aufmerksamkeit!