Informatik Programmierung

–

Sommersemester 2015

```
<script type="text/javascript">
    /*<![CDATA[*/
<!--
        // JS function for mouse-over
    function over(name,imgObj)  {   //
        if (version == "n3" && document[name]) {docu
        else if (document.getElementById && document
        else if (imgObj)    {imgObj.src = eval(name+
    }
        // JS function for mouse-out
    function out(name,imgObj)   {   //
        if (version == "n3" && document[name]) {docu
        else if (imgObj)    {imgObj.src = eval(name+
    }

// -->
    /*]]>*/
</script>
```

Kontakt:

Email: malte.wattenberg@t-online.de
Telefon: 0163 – 68 60 877

Vorstellung

- Das bin ich…

- Kontakt:
  - malte.wattenberg@t-online.de
  - 0163 – 68 60 877

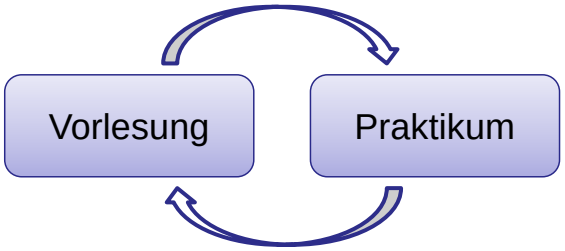
- Wer sind Sie?
  - Warum Wirtschaftsinformatik?
  - Welche Erwartungen gibt es an das Fach?
  - Welche Vorkenntnisse?

Organisatorisches

- Material steht online zur Verfügung unter:
  www.malte-wattenberg.de

- Literaturhinweise je Kapitel
- Zusammenfassende Fragen nach jedem Kapitel/Abschnitt

- Lehrformen:
  - Seminaristische Vorlesung
    Reflektion des Praktikums und neuer Input
  - Praktikum
    Anwendung des Erlernten



Organisatorisches

- 12 Termine am Freitag

  Vorlesung: 14:00 Uhr – 15:30 Uhr
  Praktikum: 15.45 Uhr – 17:15 Uhr

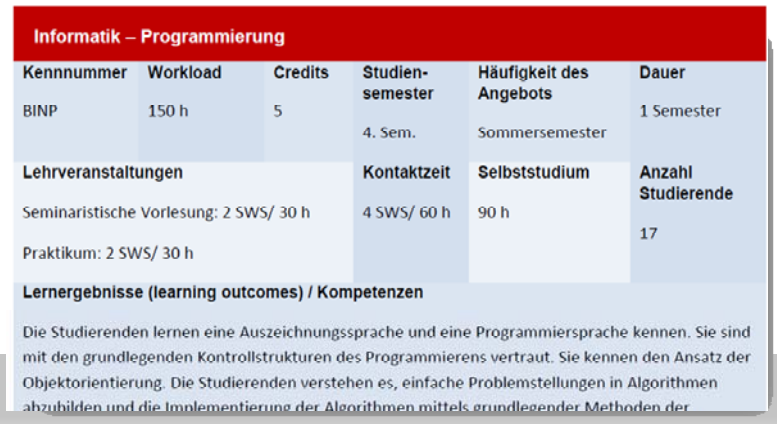
10.04.2015
17.04.2015
24.04.2015
08.05.2015
15.05.2015 ?
22.05.2015
29.05.2015
05.06.2015 ?
12.06.2015
19.06.2015
26.06.2015
03.07.2015

- Während der Vorlesung bleiben die Rechner bitte aus
- bei Fragen bitte unterbrechen
- Anwesenheit keine Pflicht, aber dringend anzuraten

- Schriftliche Prüfung (60 o. 90 min) über die Vorlesung **und** Praktikum
  - Es kann eine handbeschriebene DIN A4-Seite mitgebracht werden
  - Weitere Vorbereitungsstunde vor der Prüfung, Besprechung alter Klausuren

Ziele der Veranstaltung

- Sie wissen was Programmieren ist
- Sie kennen grundlegende Webtechnologien wie HTML, CSS, XML, JavaScript näher
- Sie kennen grundlegende Kontrollstrukturen des Programmierens
- Sie können einfache Problemstellungen in Algorithmen abbilden
- Sie haben die Sache selbst ausprobiert.



Informatik – Programmierung

Kennnummer	Workload	Credits	Studien-semester	Häufigkeit des Angebots	Dauer
BINP	150 h	5	4. Sem.	Sommersemester	1 Semester

Lehrveranstaltungen	Kontaktzeit	Selbststudium	Anzahl Studierende
Seminaristische Vorlesung: 2 SWS/ 30 h Praktikum: 2 SWS/ 30 h	4 SWS/ 60 h	90 h	17

Lernergebnisse (learning outcomes) / Kompetenzen

Die Studierenden lernen eine Auszeichnungssprache und eine Programmiersprache kennen. Sie sind mit den grundlegenden Kontrollstrukturen des Programmierens vertraut. Sie kennen den Ansatz der Objektorientierung. Die Studierenden verstehen es, einfache Problemstellungen in Algorithmen abzubilden und die Implementierung der Algorithmen mittels grundlegender Methoden der

Inhalt der Veranstaltung

Kapitel 1: Einführung	(1. Termin)
Kapitel 2: HTML	(ca. 2 Termine)
Kapitel 3: CSS	(ca. 2 Termine)
Kapitel 4: XML	(1 Termin)
Kapitel 5: JavaScript	(ca. 6 Termine)

Lesen..Nachschlagen..Lernen..

- Literatur zu diesem Kapitel z.B.:

  Balzert, H.: Lehrbuch Grundlagen der Informatik,
  2. Aufl. 2004

Kapitel 1
Einführung

Grundbegriffe

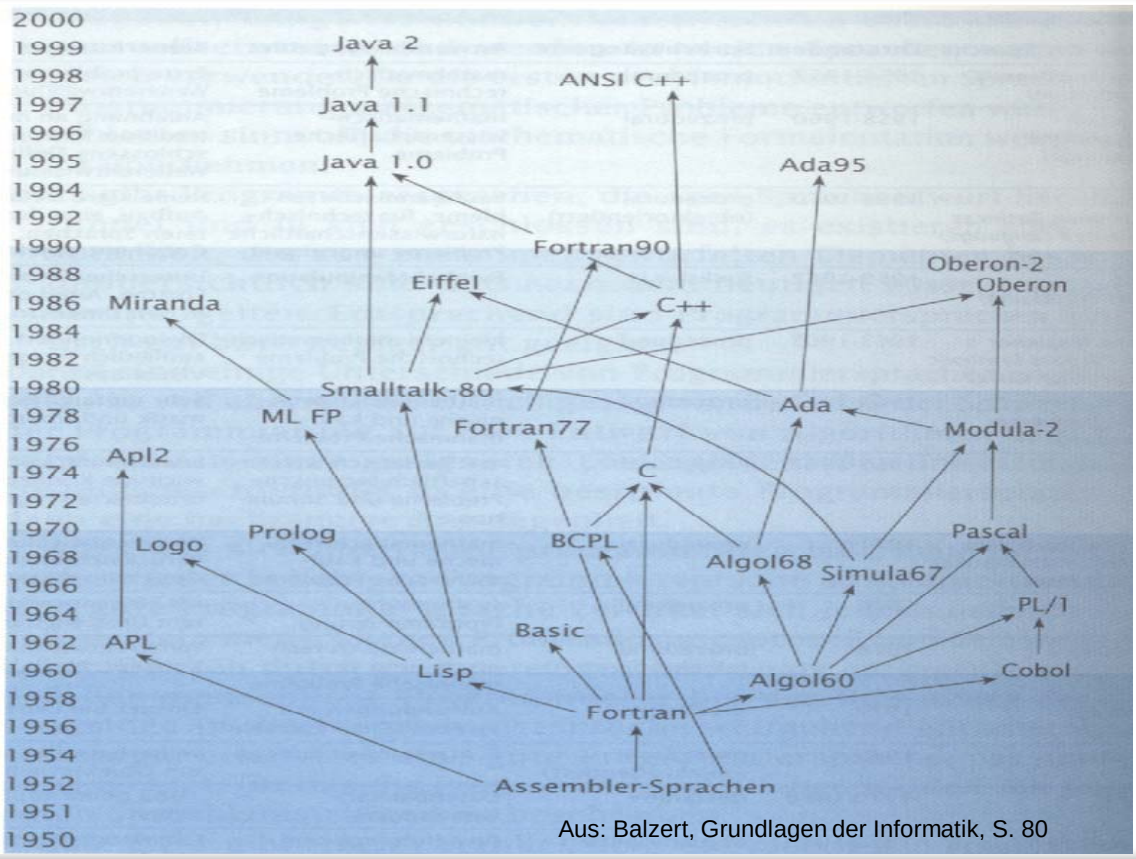
- Ein **Programm** …
  … ist eine Folge von **Anweisungen** nach den Regeln einer Programmiersprache und stellt eine Funktionalität zur Verfügung.

- Eine **Programmiersprache** …
  … ist die Sprache, in der ein Programm, das ein Rechner ausführen soll, geschrieben ist.

- Eine Programmiersprache ist durch die **Festlegung ihrer Syntax und Semantik** vollständig beschrieben.

- **Programmieren** ist das Erstellen eines Programms in einer Programmiersprache.

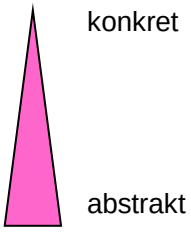
Geschichte



Aus: Balzert, Grundlagen der Informatik, S. 80

Klassifizierung der Sprachen

- Nach ihrem Abstraktionsgrad
  - Maschinennahe Sprachen → z.B. Assembler
  - Höhere Sprachen → z.B. C, C++, Java, JavaScript
  - Anwendungsorientierte Sprachen → z.B. MySQL



- Klassifizierung höherer Sprachen nach ihrem Grundansatz
  - Prozedural: Anweisungen sind Prozeduren
  - Funktional: Die Welt besteht aus Funktionen
  - Objektorientiert: Die Welt besteht aus Objekten
  - Logisch: Programme sind logische Aussagen

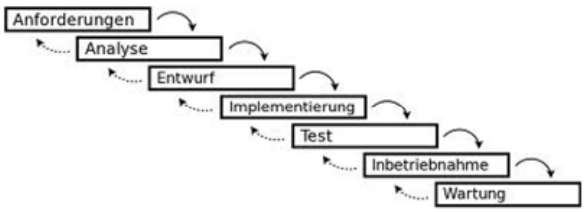
Wie wird programmiert?

- Programmcode wird von Programmierern in Textform erstellt.
- Ausführung des Quelltextes durch…
  - **Compiler**: Programm zur Übersetzung des Quelltextes in eine Zielsprache, bspw. Maschinencode >lauffähige Datei

    oder

  - **Interpreter**: Programm, das den Quelltext zeilenweise zur Laufzeit des Programms einliest und ausführt

Bei uns steht nicht der Prozess, sondern das Handwerk im Fokus. Tatsächlich werden bei der Softwareentwicklung untersch. Vorgehensmodelle und Methoden herangezogen → Fach DBSE



Kapitel 1
Einführung

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

HTML

- Dokumentbeschreibungssprache für Seiten im www
  - Grundlegende Sprache, sehr weit verbreitet
  - Einfach zu erlernen
  - Charakterisiert durch Hyperlinks, extern und intern
  - keinerlei Funktionalität

- Klartext:
  - Eingabe in Texteditor
  - Softwareunabhängig
  - Zeilenweise Interpretation durch Webbrowser

```
<html>
<head>
<title>Informatik Programmierung</title>
<LINK REL="stylesheet" TYPE="text/css" href="../../../css/content.css">
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<SCRIPT LANGUAGE="JavaScript" src="go.js"></SCRIPT>
</head>

<body>


<h1 align="center">Informatik Programmierung</h1>
<!--
<h3><a href="INP2006/INP2006.html"><font  color="red">Link zur alten Seite (am
10.3., 21 Uhr repariert!)</font></a></h3>
 -->
<p>
In der Veranstaltung <i>Informatik Programmierung</i> geht es darum,
grundlegende Konzepte der Programmierung kennen zu lernen und in praktischen
```


Standardisiert durch www-Consortium

Aktuell ist HTML 5

CSS

- Stylesheet Sprache (=Formatvorlage)
- Ideal kombinierbar mit HTML und XML
- Grundidee: Trennung von Inhalt und Layout der Webseiten
- Folge:
  - Einheitliche Darstellung
  - mehrere Layouts möglich, einfacher Wechsel
  - Unterscheidung zwischen verschiedenen Medien (Print, Handheld,…)

```
p.Farbeblau {
    border-color: blue;
    font-family: Verdana
    }

p.Kombination {
    border-top: double red 10px;
    border-bottom: solid blue 5px;
    border-left: dotted yellow 15px;
    border-right: dashed green 20px;
    font-family: Verdana
    }
```


CSS =

XML

- Meta-Auszeichnungssprache zum Austausch strukturierter Daten in Textform
- Metasprache: Möglichkeit zur Definition weiterer spezifischer Untersprachen
- Einsatz zum/als
  - Austausch von Daten und Dokumenten im www
  - Schnittstelle (z.B. eBay)

```
<?xml version="1.0"?>
<!DOCTYPE welt
    [
    <!ELEMENT welt (gruss*)>
    <!ELEMENT gruss (#PCDATA)>
    <!ATTLIST gruss
        id     ID       #IMPLIED>
    ]>
<?xml-stylesheet type="text/css" href="style.css" ?>
<welt>
    <!-- meine vierte XML-Datei-->
    <gruss id="1">Hallo Welt</gruss>
    <gruss id="2"><![CDATA[<<Wie gehts?>>]]></gruss>
    <gruss />
</welt>
```


XML =

JavaScript

- Vollwertige Programmiersprache
  - Anweisungen werden zur Laufzeit Zeile für Zeile interpretiert und ausgeführt > Interpretersprache
  - Scriptsprache: Skripte sind i.d.R. kleine Programme
  - Objektorientiert
  - Plattformübergreifend
  - Clientseitig

```
<script language="JavaScript">
<!--
function Schaltjahr(jahr)
{
if (jahr % 4 != 0)  //Falls das Jahr nicht ganzzahlig durch vier teilbar ...
    return false;   //.. gib false zurück

//ab hier wissen wir, dass die Zahl durch vier teilbar ist!

if (jahr % 400 == 0)    //Falls es durch 400 teilbar ist
    return true;        //gib true zurück

if (jahr % 100 == 0)    //Falls es durch 100 teilbar ist,
    return false;       //gib false zurück

return true;
}
```

JavaScript

- Eingebunden in HTML-Dokument
  - Ausführung durch Webbrowser (=Interpreter)
  - Sandbox Prinzip: Nur Zugriff auf Objekte des Browsers, nicht auf Dateisystem! (>Sicherheit ist Problem des Browsers)
- Erweiterung von statischen HTML-Seiten um dynamische Aspekte, z.B.:
  - Korrekte Eingabe eines Formulars prüfen, bevor es versendet wird
  - Kleine Anwendungen laufen im Browser
- Missbrauch möglich >Imageproblem

Zusammenfassende Fragen: Kapitel 1

- Was ist ein Programm?
- Was ist eine Programmiersprache?
- Was ist der unterschied zwischen Compilern und Interpretern?
- Was charakterisiert
  - HTML
  - CSS
  - XML
  - JavaScript?

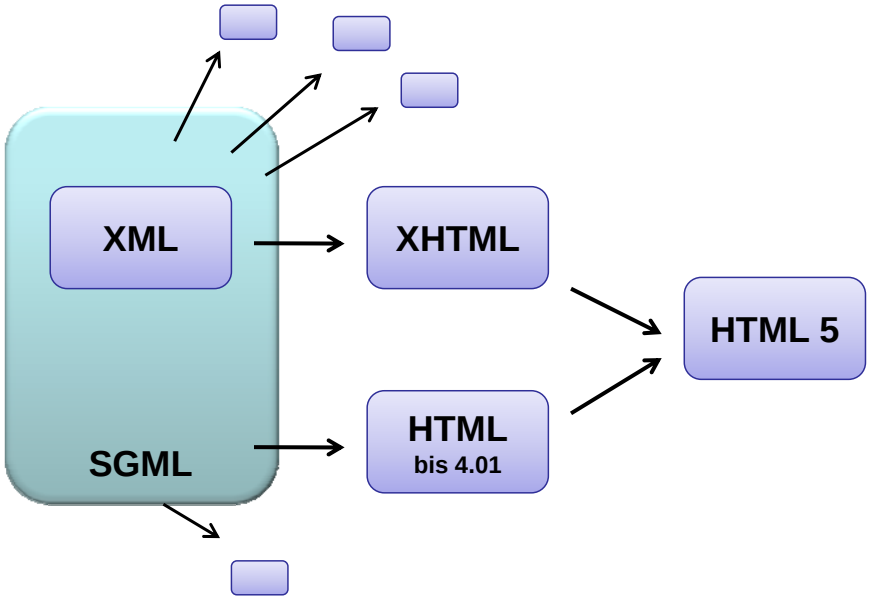
Literatur

- Münz; Gull; HTML5-Handbuch; München; 2011
- Freeman, Robson,
  "HTML5-Programmierung von Kopf bis Fuß",
  O'Reilly; 2012
- http://de.selfhtml.org
- http://www.w3schools.com

Kapitel 2
HTML

Auszeichnungssprachen

- Auszeichnungssprachen strukturieren Dokumente durch Auszeichnungen einzelner Bereiche



Hochschule Ostwestfalen-Lippe
University of Applied Sciences

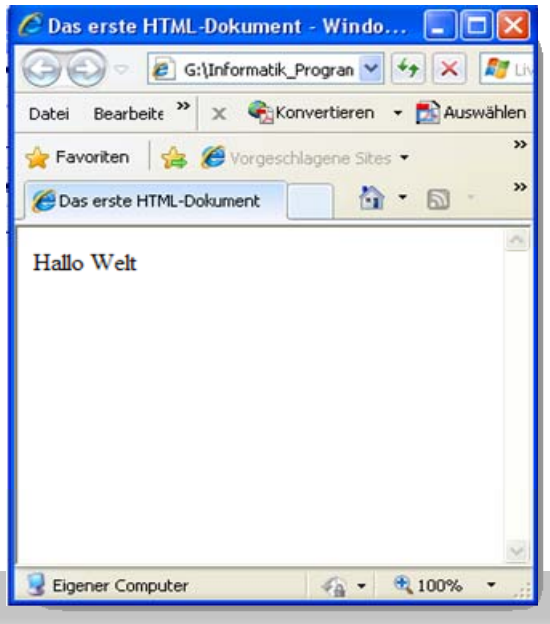
Zusammenspiel von HTML und CSS

- HTML legt die Struktur fest > Auszeichnungssprache
- CSS legt das Aussehen fest > Stylesheet Sprache

Inhalt	Unsere Produkte	
↓		
Struktur	`<h1>Unsere Produkte</h1>`	Unsere Produkte
↓		
Aussehen	`<h1 style="color:blue">` `Unsere Produkte</h1>`	Unsere Produkte

Hallo Welt

```
<!DOCTYPE html>
 <html>
 <head>
  <title> Mein erstes HTML-Dokument </title>
 </head>

 <body>

 Hallo Welt :)

 </body>
</html>
```



- Dateiname hat die Endung .htm oder .html

Achtung: in der Browserzeile gibt es keine Leerzeichen. Viele Server scheitern an der Umwandlung.
Deswegen: Niemals die HTML-Dokumente (oder andere) mit Leerzeichen im Dateinamen abspeichern.

Festlegung des Dokumenttyps

- Die Document Type Definition (DTD) zu Beginn des Dokumentes legt fest, welche Strukturelemente gültig sind, z.B.:

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

<!DOCTYPE HTML>
```

← **HTML 5** ☺

- Hilft dem Browser zu erkennen, welches Format und Version das Dokument besitzt und welche Elemente gültig=valide sind
  - strict: exakte Definitionen müssen eingehalten werden
  - transitional/ frameset: veraltete Elemente dürfen weiterhin genutzt werden

- Validität wichtiges Thema zur korrekten Darstellung im Browser und für eventuelle weitere Verarbeitung und Suchmaschinen

Dies reicht für ein Grundverständnis.
Weitere Informationen zur DTD folgt im Kapitel XML ☺

Elemente und Tags

- Ein Tag ist eine Auszeichnung (und wird in Auszeichnungssprachen benutzt ☺) eines Text- bzw. Datenbereichs

```
<tagname> ... Inhalte ... </tagname>
```

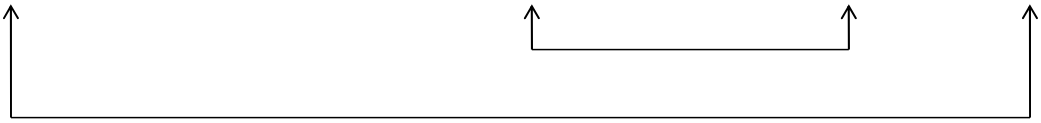
öffnender Tag — schließender Tag

- Tags müssen geschlossen werden (einige erlauben eine Kurzschreibweise)
- Öffnender + schließender Tag bilden ein Element
- Tags werden kleingeschrieben (stricte Anwendung)

Elemente und Tags

- Tags dürfen geschachtelt sein
- Tags müssen ebenentreu - paarig geschachtelt sein

```
<tagname> ... Inhalte ... <neuertag>Inhalt</neuertag></tagname>
```



- Unterscheidung in
  - Block-Level-Elemente: führen zu einen automatischen Zeilenumbruch am Ende, z.B. Überschrift
  - Inline-Elemente: zeichnen Teile von Inhalte aus, kein Zeilenumbruch, z.B. Fettdruck

Enthalten Elemente Unterelemente, so spricht man von „Eltern-“ bzw. „Kindelementen“.

Nicht jedes Element darf zu jedem anderen Element ein Kindelement sein.

Nicht alle Elemente dürfen Unterelemente enthalten, andere benötigen zwingend welche.

Attribute

- Attribute (Eigenschaften) spezifizieren Tags näher

```
<tagname attribut="wert"> ... Inhalte ... </tagname>
```

- 3 Wertarten für Attribute:
  - Normiert
    `<p align="right">` rechtsbündiger Absatz
  - Zahlenwerte (prozentual oder absolut)
    `<table width="600px">` Tabelle, Breite von 600 Pixel
    `<table width="50%">` Tabelle, 50% der Bildschirmbreite
  - Variable Werte
    `<img src="bild1.jpg">` eingefügtes Bild

Manche Tags benötigen zwingend wenigstens ein Attribut.
Beispiel?

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Grundgerüst eines HTML Dokumentes

- Umschließendes Element `<html></html>`
- Kopfbereich `<head></head>`
  - Enthält Informationen zu Autor, Beschreibungen, weitere Dateien
  - Nicht sichtbar im Browser, (außer `<title>`)
  - Title Element ist zwingend (auch wenn es leer bleibt)
- Rumpf `<body></body>`
  - Eigentlicher Inhalt, der strukturiert werden kann
  - Wird im Browser angezeigt

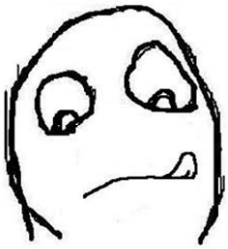
```
<html>

<head>
<title> Hallo </title>
</head>

<body>
Auf unserer tollen
Website finden Sie
alle Neuigkeiten.
</body>

</html>
```


<html> erlaub nur zwei Kindelemente: <head> und <body>.
<head> ist das Elternelement von <title>.
<title> ist das verpflichtende Kindelement von <head>, auch wenn es leer bleibt.

Nicht jedes Element darf zu jedem anderen Element ein Kindelement sein.

Zusammenfassende Fragen: Abschnitt 2.1

- Was ist eine Auszeichnungssprache?
- Wie spielen HTML und CSS zusammen?
- Was sind Tags und Attribute?
- Wie werden Attribute eingesetzt?
- Wie sieht das Grundgerüst einer HTML Datei aus?
- Wo befindet sich der Title Tag



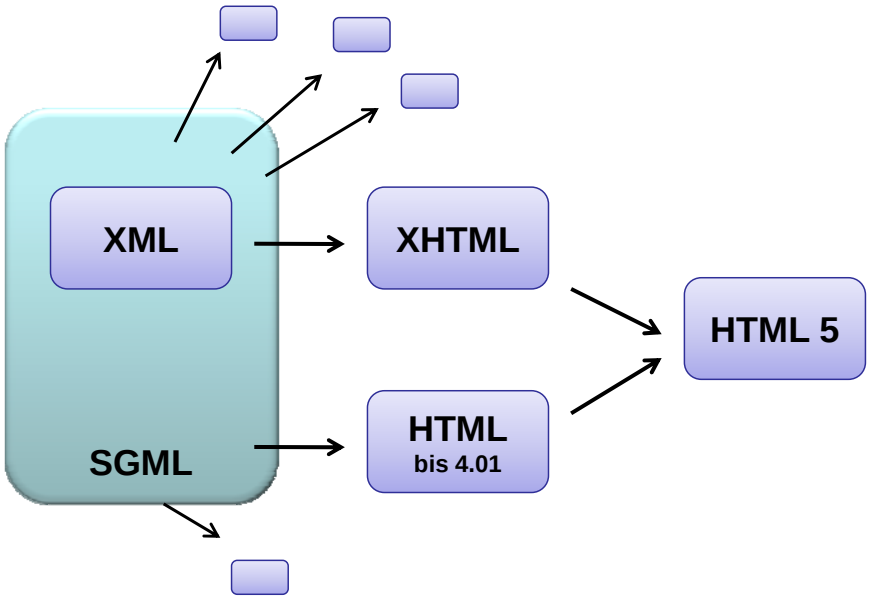
Literatur

- Münz; Gull; HTML5-Handbuch; München; 2011
- Freeman, Robson,
  "HTML5-Programmierung von Kopf bis Fuß",
  O'Reilly; 2012
- http://de.selfhtml.org
- http://www.w3schools.com

Kapitel 2
HTML

Auszeichnungssprachen

- Auszeichnungssprachen strukturieren Dokumente durch Auszeichnungen einzelner Bereiche



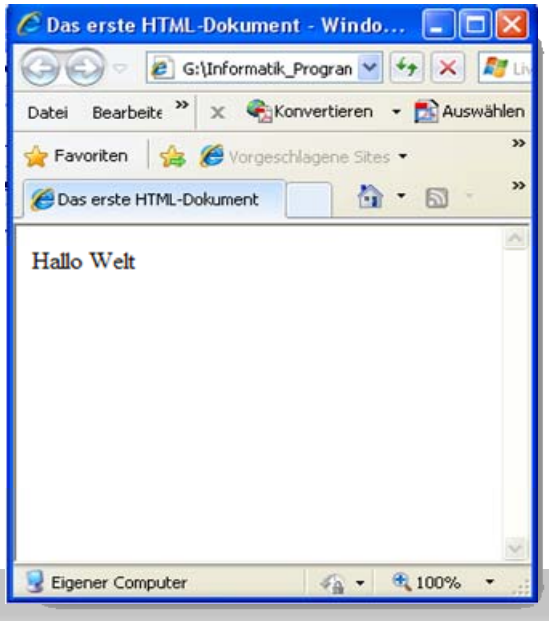
Zusammenspiel von HTML und CSS

- HTML legt die Struktur fest > Auszeichnungssprache
- CSS legt das Aussehen fest > Stylesheet Sprache

Inhalt	Unsere Produkte	
↓		
Struktur	`<h1>Unsere Produkte</h1>`	Unsere Produkte
↓		
Aussehen	`<h1 style="color:blue">` `Unsere Produkte</h1>`	Unsere Produkte

Hallo Welt

```
<!DOCTYPE html>
 <html>
 <head>
  <title> Mein erstes HTML-Dokument </title>
 </head>

 <body>

 Hallo Welt :)

 </body>
</html>
```



- Dateiname hat die Endung .htm oder .html

Achtung: in der Browserzeile gibt es keine Leerzeichen. Viele Server scheitern an der Umwandlung. Deswegen: Niemals die HTML-Dokumente (oder andere) mit Leerzeichen abspeichern.

Festlegung des Dokumenttyps

- Die Document Type Definition (DTD) zu Beginn des Dokumentes legt fest, welche Strukturelemente gültig sind, z.B.:

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

<!DOCTYPE HTML>
```

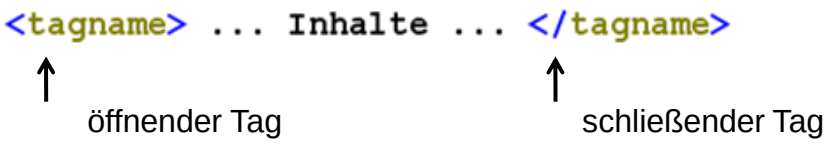
← **HTML 5** ☺

- Hilft dem Browser zu erkennen, welches Format und Version das Dokument besitzt und welche Elemente gültig=**valide** sind
  - strict: exakte Definitionen müssen eingehalten werden
  - transitional/ frameset: veraltete Elemente dürfen weiterhin genutzt werden
- **Validität** wichtiges Thema zur korrekten Darstellung im Browser und für eventuelle weitere Verarbeitung und Suchmaschinen

Dies reicht für ein Grundverständnis.
Weitere Informationen zur DTD folgt im Kapitel XML ☺

Elemente und Tags

- Ein Tag ist eine Auszeichnung (und wird in Auszeichnungssprachen benutzt ☺) eines Text- bzw. Datenbereichs

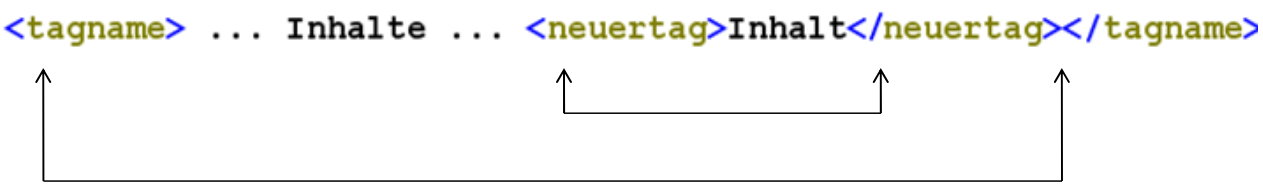


- Tags müssen geschlossen werden (einige erlauben eine Kurzschreibweise)
- Öffnender + schließender Tag bilden ein Element
- Tags werden kleingeschrieben (stricte Anwendung)

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Elemente und Tags

- Tags dürfen geschachtelt sein
- Tags müssen ebenentreu - paarig geschachtelt sein



- Unterscheidung in
  - Block-Level-Elemente: führen zu einen automatischen Zeilenumbruch am Ende, z.B. Überschrift
  - Inline-Elemente: zeichnen Teile von Inhalte aus, kein Zeilenumbruch, z.B. Fettdruck

Enthalten Elemente Unterelemente, so spricht man von „Eltern-“ bzw. „Kindelementen“.

Nicht jedes Element darf zu jedem anderen Element ein Kindelement sein.

Nicht alle Elemente dürfen Unterelemente enthalten, andere benötigen zwingend welche.

Attribute

- Attribute (Eigenschaften) spezifizieren Tags näher

```
<tagname attribut="wert"> ... Inhalte ... </tagname>
```

- 3 Wertarten für Attribute:
  - Normiert
    `<p align="right">` rechtsbündiger Absatz
  - Zahlenwerte (prozentual oder absolut)
    `<table width="600px">` Tabelle, Breite von 600 Pixel
    `<table width="50%">` Tabelle, 50% der Bildschirmbreite
  - Variable Werte
    `<img src="bild1.jpg">` eingefügtes Bild

Manche Tags benötigen zwingend wenigstens ein Attribut.
Beispiel?

Grundgerüst eines HTML Dokumentes

- Umschließendes Element `<html></html>`
- Kopfbereich `<head></head>`
  - Enthält Informationen zu Autor, Beschreibungen, weitere Dateien
  - Nicht sichtbar im Browser, (außer `<title>`)
  - Title Element ist zwingend (auch wenn es leer bleibt)
- Rumpf `<body></body>`
  - Eigentlicher Inhalt, der strukturiert werden kann
  - Wird im Browser angezeigt

```
<html>

<head>
<title> Hallo </title>
</head>

<body>
Auf unserer tollen
Website finden Sie
alle Neuigkeiten.
</body>

</html>
```

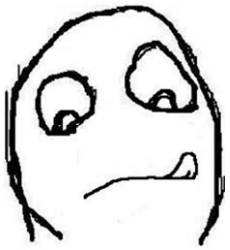

<html> erlaub nur zwei Kindelemente: <head> und <body>.
<head> ist das Elternelement von <title>.
<title> ist das verpflichtende Kindelement von <head>, auch wenn es leer bleibt.

Nicht jedes Element darf zu jedem anderen Element ein Kindelement sein.

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Zusammenfassende Fragen: Abschnitt 2.1

- Was ist eine Auszeichnungssprache?
- Wie spielen HTML und CSS zusammen?
- Was sind Tags und Attribute?
- Wie werden Attribute eingesetzt?
- Wie sieht das Grundgerüst einer HTML Datei aus?
- Wo befindet sich der Title Tag



Kapitel 2
HTML

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Sonderzeichen

- Sonderzeichen sind länderspezifisch und werden daher gesondert notiert

```
&zeichen;
```

- Beispiele:

Ä	Ä	ß	ß
ä	ä	©	©
<	<	€	€
>	>		

- Sonderzeichen wie Umlaute werden aber auch oft vom Browser automatisch erkannt oder die benötigte Zeichencodierung in einer Angabe zu Beginn des Dokuments mitgeteilt*.

*in einem sog. meta tag.
Siehe eigene Recherche im Praktikum.

Fließtext

- Mehrere Leerzeichen werden im Browser zu einem Leerzeichen zusammengesetzt
- Ein Zeilenumbruch im Quelltext wird im Browser zu einem Leerzeichen
- Eine neue Zeile wird durch Block-Level-Elemente erzwungen oder dem Element `<br />` ← kurze Endung
- Leerzeichen können erzwungen werden durch das Sonderzeichen ` `

Textauszeichnungen (Auswahl)

- Textauszeichnungen dienen der Formatierung einzelner Textfragmente
- Definiert das visuelle Erscheinungsbild und kann auch durch CSS erreicht werden

Format	engl. Bez.	HTML-Code	Beispiel
fett	bold	`<b>fett</b>`	**fett**
kursiv	italic	`<i>kursiv</i>`	*kursiv*
gesperrt	teletype	`<tt>gesperrt</tt>`	`gesperrt`
unterstrichen	underline	`<u>unterstrich</u>`	unterstrich
durchgestrichen	strike	`<strike>str</strike>`	~~strike~~
groß	big	`<big>gro&szlig;</big>`	groß
klein	small	`<small>klein</small>`	klein
hochgestellt	superscript	`<sup>hoch</sup>`	hoch
tiefgestellt	subscript	`<sub>tief</sub>`	tief

Überschriften und Absätze

- Überschriften zur Gliederung

```
<h1> … </h1>
<h2> … </h2>
…
<h6> … </h6>
```

Große Überschrift

Kleine Überschrift

- Absätze (engl. paragraph)

```
<p> … </p>
```

neuer Absatz

Ausrichtung durch Attribut align mit den Werten “left”, “center”, “right”

```
<p align=“right”> … </p>
```

Deprecated in HTML 5

Rechtsbündig ausgerichteter Absatz

Deprecated bedeutet „missbilligt“.
Die Attribute funktionieren zwar, validieren aber nicht. Das bedeutet, sie sind nicht mehr korrekter Bestandteil der Sprache.

Viele Attribute aus HTML4 sind deprecated, weil die gleiche Darstellung durch CSS erreicht werden kann und eine noch striktere Trennung zwischen Inhalt und Design angestrebt wird.

Gruppierungen

- Elemente von HTML können in Gruppen zusammengefasst werden. Hierdurch ergeben sich weitere Optionen hinsichtlich der Darstellung/Layout durch CSS.

- Container (division)
  `<div id="header"> … </div>`

- Inline-Block
  ```
  Dies ist <span style="color:red"> roter Text </span>
  gewesen
  ```
  ⟶ Dies ist roter Text gewesen

→ `<div>` ist ein Blocklevel-Element, `<span>` ein Inline Element

Desweiteren bietet HTML5 zahlreiche neue Elemente zur Strukturierung einer Webseite wie
- header
- footer
- summary
- section
- article

Diese bieten, genau wie oder , ohne CSS zunächst keine optischen Veränderungen auf der Seite

Kommentare

- Dienen zur Erläuterung des HTML-Codes
  ```
  <!-- Kommentar -->
  ```
- Werden vom Browser nicht angezeigt
- Beispiel
  ```
  <!-- Formular zur Dateneingabe -->
  ```
- Warum sollte man seinen eigenen Code erläutern?
  Den hab ich ja schließlich selbst geschrieben ☺

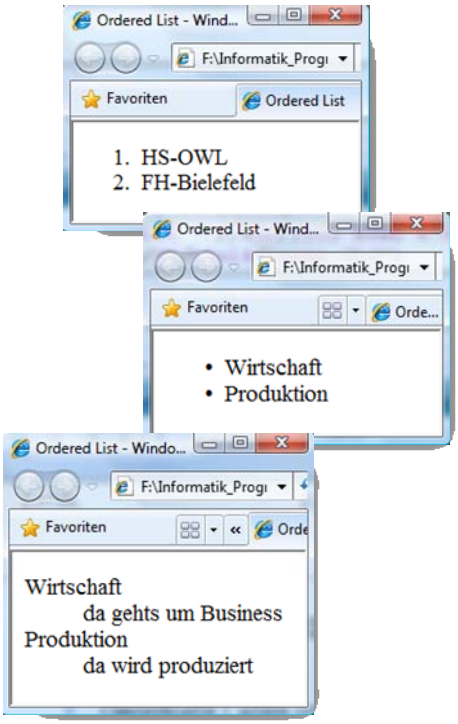
-

-

-

Listen

- Es gibt drei Arten von Listen:
  - Geordnete Listen (ordered lists): `<ol></ol>`
    Nummerierung durch Zahlen oder Buchstaben
  - Ungeordnete Listen (unordered lists): `<ul></ul>`
    Hervorhebung durch besondere Zeichen
  - Definitionslisten (definition list): `<dl></dl>`
    Hervorhebung der Begriffe wie in einem Glossar

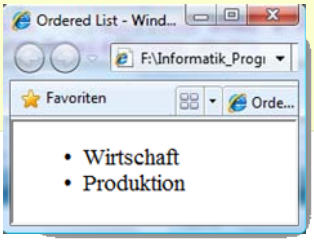


Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Aufbau von Listen

- Liste wird eröffnet:
  `<ul type="disc"> …`
- Jeder Listeneintrag wird gesondert ausgezeichnet: `<li> … </li>`
- Listeneinträge können Attribute bekommen:
  `<li type="disc"> … </li>`
- Liste wird geschlossen:
  `… </ul>`
- Listen dürfen geschachtelt sein

```
<ul type=„disc“>

<li>
Wirtschaft
</li>

<li>
Produktion
</li>

</ul>
```



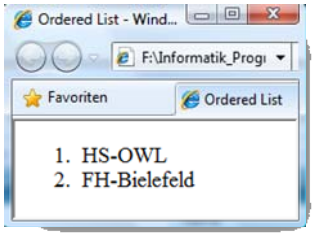
Die Attribute sind in HTML5 deprecated → CSS

Arten:

Schachtelung:

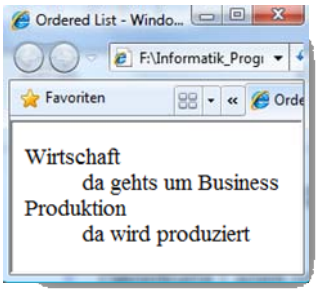
Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Aufbau von Listen




```
<ol>
    <li>HS-OWL</li>
    <li>FH-Bielefeld</li>
</ol>
```


```
<ul>
    <li>Wirtschaft</li>
    <li>Produktion</li>
</ul>
```




```
<dl>
    <dt>Wirtschaft</dt>
    <dd>da gehts um Business</dd>
    <dt>Produktion</dt>
    <dd>da wird produziert</dd>
</dl>
```

Einsatz von Listen

- Listen geben kurz und prägnant Inhalte wider
  - wichtig für die Lesbarkeit
  - wichtig für Suchmaschinen

- Listen können mit CSS vielfältig formatiert werden
  - z.B. Hintergründe
- Dienen häufig als Strukturelement für Menüführung

Tabellen

- Tabellen dienen zur tabellarischen Darstellung von Inhalten wie bspw. von Datenbanken
- Tabellen (z.B. ohne Ränder) können auch Webseiten strukturieren

	Feld	Typ	Kollation	Attribute	Null
	id	int(11)			Nein
	name	varchar(50)	utf8_general_ci		Nein
	link	varchar(255)	utf8_general_ci		Nein
	menuid	int(11)		UNSIGNED	Nein
	parent	int(11)		UNSIGNED	Nein
	admin_menu_link	varchar(255)	utf8_general_ci		Nein
	admin_menu_alt	varchar(255)	utf8_general_ci		Nein
	option	varchar(50)	utf8_general_ci		Nein
	ordering	int(11)			Nein
	admin_menu_img	varchar(255)	utf8_general_ci		Nein
	iscore	tinyint(4)			Nein
	params	text	utf8_general_ci		Nein
	enabled	tinyint(4)			Nein

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Aufbau von Tabellen

Tabellen werden **zeilenweise** definiert!

- Beginn der Tabelle:
  `<table> …`
- Tabellenzeile (table row)
  durch: `<tr> … </tr>`
- Tabellenzelle (table definition)
  durch: `<td> … </td>`
- Tabellenende durch
  `… </table>`

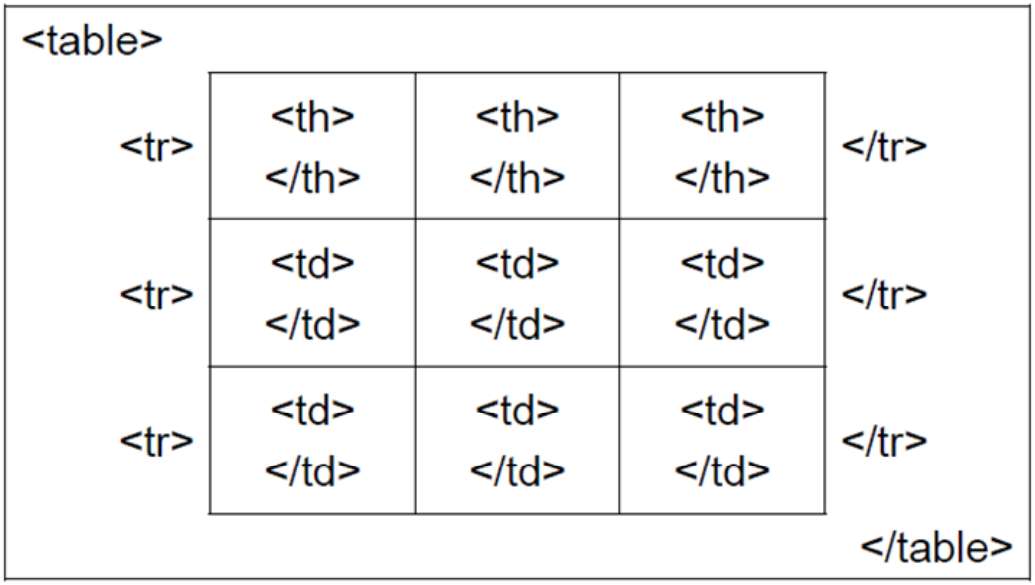
```
<table>

  <tr>
     <td>Wirtschaft</td>
     <td>Produktion</td>
  </tr>

</table>
```

! Ohne Attribut “border“ leider unsichtbare Tabelle

durch den Tag <th> (table head) anstatt von <tr> erreicht man fetten und zentrierten Zelleninhalt.

Aufbau von Tabellen

Tabellen werden zeilenweise definiert!



nach S. Münz: SELFHTML

Gestaltung von Tabellen durch Attribute

- Zusammenfassung von Zeilen (rows)

```
<td rowspan="2"> … </td>
```

Zeile erstreckt sich über 2 Zeilen

<table>
<tr><td rowspan="2">zeile1 zelle1</td><td>zeile1 zelle2</td><td>zeile1 zelle3</td></tr>
<tr><td>zeile2 zelle1</td><td>zeile2 zelle2</td></tr>
</table>

- Zusammenfassung von Spalten (columns)

```
<td colspan="2"> … </td>
```

Zeile erstreckt sich über 2 Spalten

<table>
<tr><td colspan="2">zeile1 zelle1</td><td>zeile1 zelle2</td></tr>
<tr><td>zeile2 zelle1</td><td>zeile2 zelle2</td><td>zeile2 zelle3</td></tr>
</table>

Gestaltung von Tabellen durch Attribute

- Gestaltung der Tabelle
  - Breite der Tabelle `<table width="600px">`
  - Rahmen `<table border="1">`
  - Ausrichtung `<table align="center">`
  - Abstände der Zellen `<table cellspacing="3">`
- Gestaltung von Zellen
  - Horizontale Ausrichtung `<td align="left">`
  - Vertikale Ausrichtung `<td valign="top">`
  - Breite `<td width="20%">`
  - Höhe `<td height="25px">`

Attribute teilweise wieder deprecated

Zusammenfassende Fragen: Abschnitt 2.2

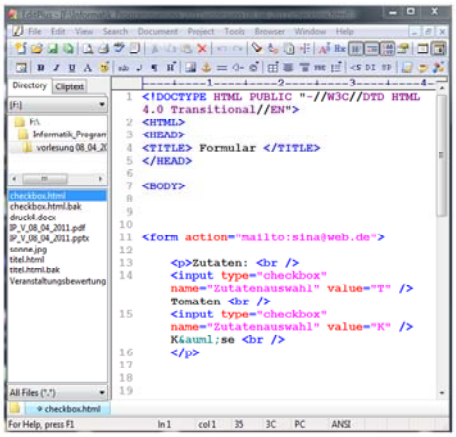
- Wie werden Sonderzeichen notiert?
- Wie können Leerzeichen dargestellt werden?
- Wie könnten drei versch. Textauszeichnungen aussehen?
- Welche Arten von Listen kennen Sie und wie ist der Aufbau?
- Wie ist der Aufbau von Tabellen?
- Wie werden Zeilen und Spalten von Tabellen zusammengefasst?

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Organisatorisches

Tip:

- Validieren des Dokumentes unter http://validator.w3.org/
- Shortcuts vereinfachen das Arbeiten erheblich



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML
4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> Formular </TITLE>
</HEAD>

<BODY>



<form action="mailto:sina@web.de">

    <p>Zutaten: <br />
    <input type="checkbox"
    name="Zutatenauswahl" value="T" />
    Tomaten <br />
    <input type="checkbox"
    name="Zutatenauswahl" value="K" />
    K&auml;se <br />
    </p>
```

Strg + s → speichern
Strg + b → Ansicht
Strg + z → Rückgängig
Strg + c → Kopieren
Strg + v → Einfügen

F5 → Seite neu laden

Kapitel 2
HTML

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Generelles zu Grafiken

- zur Ansicht im Browser gebräuchliche Web-Formate verwenden
  - png, jpg, gif
- Größe der Dateien beachten
  - Möglichst unter 1 MB

- dringend zu beachten:

Wir erinnern uns: Leerzeichen sollten wir im Dateinamen vermeiden!!

Was sind die Unterschiede in den Formaten?

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Einbinden von Grafiken

- Grafiken werden durch das Tag **img** (image) und dem Attribut **src** (source) eingebunden

```
<img src="bilder/kuh.jpg" />
```

kurze Endung

- Weitere Attribute
  - `alt` Alternativer Text
  - `title` Titel des Bildes
  - `border, width, height` Rahmen, Breite, Höhe

```
<img src„bilder/kuh.jpg" width="200px" height="150px"
alt="Kuh!" title="Eine Kuh auf der Weide" />
```

→Skalieren (=vergrößern oder verkleinern) sowie Verzerren ist möglich, aber ungeschickt.

Das Attribut src ist natürlich verpflichtend.
Die Attribute width und height können angegeben werden, müssen aber nicht. Der Browser erkennt die Größe, braucht dafür aber länger zum laden der Webseite, da erst nach dem Laden des Bildes klar ist, wieviel Platz denn reserviert werden muss.
Die Attribute erlauben auch ein Vergrößern (sieht dann pixelig aus) oder Verkleinern (verschwendet einfach nur Bandbreite) der Bilder. Sie können das Bild auch verzerren wenn gewünscht.

Wozu dienen die Attribute alt und title?

Einbinden von Grafiken

- Relativ: Verlinkung im Verzeichnisbaum

```
<img src="../layout/bilder/kuh.jpg" />
```

- Absolut: Verlinkung im Web

```
<img src="http://www.hs-owl.de/fb7/kuh.jpg" />
```

Einbinden von Hyperlinks

- Ein Hyperlink ist ein Querverweis auf
  - ein HTML-Dokument (oder eine Stelle in einem HTML Dokument)
  - ein Dokument anderen Formats (z.B. pdf)

- Einbindung durch das tag a (anchor) und dem Attribut **href** (hyperreference)

```
<a href="stundenplan.html"> Linktext </a>
```

- Relative oder absolute Verlinkung
  (siehe Einbindung Grafiken)

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Weitere Verwendung von Links

- Einsatz von Sprungmarken
  - Vordefiniert:
    ```
    <a href="#top"> hier geht's nach oben </a>
    ```
  - Selbstdefiniert:
    ```
    <h1 id="aufg2"> Aufgabe 2</h1>

    <a href="praktikum.html#aufg2">Aufgabe 2</a>
    ```

- Aktivierung des Email Programms

```
<a href="mailto:malte.wattenberg@hs-owl.de">Schreiben
Sie mir eine Mail!</a>
```

- Bild als Hyperlink

```
<a href="http://www.hs-owl.de/fb7/"><img src="logo.png" /></a>
```

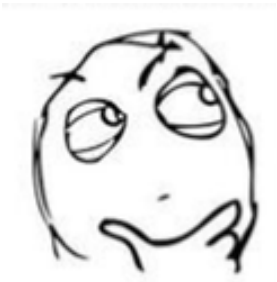
- Target-Attribut: `<a href="http://www.lemgo.de" target="_blank">`

Andere vordefinierte

Zusammenfassende Fragen: Abschnitt 2.3 + 2.4

- Was ist hinsichtlich Grafiken im Netz zu beachten?
- Wie werden Grafiken eingebunden? Bsp.?
- Wozu dienen das title und alt Attribut bei Grafiken?

- Wie können Hyperlinks eingebunden werden?
- Wie kann innerhalb von Dokumenten verlinkt werden?
- Wie können Grafiken als Link fungieren?

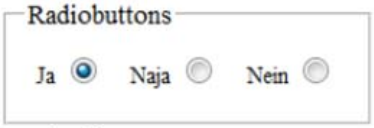


Kapitel 2
HTML

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Formulare

- Formulare dienen der Eingabe von Informationen durch einen Benutzer und deren Weitergabe
- Elemente sind
  - Eingabefelder
  - Buttons
  - Checkboxen/Auswahllisten
  - Schaltflächen



Genereller Aufbau:

```
<form name="Anmeldung" action="mailto:sina@web.de">
     <!– hier sind die einzelnen Elemente-->
</form>
```

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Eingabefelder

- Einzeilig

```
<input type="text" name="Vorname" />
```

Kurze Endung

Wichtige Attribute:

`size`	Breite des Feldes
`maxlength`	maximale Eingabelänge
`value`	vorbelegter Wert

- Mehrzeilig

```
<textarea name="Eingabe" cols="20" rows="50"></textarea>
```

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Buttons

- Submit Button

  - Löst die im Formularkopf definierte Aktion aus, z.B. Mailversand

```
<input type="submit" name="Bestellung" value="absenden" />
```

- Reset Button

  - Löscht alle eingegebenen Inhalte

```
<input type="reset" name="loesche" value="zurücksetzen" />
```

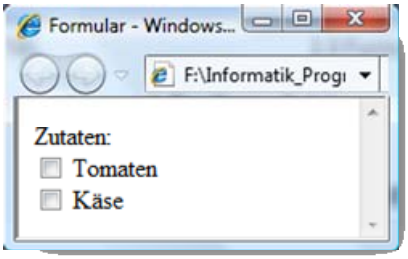
- Selbstdefinierte Buttons
  - Führt bestimmte Funktion aus, meist JavaScript

```
<button name="Berechnung" type="button"
        onclick="addiere();" value="abschicken">
        <img src="klick.jpg" />
</button>
```

Kontroll- und Optionsfelder

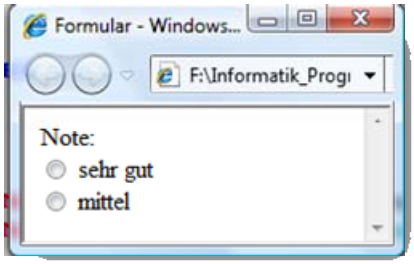
- Kontrollfeld (checkbox)
  - Mehrere Elemente können ausgewählt werden

```
<input  type="checkbox"
        name="Zutatenauswahl"
        value="T" />
```



- Optionsfeld (radiobutton)
  - Nur ein Element kann ausgewählt werden

```
<input  type="radio"
        name="Note"
        value="1" />
```

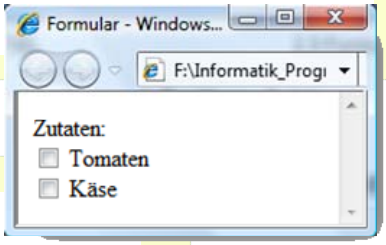


Value ist der Wert, der an den Server übergeben wird

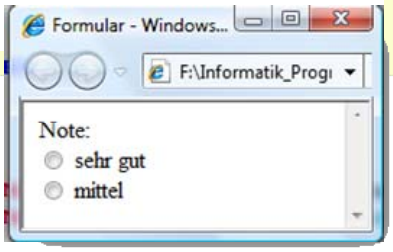
Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Vollständiges Beispiel: Kontroll- und Optionsfelder

```
<form action="mailto:sina@web.de">

    <p>Zutaten: <br />
    <input type="checkbox" name="Zutatenauswahl" value="T" /> Tomaten   <br />
    <input type="checkbox" name="Zutatenauswahl" value="K" /> K&auml;se <br />
    </p>

</form>
```

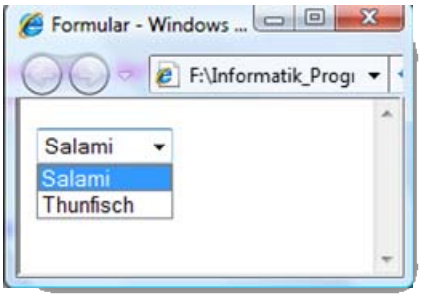



```
<form action="mailto:sina@web.de">

    <p>Note: <br />
    <input type="radio" name="Note" value="1" /> sehr gut <br />
    <input type="radio" name="Note" value="2" /> mittel   <br />
    </p>

</form>
```



Auswahllisten

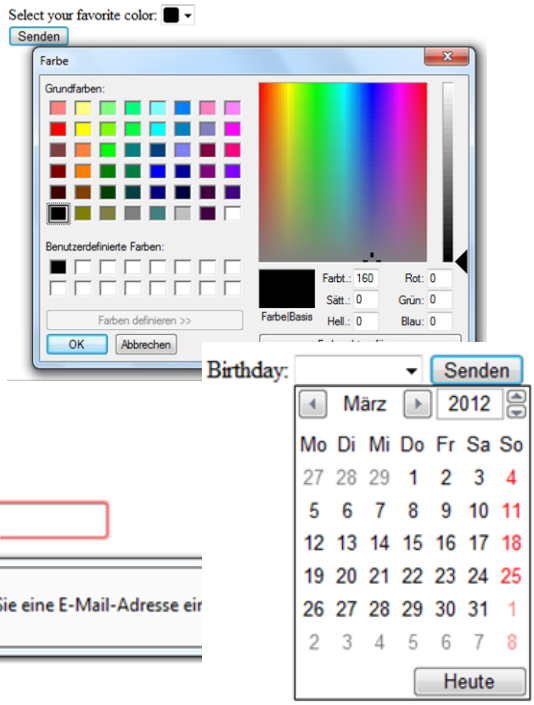
- Auswahl von einem oder mehreren Werten aus einer Menge von vordefinierten Werten
- Genereller Aufbau:

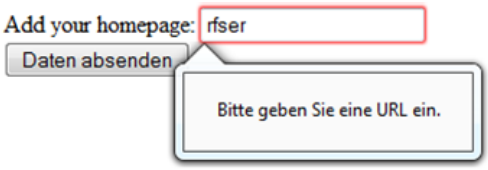



```
<select name="Pizza">
        <option value="Sal"> Salami </option>
        <option value="Thu"> Thunfisch </option>
</select>
```

Neue Formularattribute in HTML5

- Neue input types, z.B.
  - Colorpicker: color
  - Emaileingabe: email
  - Datumseingabe: date
  - Telefonnummer: tel
  - Zahlen: number
  - Webadresse: url





Html5test.com

Zusammenfassende Fragen: Abschnitt 2.5

- Wozu dienen Formulare?
- Wie ist der grundlegende Aufbau von Formularen?
- Welche Elemente gibt es?
- Was ist der Unterschied zw. einem Radiobutton und einer Checkbox?
- Was bedeutet das Attribut value?
- Welche Attributwerte sind neu in HTML5 ?

Kapitel 2
HTML

Einbindung von Objekten

- Beliebige Objekte (Bilder, Video, Audio, Java-Applets etc.) können durch <object> eingebunden werden
- Wenn der Browser das Objekt kennt stellt er es direkt dar, andernfalls wird ein extra PlugIn benötigt (z.B. Flash)
- HTML 5 kennt die Tags <audio> und <video>

```
<video width="640px" height="480px"
       src="videodatei.mp4"></video>
```

- Browserkompatibilität noch sehr eingeschränkt → Testen Sie Ihren unter http://html5test.com/

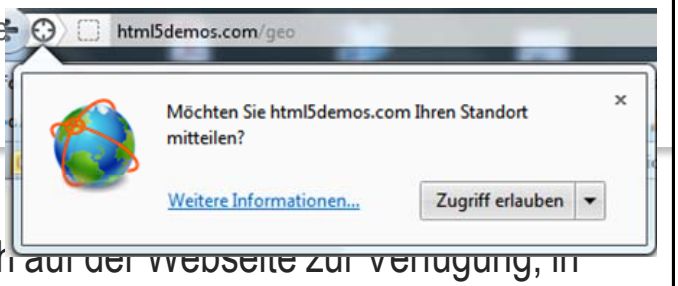
Einbindung von Objekten

- Andere HTML-Dokumente werden über <iframe> (integrated frame=Rahmen) in das eigene Dokument eingebunden

```
<iframe title="Filmtest" width="480" height="390"
src="http://www.malte-wattenberg.de/"></iframe>
```

- Einsatz häufig anzutreffen zum Bereitstellen externer Funktionalitäten (PlugIns, Widgets, …)
wie bspw. Facebook „Like-Box“, Wetter, Youtube-Videos

Neue Funktionalitäten in HTML5



- Canvas:
  - Element <canvas> stellt einen Bereich auf der Webseite zur Verfügung, in dem zur Laufzeit über ein Script Vektorgrafiken gezeichnet werden

```
<canvas id="myCanvas" width="200px" height="100px"></canvas>
```

- Geolocation:
  - Geolocation berechnet die Position aufgrund von GPS Daten (sofern vorhanden), oder IP Auswertungen über einen Google Service.

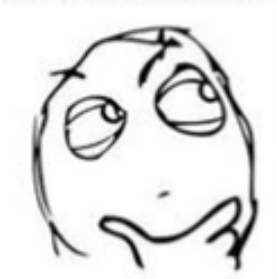
- Local storage:
  - Lokale Speicherung von Daten beim Anwender (ähnlich Cookies, jedoch deutlich mehr Inhalt), auf die Webanwendungen zugreifen können
  - Cookies laufen zeitlich ab, lokal gespeicherte Daten nicht
  - Zugriff auf Datenobjekt über JavaScript

Diskussion:
Welche Nachteile sind bei Geolocation und local storage denkbar?

Demos:
- http://www.html5canvastutorials.com/
- http://html5demos.com/geo

Zusammenfassende Fragen: Abschnitt 2.6

- Wie können externe Elemente in HTML eingebunden werden?
- Welche Beispiele für externe Inhalte können Sie nennen?
- Was ist Local Storage?



Literatur

- Meyer; CSS kurz und gut; O'Reilly; 2011
- Laborenz; CSS: Das umfassende Handbuch; Galileo Computing; 2011
- RRZN; HTML 4 Zusatzwissen
- http://de.selfhtml.org
- http://www.css4you.de/

Kapitel 3
CSS

Motivation

- Bisher: Zahlreiche Tags und Attribute in HTML zur Gestaltung des Inhalts vorhanden

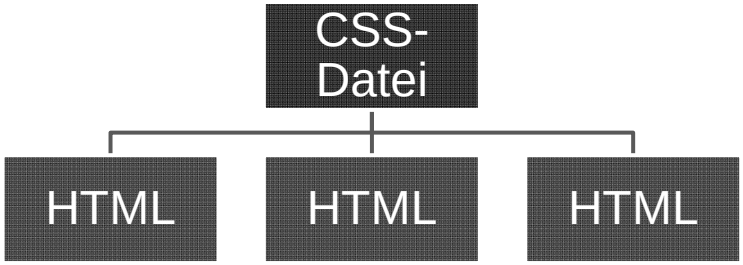
```
<table bgcolor="red">
  <tr><td><u>Hintergrund</u></td></tr>
</table>
```

- Problem:
  - Browser stellen HTML-Dokumente unterschiedlich dar
  - Attribute für die Gestaltung nicht immer ausreichend
  - Aufgeblähter Code, schlecht lesbar bei vielen Einträgen

- Deshalb:
  - Trennung von Inhalt und Layout/Darstellung der Webseite
  - Entwicklung einer eigenen Sprache → CSS (Cascading Style Sheets)

Motivation

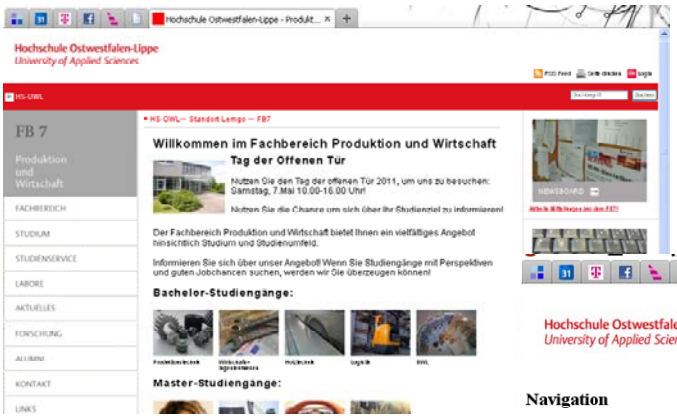
- Ergebnis:
  - Einheitliche Darstellung
  - Durch zentrale Haltung kann ein Layout kann für beliebig viele Seiten gelten
  - Mehr Möglichkeiten als in HTML
  - Ausgabe je nach Medium unterschiedlich (Smartphone, Screenreader, Print, …) Bei einer flexiblen Änderung je nach Bildschirmgröße spricht man häufig von „Responsive Webdesign“.
  - Mehrere Layouts können für ein und denselben Inhalt festgelegt werden
  - Aktuelle Version: CSS 3; CSS 2.1 wird weitgehend unterstützt



Geringe Unterschiede gibt es immer noch, da z.B. Schriften durch die Browser unterschiedlich gerendert (dh. die visuelle Repräsentation) werden.
Ebenso unterstützen einige Browser bspw. nicht alle CSS 3 bzw. 2.1 Regeln oder enthalten schlichtweg Fehler – einsamer Spitzenreiter seit vielen Jahren ist dabei der Internetexplorer von Microsoft.

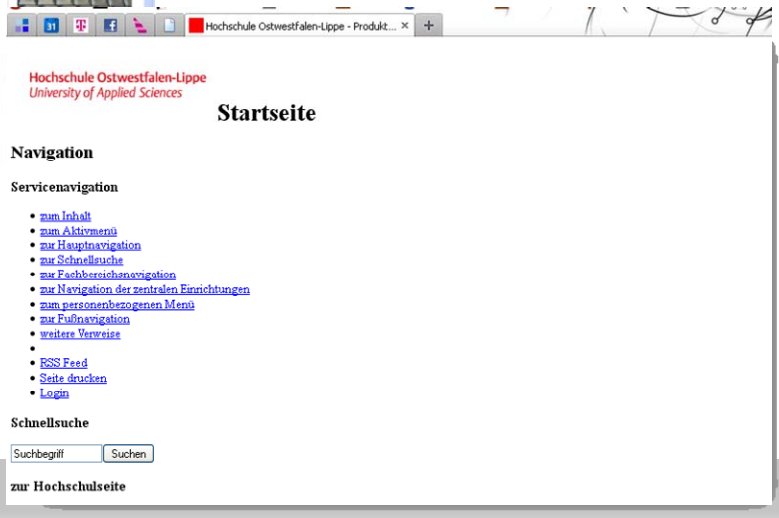
Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Beispiel



← HS-OWL, normale Ansicht

HS-OWL, Ansicht ohne CSS →



Kapitel 3
CSS

Drei Möglichkeiten der Einbindung

- 1. Style Definition in einem Tag

```
<p style="color:red;
          font-size:22px">CSS im Tag! </p>
```

CSS im Tag!

  - Attribut **style** mit CSS Angaben direkt im Tag von HTML
  - Schnellste und einfachste Lösung, aber
  - nur in diesem Tag in dieser Datei gültig

Drei Möglichkeiten der Einbindung

- 2. Style Definition im Kopfbereich der Datei

```
<html>
 <head>
  <title> Style im head </title>
<style type="text/css">
p {
    color: green;
    font-size: 32pt;
}
</style>
</head>
<body>
  <p>Style im HEAD</p>
</body>
</html>
```

Style im HEAD

- Einbindung durch den Tag style
- Gültig für das gesamte Dokument

Drei Möglichkeiten der Einbindung

- 3. Style Definition in einer separaten Datei

  - Zunächst Erstellung einer neuen Datei mit der Dateiendung:

    .css

    und CSS - Regeln

```
h1 {
    background-color:green;
    font-size:14px;
    color:yellow;
    }
```

cssBeispiel.css

Drei Möglichkeiten der Einbindung

- 3. Style Definition ein einer separaten Datei

```
<html>

 <head>
  <title> Style in separater Datei </title>

<link rel=stylesheet type="text/css"
href="cssBeispiel.css" />

</head>

<body>
  <h1>Willkommen bei uns</h1>
</body>

</html>
```

- Einbindung durch den Tag **link** im head
- Gültig für das gesamte Dokument, bei Einbindung auf allen Seiten gültig für gesamten Webauftritt

Zusammenfassende Fragen: Abschnitt 3.1 und 3.2

- Wozu dient CSS?
- Was wird durch CSS erreicht?

- Mit welchen drei Möglichkeiten kann CSS eingebunden werden?
- Was sind jeweils die Vor- bzw. Nachteile?
- Wenn mehr als eine Möglichkeit genutzt wird, welche ist gültig? (Antwort im/durch Praktikum)

Kapitel 3
CSS

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Syntax der CSS Regeln

Allgemein

```
selector {
      property1: value1;
      ...;
      propertyN: valueN;
      }
```

Beispiel

```
body {
  color: white;
  ...;
  background-color: black;
  }
```

- Selector: Auswahl der gewünschten Elemente, z.B. (p, body, h1, table, …) oder Universalselektor *
- Property: festzulegende Eigenschaft, z.B. (color, border-style, …)
- Value: Eigenschaftswert (red, 20pt, …)

Selektoren können zusammengefasst werden: s1,s2,s3 {p1:v1;}

Bsp. für eine Zusammenfassung:

```
p, h1, h2, ul {
        color: blue}
```

Hier: Alle Paragraphen, Überschriften des Grades 1 und 2 sowie ungeordnete Listen haben eine blaue Schriftfarbe

Hinweis:
Enthält ein selector nur eine Eigenschaft oder handelt es sich um die letzte Eigenschaft von vielen, kann das Semikolon weggelassen werden.
Zur besseren Bearbeitbarkeit (Erweiterung, Vergesslichkeit) empfiehlt sich dies in der Regel nicht.

Klassenselektor

- Bisher: Durch Selektoren sehen alle Elemente gleich aus
- Nun: Bildung von Klassen. Somit können gleiche Elemente unterschiedliche Layouts bekommen

- Bsp.:

Absatz auf grünem Grund

Absatz auf rotem Grund

Klassenselektor

- Schritt 1: Definition:

```
p.gruenerabsatz {background-color: green}
p.roterabsatz {background-color: red}
```

- Schritt 2: Aufruf:

```
<p class="gruenerabsatz"> Hallo </p>
```

Hallo

```
<p class="roterabsatz"> Welt </p>
```

Welt

Diese Klasse gelten in diesem Fall ausschließlich für Paragraphen.

Es ist aber auch möglich, die Klassen allgemein zu definieren. Dazu wird das Element nicht näher beschrieben (weggelassen). Die Klasse beginnt dann einfach mit dem Punkt. Bsp:

.gruenerabsatz {color:blue}

Jetzt kann diese Klasse auch von anderen Elementen aufgerufen werden. Bsp.:

<h1 class="gruenerabsatz">Hallo</h1>

Weitere Selektoren

- ID-Selektor: für genau ein HTML Element wird eine CSS-Regel erstellt

```
#footer {
    height:60px; }
```

gilt für :

```
<div id="footer">Impressum</div>
```

- Pseudoklassenselektor: Der Klassenname kommt nicht in den Auszeichnungen vor (=Pseudo)

```
a:hover {
    color:blue; }
```

Möglich sind auch Kombinatoren:

```
/* Nachbarkombinator*/

h1+p {
    color:blue; }
```

```
/* Nachfahrkombinator*/

li p {
    color:blue; }
```

Weitere Kombinationen zum Nachlesen unter http://jendryschik.de/wsdev/einfuehrung/css/selektoren

Ebenfalls:

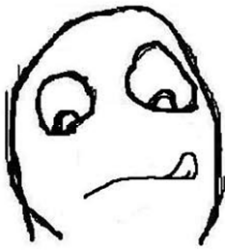
:hover überfahren mit der Maus
:visited besuchte links
:link unbesuchte Links

Nachbarkombinator: hier: ein Paragraph folgt direkt auf eine Überschrift ersten Grades
Nachfahrkombinator: hier: Alle Paragraphen in einem Listenpunkt

Sie sehen auf der Folie ebenfalls wie Kommentare in CSS erstellt werden.
Eine gesonderte Notation für einzeilige Kommentare entfällt.

Zusammenfassende Fragen: Abschnitt 3.3

- Was ist der Unterschied zw. Selector, property, value?
- Wozu dienen Klassen?
- Wie werden Klassen erstellt? Bsp.?
- Wozu dienen ID-Selektoren?
- Was sind Pseudoselektoren



Kapitel 3
CSS

Farben und Hintergründe

- Farben können hexadezimal oder als RGB-Werte angegeben werden

```
P { color: #ff0000; }
```

```
p { color: rgb(128,128,255); }
```

→16,7 Mio. Farben möglich

- Es gibt eine Reihe von vordefinierten Farben (blue, red, green, black, …)

```
P { color: yellow; }
```

- Hintergründe werden durch die Eigenschaft background-color beschrieben. Es können auch Bilder verwendet werden:

```
div.blau {
    background-image:url("hintergrund.jpg");
    background-repeat: repeat-x;}
```


Auch die hexadezimale Darstellung beinhaltet 3 Paare mit Werten zu rot, grün, blau.
So bedeutet #ffff00 max. rot, max. grün, kein blau und entspricht demnach der Farbe:

Schriften und Listen

- Schriften: Eigenschaften, die u.a. eine Schriftformatierung erlauben:

`font-family:`	Schriftart
`font-size:`	Schriftgröße
`font-weight:`	Schriftstärke
`font-style:`	Schriftstil (z.B. italic)

- Listen: das bisherige type Attribute ist deprecated. CSS bietet u.a. die Eigenschaften:

`list-style-type`	disc, circle, lower-roman etc…
`list-style-image`	Angabe einer url

Hyperlinks

- Darstellung von Hyperlinks - Festlegung in CSS

```
a:link { color:blue;}
a:visited { color:green;}
a:active { color:red;}
a:hover { color:yellow;}
a:focus { color:black;}
```

normaler Link
bereits besucht
gerade angeklickt
unter dem Mauszeiger
mit Tastatur angewählt

Ränder

- Ränder können über die Eigenschaft **border** beschrieben werden

```
border-width: 3px;
border-color: #ccc;
border-style: solid;
```

(solid, dotted, dashed, double, groove, …)

- Es können Art, Farbe und Dicke für jede Seite einzeln bestimmt werden

```
border-bottom-style: solid;
```

(top, right, bottom, left)

- Kurzform:

```
border: 2px solid black;
```

Abstände von Elementen

- Abstände dienen zur optischen Trennung von Elementen oder den Inhalten zum Element
- Außenabstände werden durch die Eigenschaft margin (engl. Seitenrand, Begrenzung) beschrieben

```
margin-top   margin-right   margin-bottom margin-left
```

- Innenabstände werden durch die Eigenschaft padding (Polsterung) beschrieben. Anwendung analog zu Außenabstand.
- Kurzformen bei Außen- und Innenabstand möglich:

Deklaration	Randabstand			
	oben	rechts	unten	links
`margin: 1em;`	1 em	1 em	1 em	1 em
`margin: 1em 2em;`	1 em	2 em	1 em	2 em
`margin: 1em 2m 3em;`	1 em	2 em	3 em	2 em
`margin: 1em 2em 3em 4em;`	1 em	2 em	3 em	4 em

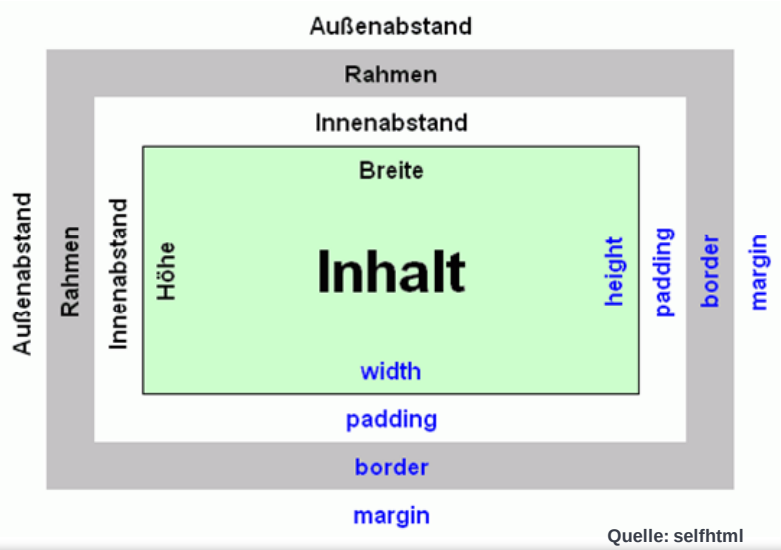
Quelle: http://jendryschik.de/wsdev/einfuehrung/eigenschaften/css-boxmodell/polsterung

Border Box Modell

- Das Box Modell dient zur Berechnung der Breite und Höhe von Elementen
- Die Gesamtbreite eines Elements besteht aus der Addition von Inhalt, Innenabstand, Rahmen und Außenabstand
- Der Außenabstand hat keine Hintergrundfarbe

- Bsp.:

```
#header {
    width: 100px;
    border: 1px solid black;
    padding: 20px;
    margin: 0 30px;}
```



Quelle: selfhtml

Positionierungsmöglichkeit 1: Art und Startpunkt

- Elemente auf der Webseite können durch die Angabe der Positionsart und –startpunkt platziert werden

Bsp. Art:

`position: static;`	Positionierung im normalen Textfluss
`position: relative;`	Positionierung relativ abweichend zum normalen Textfluss
`position: absolute;`	Ausrichtung am Elternelement oder body
`position: fixed;`	wie absolute, Element wird jedoch nicht mitgescrollt

Bsp. Startpunkt:

`top: 100px;`	Startpunkt von oben in Pixeln
`left:`	links
`right:`	rechts
`bottom:`	unten

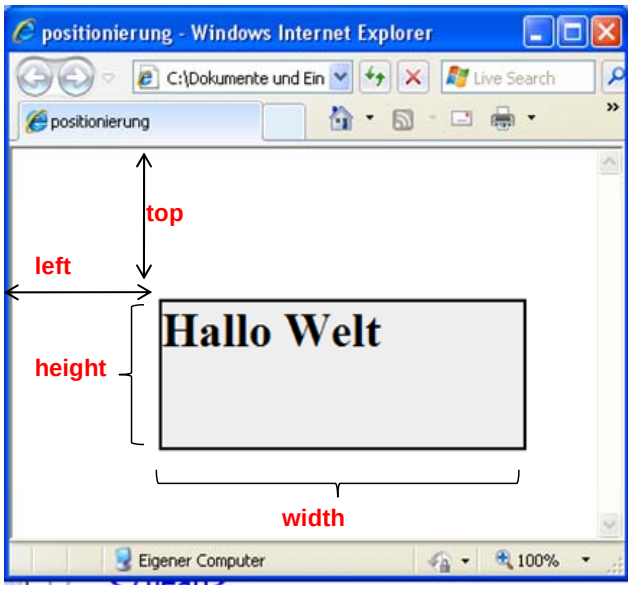
Positionierungsmöglichkeit 1: Größe

- Die Größe der Elemente wird durch width und height angegeben

```
width: 250px;
```
Breite des Elements

```
height: 100px;
```
Höhe des Elements

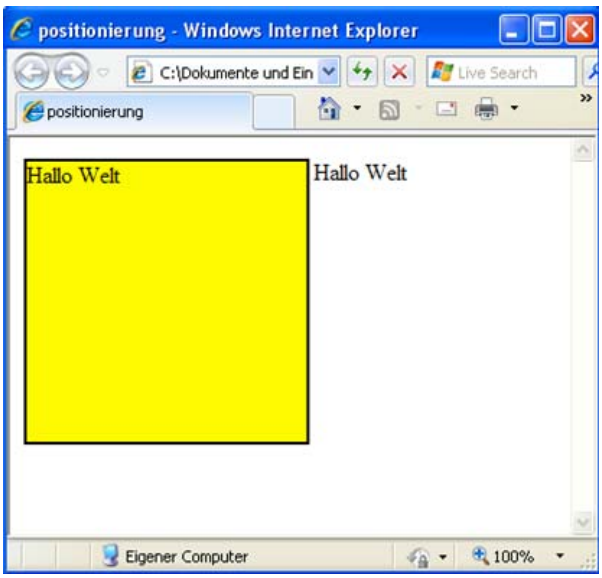
```
<html>
 <head>
  <title>positionierung </title>
 <style type="text/css" >

 .bsp {
    position:absolute;
    top:100px;
    left:100px;
    background-color:#eee;
    border:2px solid #000;
    width:250px;
    height:100px;
 }

 </style>
 </head>

 <body>
    <h1 class="bsp">Hallo Welt</h1>
 </body>
</html>
```



Positionierungsmöglichkeit 2: float

- Die Eigenschaft **float** (left, right) bewirkt, dass ein nachfolgendes Element links oder rechts „herumfliesst“
- clear (both, left, right) hebt diesen Fluss auf

```
<html>
 <head>
  <title>positionierung </title>
 <style type="text/css" >

 .left {
    border:2px solid black;
    float:left;
    height:200px; width:200px;
    background-color:#fcf900;
     }

 </style>
 </head>

 <body>
    <p class="left">Hallo Welt</p>
    <p> Hallo Welt</p>
 </body>
</html>
```



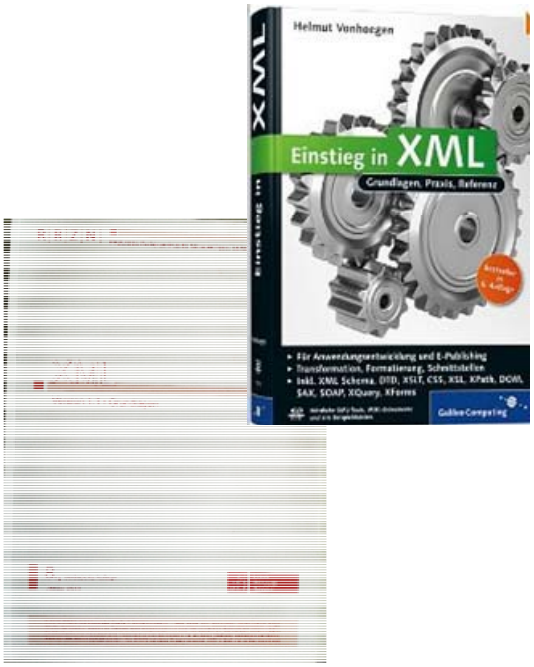
Lernfragen und -aufgaben zu Abschnitt 4.4

- Was kann für Ränder insgesamt bestimmt werden?
- Wie können Innen- und Außenabstände definiert werden?
- Wie ist die Schreibweise „`padding: 20px 10px;`“ zu deuten?
- Zeichnen und erläutern Sie das Border-Box-Modell.
- Erläutern Sie, mit welchen Mitteln Elemente auf einer Webseite positioniert werden können.

Kapitel 1.	Einführung	
Kapitel 2.	HTML	
Kapitel 3.	CSS	
Kapitel 4.	XML	
Kapitel 5.	JavaScript	

Literatur

- Vonhoegen; Einstieg in XML: Grundlagen, Praxis, Referenz Galileo Computing; 2011
- RRZN; XML Grundlagen

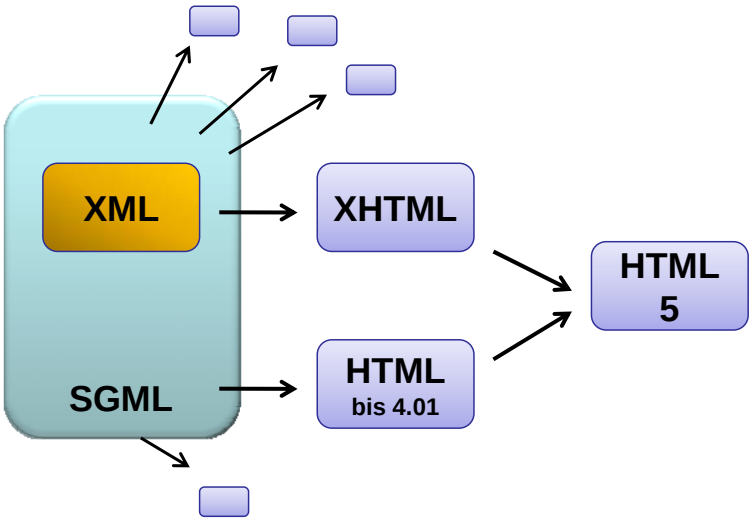


Kapitel 4
XML

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Einordnung

- XML – eXtensible Markup Language – ist eine Meta-Auszeichnungssprache zum Austausch strukturierter Daten in Textform



Metasprachen bieten Möglichkeiten zur Definition von Untersprachen für bestimmte Zwecke

Auszeichnungssprachen strukturieren Dokumente durch Auszeichnungen einzelner Bereiche

Vgl.: Vonhoegen, H.; Einstieg in XML; Bonn; 5. Aufl.; 2009
Vgl.: http://cscie12.dce.harvard.edu/lecture_notes/2010/20100127/slide37.html

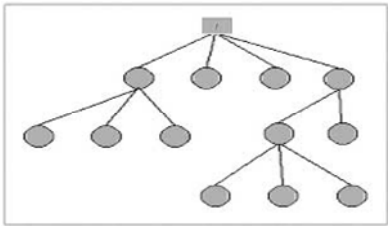
SVG
MATH ML

Kapitel 4
XML

Syntax: Hallo Welt

- XML-Deklaration
- Frei definierbare Tags
  (case sensitive, beginnen mit Buchstaben, „_“ oder „:“)
- Genau ein Wurzelelement
- Start- und End- Tag notwendig
  (bei leeren Elementen auch `< />`)
- Auf korrekte Verschachtelung achten
  (ebenentreu-paarig, vgl. HTML)

```
<?xml version="1.0"?>
<welt>
   <gruss>Hallo Welt</gruss>
</welt>
```



Syntax: Hallo Welt, Version 2

```
<?xml version="1.0"?>
<welt>
   <!-- meine zweite XML-Datei-->
   <gruss id="1">Hallo Welt</gruss>
   <gruss id="2"><![CDATA[<<Wie gehts?>>]]></gruss>
   <gruss />
</welt>
```

- Kommentare durch: `<!-- Kommentar -->`
- Attribute durch: `Attributname="Attribut-Wert"`
- CDATA durch: `<![CDATA[…]]>`
- Verarbeitungsanweisungen möglich, z.B.: `<?php … ?>`

Validität

Wann ist ein XML-Dokument **valide** (=gültig?)

- Es muss **wohlgeformt** sein, d.h. alle Regeln zur korrekten Syntax müssen eingehalten werden.
- Es muss eine **Grammatik** vorhanden sein, definiert durch z.B.
  - DTD (doctype definition)
    oder
  - XML Schema

→ Testen der Validität unter validator.w3.org

Wir haben erfahren: Browser rechnen oft mit fehlerhaftem Quellcode in HTML und versuchen eine Interpretation die oft gelingt. (in der Klausur gelingt das nicht ☺)

Bei XML jedoch führt jeder Fehler zu einem Hinweis und die Datei wird nicht dargestellt.

Schemasprache DTD

Beschreibung der gesamten Struktur und Grammatik des Dokumentes

- Elementtypen und Verschachtelung
- Attribute der Elemente und Attributsinhalte
- Entitäten
- Notationen

Externer Speicherort der Datei
(auch innerhalb der XML-Datei möglich, vgl. CSS)

Optionales Subset

```
<!DOCTYPE welt SYSTEM "datei.dtd" [ ... ]>
<!DOCTYPE welt PUBLIC "Public Identifier" "datei.dtd" [ ... ]>
```

Dokumenttypdeklaration

Wurzelelement

SYSTEM: Speicherort explizit bekannt
PUBLIC: Speicherort durch öff. Bezeichner angegeben

Was sind:
Entitäten?
Notationen?

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Schemasprache DTD

Beispiel innerhalb der XML-Datei:

```
<?xml version="1.0"?>

<!DOCTYPE welt
   [
   <!ELEMENT welt    (gruss*)>
   <!ELEMENT gruss   (#PCDATA)>
   <!ATTLIST gruss
             id  ID  #IMPLIED>
   ]>

<welt>
    <!-- meine dritte XML-Datei-->
    <gruss id="1">Hallo Welt</gruss>
    <gruss id="2"><![CDATA[<<Wie gehts?>>]]></gruss>
    <gruss />
</welt>
```

Syntax

```
<!ELEMENT Elementname
Vorgaben>

<!ATTLIST Elementname
Attributliste Typ Vorgaben>
```

Referenz unter:
de.selfhtml.org/xml/dtd/

?
+
*

Beispiel:
<!ELEMENT adresse (Name, Strasse, Ort, email?)>

Das Element adresse enthält genau ein Element Name, Strasse und Ort. Das Element email kann 0 oder 1 mal vorkommen.
Beachte: Auch die Unterelemente müssen einzeln definiert werden. Bsp: <!ELEMENT Name (#PCDATA)>

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Formatierung durch CSS

Beispiel

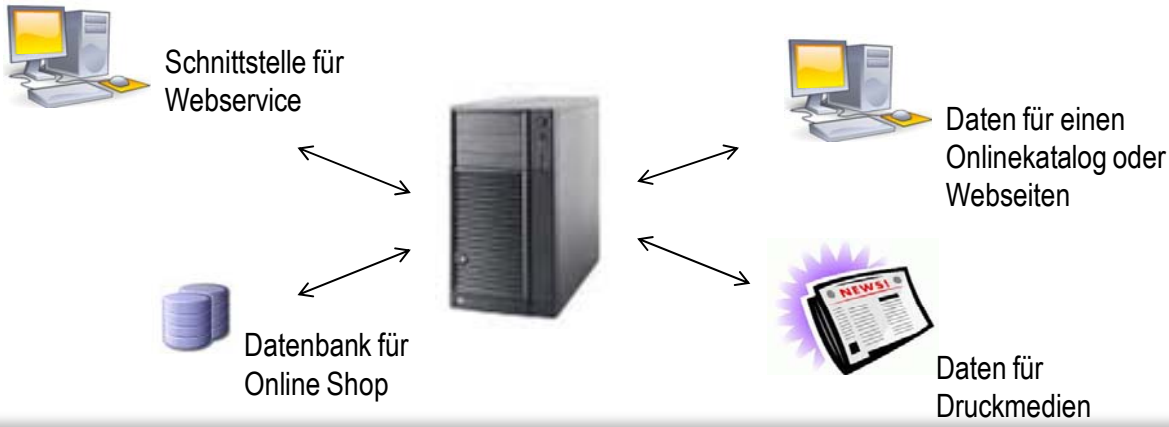
```
<?xml version="1.0"?>

<!DOCTYPE welt
    [
    <!ELEMENT welt    (gruss*)>
    <!ELEMENT gruss   (#PCDATA)>
    <!ATTLIST gruss
          id  ID  #IMPLIED>
    ]>

<?xml-stylesheet type="text/css"
  href="style.css" ?>

<welt>
    <!-- meine dritte XML-Datei-->
    <gruss id="1">Hallo Welt</gruss>
    <gruss id="2"><![CDATA[<<Wie gehts?>>]]></gruss>
    <gruss />
</welt>
```

Anwendung der CSS Regeln auf die XML Elemente analog zu HTML

Kapitel 4
XML

Anwendungsgebiete: Warum XML?

- Speicherung von Daten mit dem Ziel der einfachen Bearbeitung oder eines vereinfachten Austausches
- z.B.: Webservices:
  Softwareanwendung mit XML Schnittstelle (Bsp.: ebay)



Weitere Sprachen zur Verarbeitung

XSL(T):
Style- und Programmiersprache zur Transformation der XML-Dokumente (z.B. XHTML, PDF)

Xpath:
Abfragesprache zur Adressierung einzelner Elemente und Knoten

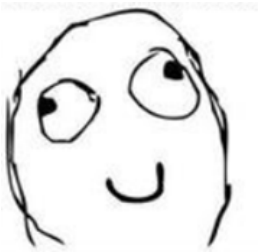
Xlink:
Syntax zum Hinzufügen von Verlinkungen innerhalb des XML-Dokumentes

XSLT-Beispiel:
eingebunden in der XML Datei durch: <?xml-stylesheet type="text/xsl" href="gedicht_xslt.xsl" ?>

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
        <xsl:output encoding="iso-8859-1" />
        <xsl:template match="Gedichtsammlung">
                <html>
                        <body>
                                <xsl:apply-templates/>
                        </body>
                </html>
        </xsl:template>
        <xsl:template match="Autor">
                <br/>
                <h4>
                        <xsl:apply-templates/>
                </h4>
        </xsl:template>
</xsl:stylesheet>
```

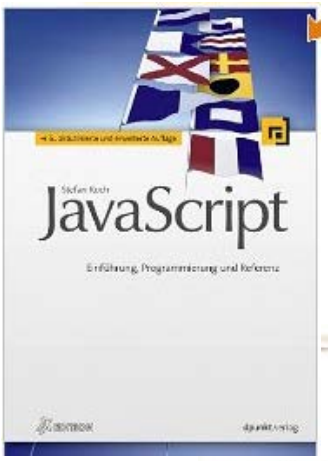
Zusammenfassende Fragen: Abschnitt 4

- Was kennzeichnet XML?
- Welche Bestandteile hat ein XML - Dokument?
- Wann ist ein XML - Dokument valide?
- Warum ist Validität wichtig?
- Was wird in einer DTD definiert?
- Welche Einsatzmöglichkeiten gibt es für XML?



Literatur

- Koch; JavaScript: Einführung, Programmierung und Referenz; Dpunkt Verlag; 2011



- RRZN; HTML 4 Zusatzwissen

Kapitel 5
JavaScript

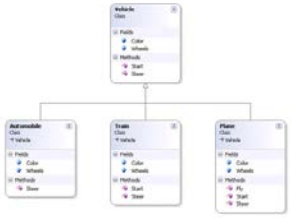
Wie läuft es ab

- Was wir machen werden:
  - Grundlegende Strukturen von Programmiersprachen am Bsp. JavaScript kennenlernen
  - Ausprobieren im Browser
  - Zusammenhänge verstehen
- Was wir nicht machen werden:
  - Teilbereiche bis ins Detail erschließen
  - Prozesse des Programmierens betrachten
  - Uns um das Design Gedanken machen

Charakteristik von JavaScript

- Vollwertige Programmiersprache

- Plattformübergreifend
- Objektorientiert



- Clientseitig, eingebunden in HTML

Was kann JavaScript?

- Erweiterung von statischen HTML-Seiten um dynamische Aspekte, z.B.:
  - Korrekte Eingabe eines Formulars prüfen, bevor es versendet wird
  - Alle Elemente einer Webseite manipulieren
  - Kleine und große Anwendungen

- Eingebunden in HTML-Dokument
  - Ausführung durch Webbrowser (=Interpreter)
  - Sandbox Prinzip: Nur Zugriff auf Objekte des Browsers, nicht auf Dateisystem!
    (>Sicherheit ist Problem des Browsers und des Betriebssystems)

Was sind Elemente einer Webseite?

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Einbindung in HTML und Struktur

- 1. Möglichkeit: **externe** Datei, Aufruf im Kopf des HTML Dokumentes

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

- 2. Möglichkeit: **direkt in** der Datei sowohl im Kopf als auch im Rumpf, typischerweise
  - Kopf: Definitionen, Funktionen, externe Dateien
  - Rumpf: Aufruf der Funktionalität

- Struktur:

```
<script type="text/javascript">
hier stehen die Anweisungen
</script>
```


Der Editor fügt u. U. um die JavaScript Anweisungen einen HTML-Kommentar ein. Dieser dient(e) dazu, JavaScript Anweisungen vor älteren Browsern zu verstecken, die die Sprache nicht verstehen.

Möchte man eine alternative Ausgabe für Browser mit abgeschaltetem JavaScript, so kann das Element <noscript> verwendet werden. Bsp:

```
<noscript>
	Der Browser unterst&uuml;tzt kein JavaScript
</noscript>
```

Zusammenfassende Fragen: Abschnitt 5.1

- Was ist die Charakteristik von JavaScript?
- Nennen Sie zwei Beispiele für die Verwendung von JavaScript!
- Wo wird JavaScript interpretiert und wie wird es eingebunden?

Kapitel 5
JavaScript

Kommentare

- Kommentare erläutern den Quelltext und werden nicht interpretiert
- Einzeilige Kommentare

```
// dies ist mein Kommentar für den Rest der Zeile
```

- Mehrzeilige Kommentare

```
/*     Dieser Kommentar erstreckt sich über mehrere
Zeilen, da ein komplexer Sachverhalt erläutert
werden muss
*/
```

- Kommentare sind wichtig. Wirklich.

Elementare Anweisungen

- Einfachste Bausteine zur Beschreibung der Funktionalität eines Programms
- Werden vom Browser interpretiert und ausgeführt

- Beispiel:

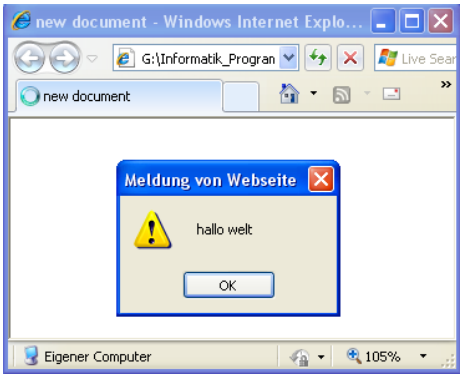
```
ergebnis = a + b;
```

- Anweisungen schließen mit einem Semikolon (häufige Fehlerquelle) !

- Die Anweisung

```
alert ("hallo welt");
```

gibt ein Hinweisfenster im Browser aus



Reservierte Wörter / Schlüsselwörter

- Sind Bestandteil der Sprache
- Haben eine feste vorgegebene Bedeutung
- Dürfen/können zu keinen anderen Zwecken benutzt werden

- Beispiele:

```
if          break          while          continue
return      case           true           var
```

- Vollständige Liste (ca. 60) unter selfhtml oder Literatur

Bezeichner

- Bezeichner benennen oder verweisen auf Variablen oder Funktionen

```
ergebnis = eurobetrag + steuerbetrag;
```

- Werden frei vom Programmierer vergeben
  → auf Sinn achten: **`a,b,c, x,`** oder **`ergebnis, eingabe, summe`**, ?

- Regeln:
  - Keine reservierten Wörter
  - Beginnen mit Buchstabe, $ oder _
  - Bestehen nur aus Buchstaben, Ziffern, $ oder _
  - → keine Leerzeichen, Punkte, %, etc.
  - Groß- und Kleinschreibung wird unterschieden

Wenn Regeln verletzt werden, dann

-bricht Editplus ab und gibt die Fehlerquelle an
-führen Browser das Skript trotzdem aus und es funktioniert u.U. eben nicht.

Variablen - Eigenschaften

- Können während der Laufzeit des Programms unterschiedliche Werte annehmen. Sie sind variabel.

- Für jede Variable wird Speicherplatz reserviert
- Werden über ihren Namen angesprochen, die Namensbildung erfolgt über die Regeln für Bezeichner
- Es wird kein Datentyp zugewiesen (loose typing)

Anderen Programmiersprachen muss bspw. direkt bei Erstellung einer Variablen ein Datentyp (Zusammenfassung konkreter Wertebereiche, z.B. Zahlenbereich) mitgegeben werden (strong typing). Die Bearbeitung erfordert dann meist weniger Speicherplatz und die Ausführung ist schneller.

Variablen - Definition

- Möglichkeiten zum Ansprechen von Variablen

```
var alter;      // Variable wird nur im Speicher angeglegt
var alter=21;   // Variable bekommt einen Wert zugewiesen
alter=21;       //   "
```

- Praxis: Beim ersten Auftreten einer Variable Schlüsselwort **`var`** benutzen

- Mehrere Variablen, getrennt durch Kommata

```
var i=5, k=6;    // Var i und k bekommen versch. Zahlenwerte
var i=k=5;       // Var i und k bekommen den Wert 5
var i, k=5;      // Var i ist undefiniert, k bekommt den Wert 5

var a=21, geschlecht="weiblich";     //  Var a ist eine Zahl,
                                     //  geschlecht ist ein String
```


Sollen Variablen schreibgeschützt werden, spricht man von Konstanten. Die Erstellung von Konstanten erfolgt über das Schlüsselwort „const“

const a=3;

Organisatorisches

- Fragen?
- Tip:

  Fehlerbehandlung: Wo und wie werden Fehler ausgeben?
  Editplus: ✓
  Browser: Fehlerkonsole!

Datentypen

JavaScript ist nur schwach typisiert

- Zahlen
  - Ganze Zahlen
  - Gleitkommazahlen, getrennt mit einem Punkt
- Zeichenketten (engl. strings)
  - Werden mit “ “ oder ‘ ‘ umschlossen
- Boolesche Werte
  - true (entspricht 1)
  - false (entspricht 0)

```
var alter=21.7;
```

```
geschlecht="weiblich";
text='ich bin "weiblich"';
```


Das Thema Anführungsstriche ist eine häufige Fehlerquelle. Denken Sie an das Syntax-Highlighting des Editors als Unterstützung!!!

Ebenfalls gibt es die Möglichkeit, in Zeichenkettenvariablen Steuerzeichen einzufügen.
Diese werden durch das Zeichen \ eingeleitet.

\\ erzeugt einen Backslash
\“ erzeugt Anführungsstriche
\n erzeugt einen Zeilenumbruch

Bsp.:

var b=“ich habe mit der Note \“sehr gut\“ bestanden“;

Bildschirmausgabe

- Ausgabe auf dem Bildschirm durch die Anweisung

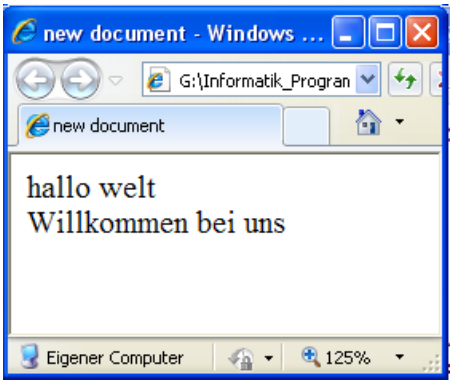
```
document.write();
```

- Inhalt können Variablen oder Datentypen sein

```
document.write("hallo welt");
document.write(i,k,a,geschlecht);
document.write("Ergebnis: " + euro);
```

- Inhalt wird direkt ins HTML geschrieben
  → auch HTML kann/muss eingebunden werden

```
document.write("hallo welt <br />");
document.write("Willkommen bei uns");
```



Eine Besondere Art der Bildschirmeingabe und -ausgabe bieten die Funktionen „prompt“ und „alert“. Wie das funktioniert sehen Sie im Praktikum. ☺

Arten von Operatoren

Operatoren sind Zeichen für Berechnungen, Vergleiche oder Verknüpfungen

- Arithmetische Operationen (Zahlen), z.B.:
  `+ - * ++`
- Vergleichsoperatoren (Zahlen, strings, Boolesche Werte), z.B.:
  `< == != >=`
- Verknüpfungsoperator (strings):
  `+`
- Logische Operatoren (Boolesche Werte)
  `&& || !`
- Zuweisungsoperatoren (alle Datentypen), z.B.:
  `= += -=`

Operatoren - Beispiele

- Durchführen von Berechnungen

```
ergebnis = a + b;
```

- Vergleich zwischen Variablen oder Werten

```
if (i==2)
```

- Herstellen von Verknüpfungen

```
name=vorname + ' ' + nachname;
```

- Operation hat ein oder zwei Operanden
  - Unäre Operatoren
  - Binäre Operatoren

```
i++; --m;
a+b;
```

Zusammenfassende Fragen: Abschnitt 5.2

- Welche reservierten Wörter kennen Sie?
- Was sind Bezeichner und welche Regeln gelten zu ihrer Bildung?
- Was sind Variablen in einer Programmiersprache?
- Was bedeutet loose typing?
- Welche Datentypen kennt JavaScript?
- Welche Arten von Operatoren gibt es in JavaScript?
- Was ist ein unärer Operator?

Kapitel 5
JavaScript

Anweisungsblöcke

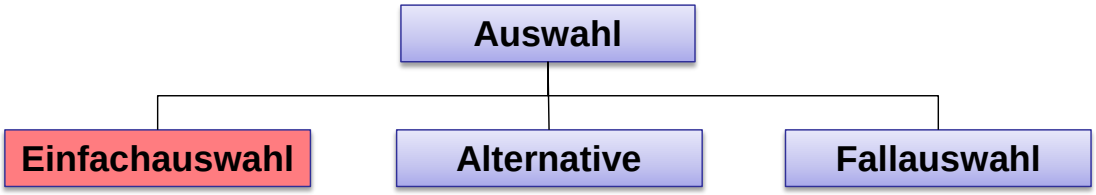
- Mehrere Anweisungen können zu einem Anweisungsblock zusammengefasst werden

```
{
Anweisung1;
Anweisung2;
}
```

- Anweisungsblöcke dürfen wiederum Anweisungsblöcke enthalten – sie sind geschachtelt

```
{
Anweisung1;
    {
    Anweisung2a;
    Anweisung2b;
    }
Anweisung3;
}
```

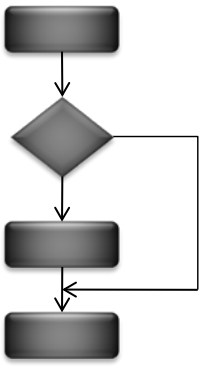
Einfache Auswahl mit if



- Auswertung eines booleschen Ausdrucks oder eines Vergleichs entscheidet über den Verlauf des Programms

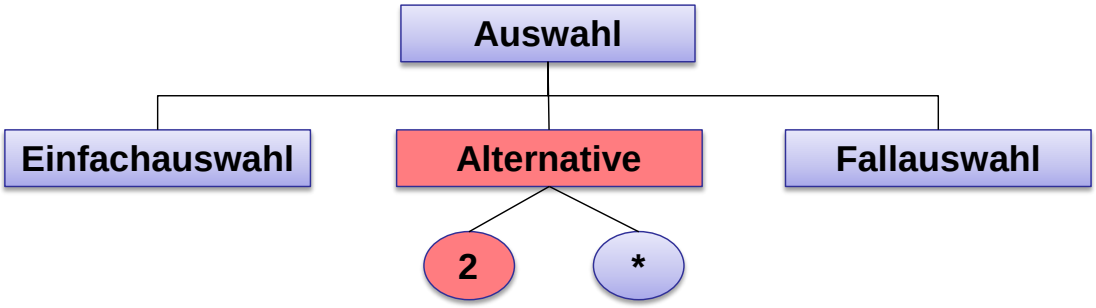
```
if (bedingung)
Anweisungsblock
```

- Der Anweisungsblock wird ausgeführt oder übergangen
- if-Anweisungen dürfen geschachtelt sein



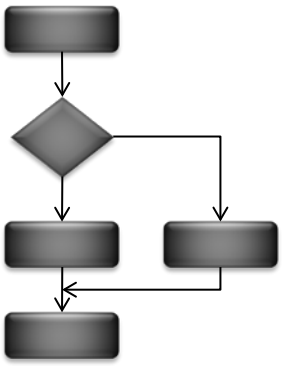

```
var alter=16;
var mensch;
if (alter<18)
        {
        mensch="Jugendlicher";
        }
document.write ("Sie sind ein ", mensch);
```

Alternative Auswahl mit if-else

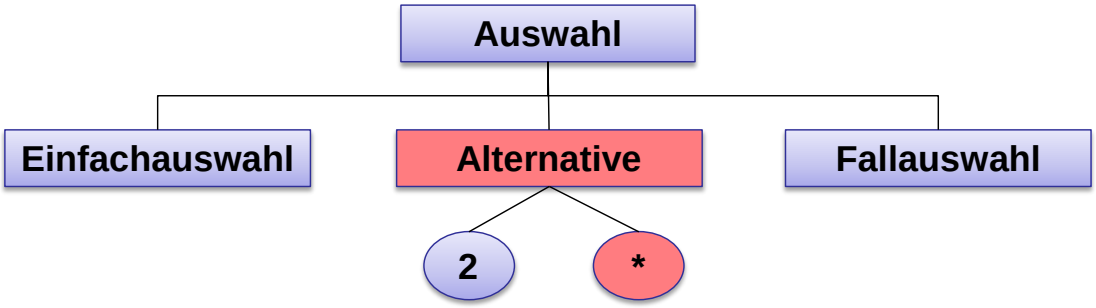


- Auswahl einer von zwei Alternativen bzw. zwei alternativen Anweisungsblöcken

```
if (bedingung)
Anweisungsblock1
else
Anweisungsblock2
```

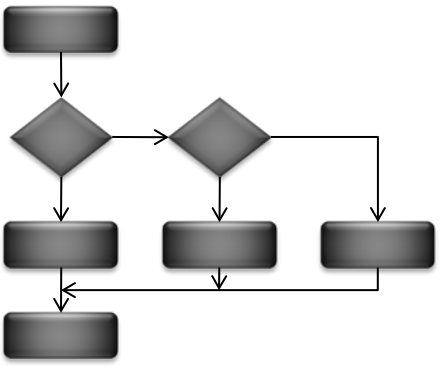


Alternative Auswahl mit if-else

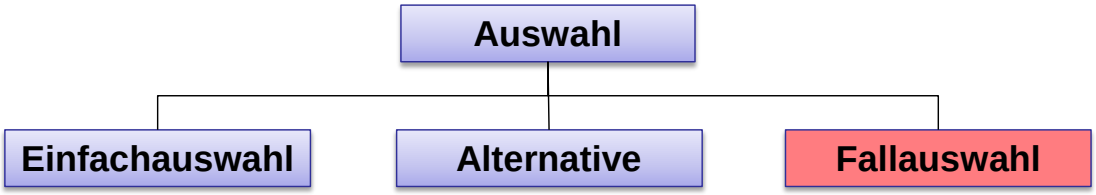


- Mehrstufige Auswahl, beliebig tief geschachtelt

```
if (bedingung_A)
Anweisungsblock1
else if (bedingung_B)
Anweisungsblock2
else
Anweisungsblock3
```



Fallauswahl mit switch



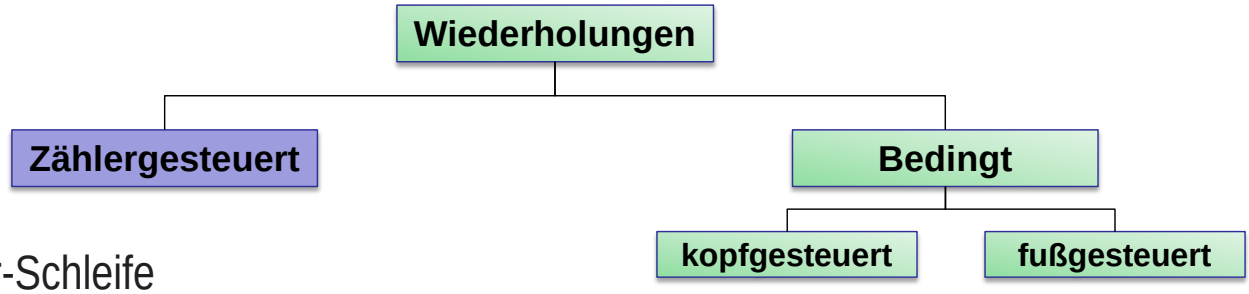
- switch (engl. Schalter/Weiche)
- Auswahl einer von vielen Alternativen

```
switch (selektor)
{
    case Wert1: Anweisungsblock; break;
    case Wert2: Anweisungsblock; break;
    case Wert3: Anweisungsblock; break;
    default     Anweisungsblock; break;
}
```



```
note=2;
var ausgabe;
switch (note)
{
        case 4: ausgabe="gerade so Bachelor";
                break;
        case 5: ausgabe="mangelhaft";
                break;
        case 6: ausgabe="miserabel";
                break;
        default: ausgabe="ein prima Bachelor";
}
document.write("Sie sind: ",ausgabe);
```

Zählergesteuerte Wiederholungen



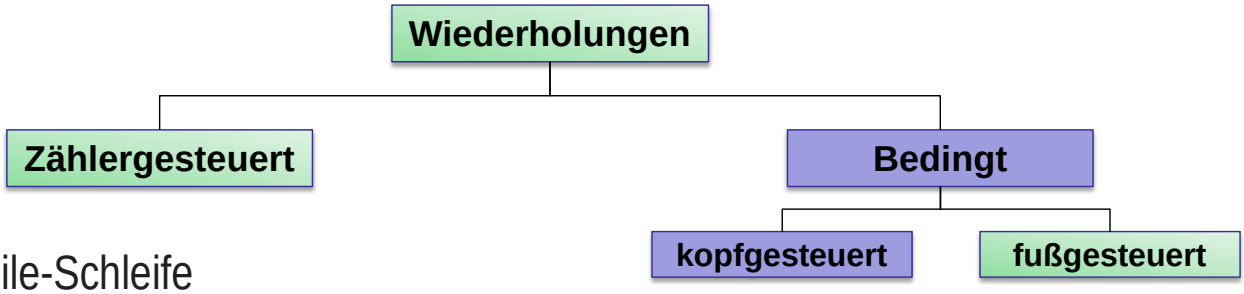
- for-Schleife
- Wiederhole die Schleife „n-mal“
- Die Anzahl der Durchläufe wird mitgezählt. Wenn die gewünschte Anzahl erreicht ist, wird die Schleife beendet

```
for (Initialisierung; Bedingung; Aktualisierung)

Anweisungsblock;
```



```
for (i=0;i<=10;i++)
        {
        document.write(“i hat den Wert: “, i, “<br />“);
        }
```

Kopfgesteuerte Wiederholungen



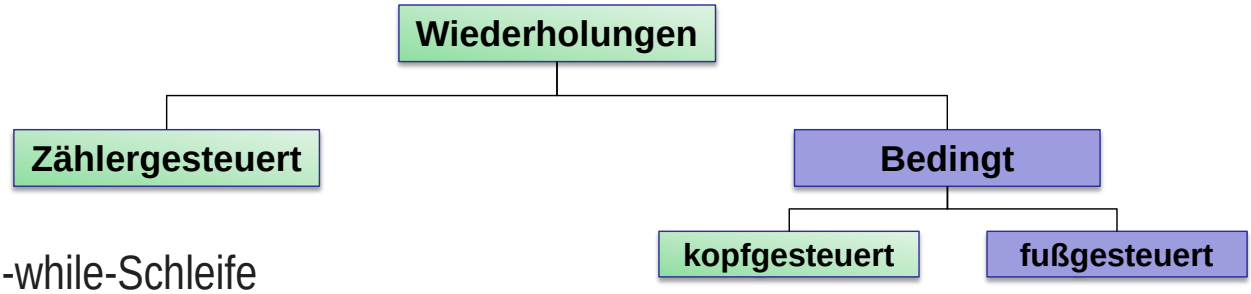
- while-Schleife
- Der Anweisungsblock wird ausgeführt, solange die im **Kopf** formulierte Bedingung wahr ist. Evtl. gar keine Ausführung.
- Gefahr der Endlosschleife, wenn die Bedingung im Rumpf nicht geändert wird

```
while (Bedingung)

Anweisungsblock;
```



```
var i=0;
while (i<=10)
        {
        document.write("i hat den Wert: ", i , "<br />");
        i++;
        }
```

Fußgesteuerte Wiederholungen



- do-while-Schleife
- Der Anweisungsblock wird ausgeführt, solange die im Fuß formulierte Bedingung wahr ist
- Die Schleife wird wenigstens einmal ausgeführt
- Ebenfalls Endlosschleife möglich

```
do

Anweisungsblock;

while (Bedingung);
```



```
var i=0;
do

        {
        document.write("i hat den Wert: ", i , "<br />");
        i++;
        }

while (i<=10);
```

Ausstieg aus einer Schleife

Vorzeitiger Ausstieg aus einer Schleife – in der Regel weil eine bestimmte Bedingung erfüllt ist

```
continue;
```

beendet den aktuellen Schleifendurchlauf und fährt mit dem nächsten Schleifendurchlauf fort

```
break;
```

beendet die Abarbeitung der Schleife insgesamt. Der Programmablauf wird mit der nächsten Anweisung nach der Schleife fortgesetzt.

Zusammenfassende Fragen: Abschnitt 5.3

- Was sind Anweisungsblöcke und wie werden sie definiert?
- Wie funktioniert eine mehrstufige Auswahl mit if,else?
- Welche Arten von Wiederholungen gibt es?
- Wie funktioniert eine do-while Schleife?
- Wozu dienen break und continue und was unterscheidet sie?

Kapitel 5
JavaScript

Intention

- Vereinfacht ausgedrückt: eine **Funktion** gibt einem Anweisungsblock einen Namen und evtl. Parameter, um
  - komplexe Werte zu Berechnen
  - die Anweisungen/Berechnungen im selben oder verschiedenen JavaScript-Programmen häufiger aufrufen zu können

- Funktionen werden zuerst definiert und können danach aufgerufen werden

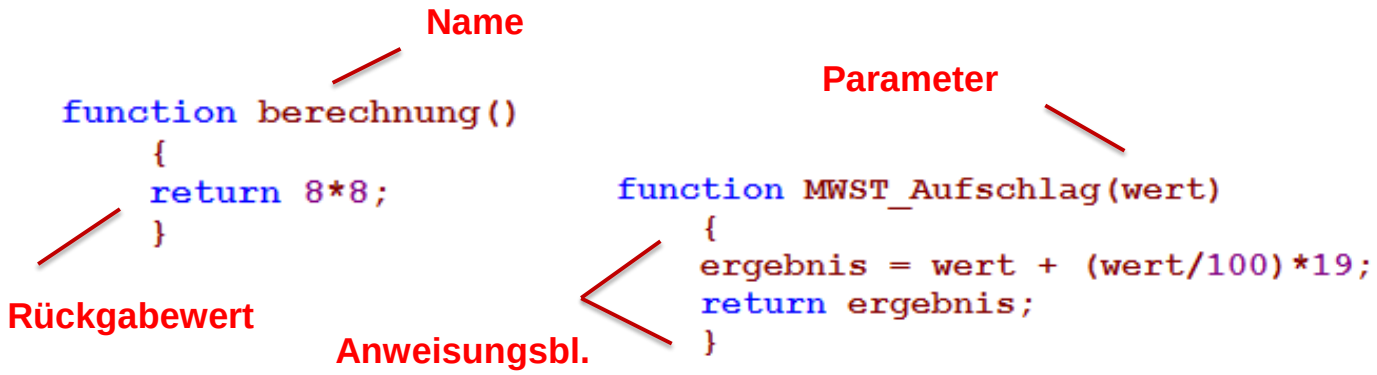
- Beispiel: Berechnung des Flächeninhalts eines Quadrats

```
Inhalt = a*a;
```

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Definition und Aufruf (1)

- Eine Funktion ist ein eigenständiges Unterprogramm, das in der JavaScript-Umgebung aufgerufen werden kann
- Eine Funktion hat einen Namen, über den sie identifiziert wird (→ Regeln für Bezeichner)
- Eine Funktion kann (keinen, einen oder mehrere) Parameter besitzen
- Eine Funktion kann einen Wert zurückgeben




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

        function berechnung_flaeche(seite_a, seite_b)
                {
                ergebnis = seite_a*seite_b;
                return ergebnis;
                }

        document.write(berechnung_flaeche(4,5));

 //-->
 </script>
```

Aufruf einer Funktion...

- ...innerhalb einer JavaScript Anweisung

```
ergebnis = 100+ berechnung_flaeche(4,5);
```

- ...innerhalb des Programmes

```
Hallo();
```

- ...innerhalb einer anderen Funktion

```
function grosses_Hallo()
    {
    Hallo();
    }
```



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

        function Hallo()
                {
                document.write("Hallo");
                }

        function grosses_Hallo()
                {
                document.write('<p style="font-size:30px;">');
                Hallo();
                document.write('</p>');
                }

        grosses_Hallo();

  //-->
  </script>
```

lokale und globale Variablen

- Lokale Variable
  - wird innerhalb einer Funktion durch das Schlüsselwort `var` definiert und ist nur dort gültig
  - außerhalb der Funktion unbekannt

- Globale Variable
  - wird außerhalb jeder Funktion definiert
  - überall bekannt – auch innerhalb jeder Funktion

```
function Hallo()
     {
     var b="test";    // lokale Variable
     a="test";        // globale Variable
     }
c="test";             // globale Variable
```


Gute Frage: wie sieht die Bildschirmausgabe dieses Programms aus?? ☺

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

        b="testausserhalb";
        function Hallo()
                {
                b="test";
                document.write(b);
                }

        Hallo();
        document.write(b);

  //-->
  </script>
```

vordefinierte Funktionen

- viele Funktionen werden von vielen Programmierern benötigt, sie sind bereits vordefiniert
- Test, ob ein Wert eine Zahl ist:

```
if (isNaN(postleitzahl)) {...}
```

- Umwandeln einer Zeichenkette in eine Zahl

```
var eingabe="50";
ergebnis= 10 + eingabe;              //ergibt "1050"
ergebnis= 10 + parseInt(eingabe);    //ergibt 60
```

parse = analysieren,
Int = Integer (ganze Zahl)


```
vorname=18;
        if (!isNaN(vorname))
                {
                document.write("Fehler: Vornamen haben keine Zahlen");
                }
```

Zusammenfassende Fragen: Abschnitt 5.4

- Wozu dienen Funktionen?
- Wie ist der Aufbau einer Funktion?
- Wie werden Funktionen aufgerufen?
- Was ist der Unterschied zwischen einer lokalen und globalen Variable?
- Wie kann geprüft werden, ob ein Wert (k)eine Zahl ist?

Kapitel 5
JavaScript

Die Begriffe: Klassen und Objekte

- Eine Programmiersprache, die wie JS im Wesentlichen mit Objekten arbeitet, heißt **objektorientierte Programmiersprache** (OOP)
- Eine **Klasse** ist in der Programmierung ein Modell eines Begriffs, in dem alles benötigte Wissen über den Begriff gekapselt enthalten ist
- Ein **Objekt** ist eine Instanz einer Klasse

→ Eine Klasse liefert den Bauplan für ein Objekt

- Klassen und Objekte haben
  - Immer einen Namen
  - Meistens Eigenschaften (Attribute)
  - Oft Fähigkeiten (Methoden)

Die Begriffe: Klassen und Objekte

- Eine Programmiersprache, die wie JS im Wesentlichen mit Objekten arbeitet, heißt **objektorientierte Programmiersprache** (OOP)
- Eine **Klasse** ist in der Programmierung ein Modell eines Begriffs, in dem alles benötigte Wissen über den Begriff gekapselt enthalten ist
- Ein **Objekt** ist eine Instanz einer Klasse

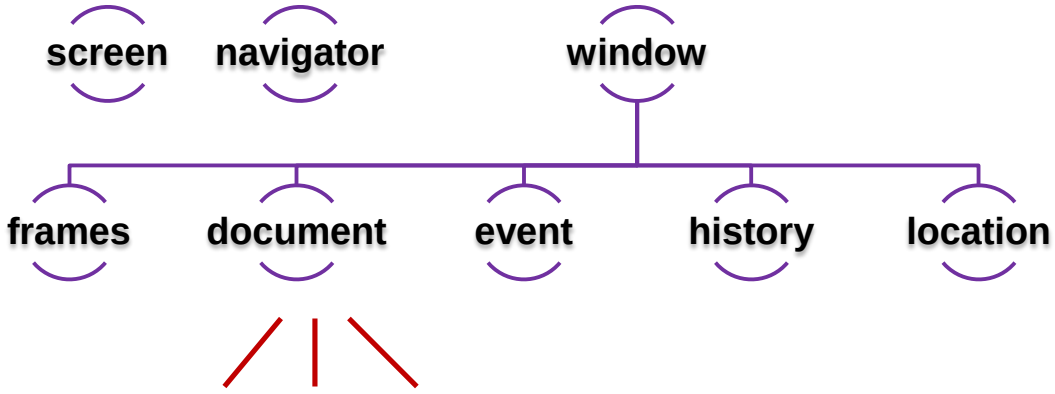
→ Eine Klasse liefert den Bauplan für ein Objekt

- Klassen und Objekte haben
  - Immer einen Namen
  - Meistens Eigenschaften (Attribute)
  - Oft Fähigkeiten (Methoden)

Attribute kann man als Variablen verstehen
Methoden kann man als Funktionen verstehen

Vordefinierte Klassen und Objektstruktur

- In JavaScript sind bereits einige Objekte/Klassen vordefiniert, z.B. String, Math, Array, Date,…
- Vordefinierte Browserobjekte werden vom Interpreter (=Browser) erzeugt und sind hierarchisch strukturiert



Aufgrund der häufigen Verwendung kann „window“ weggelassen werden.

Was bedeutet:
window.document.write(„Hallo zusammen“);
window.alert(x);

Arbeiten mit Objekten

- Klasse definieren
  z.B. die Klasse Fahrrad.
- Objekt erzeugen, durch `new objekt(argumente);`

```
var str = new String("Dies ist ein Test");
```

- Auf Eigenschaften und Methoden zugreifen

```
objekt.methode(argumente);
objekt.eigenschaft;
```

```
var x = str.length;
```

- Objekt zerstören
  nach Script-Ende automatisch

Wie eigene Klassen erstellt werden können behandeln wir nicht, da dies in JS eher selten der Fall ist. Wir konzentrieren uns auf von JavaScript bereits vorgefertigte Klassen.

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

        var str = new String("Dies ist ein Test");
        var x = str.length;

        document.write(x);

/*oder document.write(str.length):
        docuement.write(„Dies ist ein Test“.length);
*/


  //-->
  </script>
```

Objekt: String

- Alle Variablen, denen eine Zeichenkette zugewiesen wurde, sind String Objekte, und können die Methoden der String Klasse nutzen
  - Methoden zur Analyse oder Manipulation der Zeichenkette
  - Methoden zur Darstellung der Zeichenkette

- Aufruf in kurzer Schreibweise möglich:
  - Objekt erzeugen

```
                var meintext = new String("Dies ist ein Test");
/*oder*/        var meintext = "Dies ist ein Test";
```

  - Zugriff

```
                var laenge= meintext.length;
                document.write(laenge);

/*oder*/        document.write(meintext.length);
```

Objekt: String; Methoden zur Analyse u. Manipulation

Methode	Bedeutung
`charAt(n)`	Liefert das Zeichen an der Stelle n+1
`concat()`	Verbindet die Zeichenkette mit dem Argument
`indexOf(n)`	Gibt die erste Stelle zurück, an der sich das Argument befindet, sonst -1
`lastIndexOf()`	Wie indexOf, die Suche nach dem Argument beginnt jedoch am Ende des Strings
`toLowerCase()` `toUpperCase()`	Wandelt die Zeichenkette in Klein/Großbuchstaben um

u.a.

Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Objekt: String; Methoden zur Analyse u. Manipulation

- Beispiel:

```
//      0123456789
var meintext = new String("Hallo Welt");
```

Methodenaufruf	Ergebnis
`meintext.charAt(6)`	W
`meintext.concat(“, wie gehts?“)`	„Hallo Welt, wie gehts?“
`meintext.indexOf(“l“)`	**2**
`meintext.lastIndexOf(“l“)`	**8**
`meintext.toLowerCase()` `meintext.toUpperCase()`	„hallo welt“ „HALLO WELT“

Was bedeutet:

```
var meintext = new String("Hallo Welt");
document.write(meintext.concat(", wie gehts?"));
document.write(meintext.lastIndexOf("l"));
```

Objekt: String; Methoden zur Formatierung

- Verändern die Ausgabe einer Zeichenkette
- Funktionieren wie HTML Tags

- Beispiele:

```
document.write(meintext.big());      //große Schrift
document.write(meintext.bold());     //fette Schrift
document.write(meintext.strike());   //durchgestrichen
```

anstatt z.B.:

```
document.write("<b>" + meintext + "</b>");
```

durch CSS kaum noch Bedeutung

document.write(meintext.strike().bold());//durchgestrichen und fett

Objekt: Math

- Das Objekt **Math** hat einige mathematische Konstanten als Attribute, die wie jedes Attribut eines Objektes aufgerufen werden können

Math.PI	=	3.1415 …
Math.E	=	2.7182 …
Math.SQRT2	=	1.4142 …

```
function flaeche(r) {
    return Math.PI*r*r;
    }
```



```
document.write(Math.SQRT2*Math.PI);
```

Objekt: Math

- Beispiele von Methoden des Objekts Math

Methode	Bedeutung
abs(zahl) `Math.abs(-3.4)`	Liefert den Absolutbetrag 3.4
round(zahl) `Math.round(32.56)`	Rundet die Zahl auf oder ab 33
floor(zahl) `Math.floor(32.56)`	Rundet die Zahl immer ab 32
sqrt(zahl) `Math.sqrt(25)`	Liefert die Wurzel (square root) 5
random() `x = Math.random()`	Liefert eine Zufallszahl zwischen 0 und 1 z.B.: 0.675 ...

wie bekommt man eine zufallszahl zw. 10 und 20?

Math.floor(Math.random()*11)+10)

Math.round führt zu einer Ungleichverteilung,
vgl. auch http://aktuell.de.selfhtml.org/artikel/javascript/zufallszahlen/

Objekt: Date

- Das Objekt **Date** ermöglicht das Modellieren von Zeitangaben
- Definition:

```
heute  = new Date();
morgen = new Date(2011,5,18,11,47,13);
                  //jahr, Monat, Tag, Stunde, Minute, Sekunde
```

- Methoden u.a.:
  - `getYear(), getDate(), getDay(), getHours(), ...`
  - `getTime()` // liefert die Anzahl der Millisekunden seit 1.1.1970
  - `toLocaleString()` // liefert die ortsübliche Darstellung

Monate beginnen mit Januar =**0**
Wochen beginnen mit Sonntag =**0**

Wie könnte man Zeitabstände berechnen?

```
heute  = new Date();
morgen = new Date(2011,11,24,20,00,00);

differenz=Math.floor((morgen-heute)/1000/60/60/24);
document.write("Bis Weihnachten dauert es noch ",differenz," Tage");
```

Objekt: Array

engl.: Reihe, Aufreihung

- Das Objekt **Array** (auch: Variablenliste, Feld) fast mehrere Variablen zu einer Variable zusammen
- Bsp.: Alle Noten der Schüler einer Klasse
- Array-Elemente sind durchnummeriert, Algorithmen können darauf zugreifen
  - Alle Noten der Klasse anzeigen, zusammenzählen, Durchschnitt bilden

Objekt: Array

- 3 Möglichkeiten zur Bildung:
  - Array mit dynamischer Länge:

```
reihe = new Array();
```

  - Array mit fester Länge:

```
noten = new Array(30);
```

  - Array mit direkter Wertzuweisung:

```
namen = new Array("Hannah", "Anita", "Laura");
```

- Zugriff mittels Indexoperator (beginnend bei 0):

```
namen[1]="Elisa";
```


Ausgabe durch z.B.:

```
document.write(namen[1]);
```

Das Objekt **Array** besitzt das Attribut **length**.

Wie kann eine Ausgabe aller Werte erfolgen?

```
namen = new Array("Hannah", "Anita", "Laura");

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

Achtung: length beginnt bei 1, das Array bei 0

Objekt: Array

- Neben dem Attribut **length** besitzen Arrays u.a. folgende Methoden:

Methode	Bedeutung
shift() `namen.shift();`	Auslesen und Löschen des ersten Feldelementes
push(Wert) `namen.push("Elisa");`	Fügt Wert hinten ans Feld an "Hannah","Anita","Laura","Elisa"
sort() `namen.sort()`	Sortiert die Liste alphabetisch "Anita","Elisa","Hannah","Laura"
join(string) `namen.join("--")`	Fügt die Elemente mit den angegebenen Trennzeichen zu einem String zusammen "Anita--Elisa--Hannah--Laura"

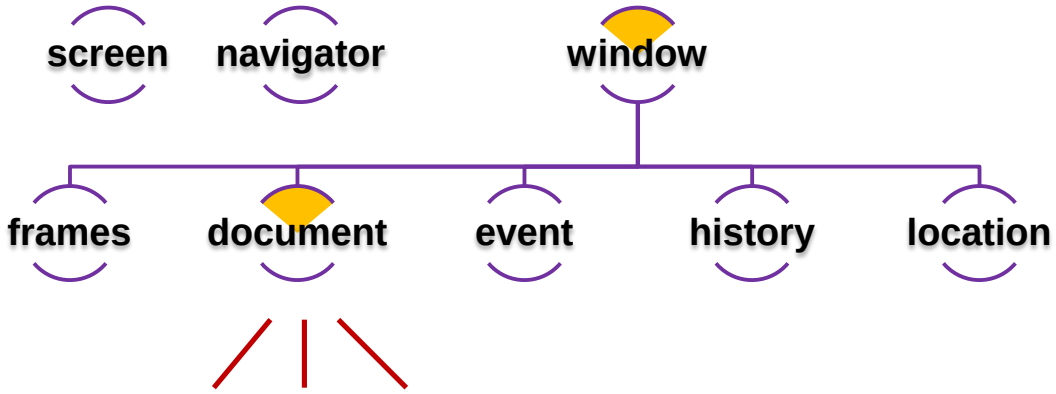
sort() sortiert als Zeichenkette → 19 steht vor 9

Wie kann ein Array zusammengezählt und z.B. der Durchschnitt gebildet werden?

```
noten = new Array(2,4,5,2,4,6,2,1);
var ergebnis=0;

for (i=0;i<noten.length;i++ )
        {
        ergebnis+=noten[i];
        }
document.write("Der Notendurchschnitt beträgt: ", ergebnis/noten.length);
```

Document Object Model (DOM)

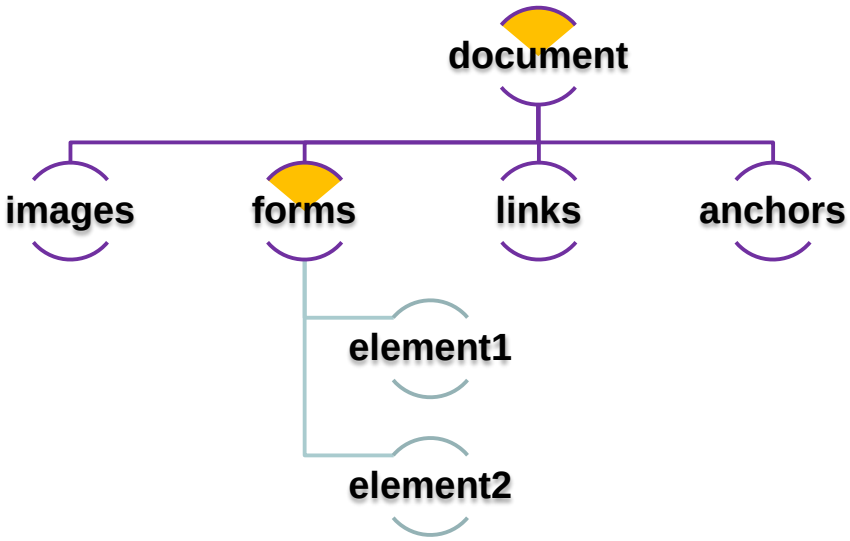
- String, Math, Array, Date,…



Document Object Model (DOM)

- Idee: Die Elemente einer Webseite werden in einer Objekthierarchie abgelegt. Dadurch können einzelne Objekte angesteuert und verändert werden

- u.a.:



Wir behandeln Formulare als Beispiel.
Es lassen sich aber auch Bilder, Links, Anker et al. ansprechen und ihre Attribute verändern.

Document Object Model (DOM)

- Die Objekte befinden sich in einem Array
- Zugriff über den Indexoperator `document.forms[1]`

  oder den Namen `document.meinformular`

  oder z.B. der Methode `document.getElementById("meinformular")`
  `getElementById()`
- Typische Funktion:
  - Ausgabe von neuem Inhalt in bestimmte Elemente
  - Änderung einzelner Attribute
  - Häufig auch: Zugriff auf Formularwerte

Die Methode getElementById() wir am Häufigsten verwendet.
Tatsächlich gibt es weitere wie getElementsbyName() oder getElementsByTagName()

DOM - Beispiel

Änderung von Inhalten über Formularbutton und der Eigenschaft `innerHTML`

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

function aendern() {
    document.getElementById("hier").innerHTML="neuer Inhalt";
}

</script>

<form action="">
    <input type="button" value="aendern" onClick="aendern()"/>
</form>

<p id="hier">Bereich zum Verändern</p>
```

DOM - Beispiel

Änderung einzelner Attribute in einem Formular

```
document.forms[1].eingabe.value="test2";
//oder
document.meinformular.eingabe.value="test1";
//oder
document.getElementById("meinformular").eingabe.value="test3";
```


Wie sieht die Bildschirmausgabe aus?

```
<body>

        <form id="meinformular" name="meinformular">
                <input type="text" name="eingabe" value="" />
        </form>
        <form>
                <input type="text" name="eingabe" value="" />
        </form>

 <script type="text/javascript">
 <!--
                document.forms[1].eingabe.value="test2";
                //oder
                document.meinformular.eingabe.value="test1";
                //oder
                document.getElementById("meinformular").eingabe.value="test3";
 //-->
 </script>
</body>
```

Zusammenfassende Fragen: Abschnitt 5.5

- Erläutern Sie die Idee und den Aufbau von Objekten
- Wie wird auf Eigenschaften und Methoden von Objekten zugegriffen?
- Erläutern Sie die Funktion und wesentliche Eigenschaften/Methoden der Objekte String, Math, Array, Date
- Wie kann auf einzelne Elemente eines Dokumentes zugegriffen werden?

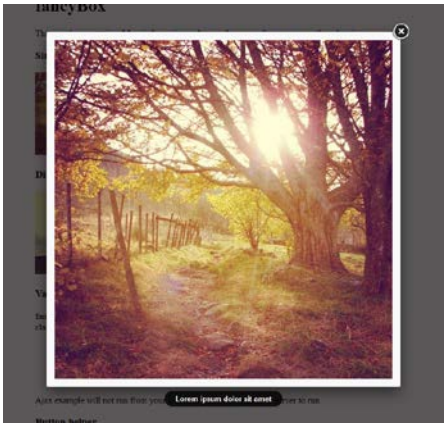
Kapitel 5
JavaScript

Frameworks

- Viele Klassen, Funktionen/Methoden kommen häufig zum Einsatz.
  Die Idee von Frameworks besteht darin, diese vorgefertigt zur Verfügung zu stellen

- Typische Funktionalitäten:
  - Auswahl und Manipulation der Elemente
  - Grafische Animationen

- Anwendungsbeispiele:
  - Boxansicht von Bildern
  - Tabs/Sliders



Hochschule Ostwestfalen-Lippe
University of Applied Sciences

Frameworks

- Das bekannteste Framework ist jQuery

- Die Einbindung erfolgt wie bekannt im Kopf der Datei von extern oder lokal:

```
<script src="http://code.jquery.com/jquery-latest.js"></script>
```

- Der generelle Zugriff erfolgt durch z.B.

```
$(selector).action();
```

```
$('h1').hide();
```

Einleitung — Auswahl des Elements — Methode


```
<!DOCTYPE html>
<html lang="de">
<head>
  <title>jQuery Beispiel</title>
  <script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

<script type="text/javascript">
//verhindert die Ausführung, bevor das ganze Dokument geladen ist
$(document).ready(function(){
    // Hier die jQuery Befehle
    $("input").click(function(){
                $('h1').hide();
                }
            );
});
</script>

</head>
<body>
<h1>jQuery Beispiel</h1>
<form action="">
        <input type="button" value="ausblenden"/>
</form>
</body>
</html>
```

Zusammenfassende Fragen: Abschnitt 5.6

- Was ist die Idee von Frameworks?
- Was sind typische Funktionen und Anwendungsbeispiele
- Wie heißt das bekannteste JavaScript Framework?
- Wie werden Frameworks eingebunden?