

# 4. Veranstaltung

## Inhalt CSS:

- ❑ Definition, Ziel, Versionen
- ❑ Syntax und Kommentare in CSS
- ❑ class- und id-Selektoren
- ❑ Gruppierte, verschachtelte, attributabhängige Selektoren
- ❑ Möglichkeiten der Einbindung und Priorität
- ❑ Gestaltungs-Möglichkeiten mit CSS
- ❑ Das CSS-Boxmodell
- ❑ Pseudoklassen
- ❑ Die tags 'div' und 'span' im Zusammenhang mit CSS

# 4. Veranstaltung

## Inhalt CSS:

- ❑ **Definition, Ziel, Versionen**
- ❑ Syntax und Kommentare in CSS
- ❑ class- und id-Selektoren
- ❑ Gruppierte, verschachtelte, attributabhängige Selektoren
- ❑ Möglichkeiten der Einbindung und Priorität
- ❑ Gestaltungs-Möglichkeiten mit CSS
- ❑ Das CSS-Boxmodell
- ❑ Pseudoklassen
- ❑ Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Definition

## CSS

(**C**ascading **S**tyle **S**heets = *stufenförmige* oder *(hintereinander) geschachtelte Gestaltungsvorlagen*)

sind eine unmittelbare **Ergänzung** zu HTML.

Es handelt sich dabei um eine Sprache zur **Definition** von **Formateigenschaften** einzelner **HTML-Elemente**.



# CSS: Definition

Die **semantische** und **inhaltliche** Struktur des Dokumentes werden damit strikt von **Gestaltung**, **Formatierung** und **Layout** getrennt.

Die **Gestaltungsmöglichkeiten** gehen dabei **weit** über das hinaus, was mit "purem" HTML möglich ist.

# CSS: Ziel

- ❑ HTML war **nie** dafür gedacht, tags für die **Gestaltung** oder **Formatierung** eines Dokumentes zu bieten.
- ❑ Mit tags wie `<font>` und **Farbeigenschaften** etc. (ab **HTML 3.2**) begann der Webentwickler- Alptraum.
- ❑ Um dieses **Problem** zu **lösen**, erschuf das World Wide Web Consortium (**W3C**) die 'Style-Sprache' **CSS**.

# CSS: Ziel

---

Seit **HTML 4.0** können **alle** Format-Eigenschaften vom HTML Dokument **entfernt** und in einer **separaten** CSS-Datei untergebracht werden.

# CSS: Versionen



## **Dezember 1996: CSS Level 1 *Recommendation***

Diesen Normierungsvorschlag befolgen die aktuellen Browser mittlerweile fast vollständig.

# CSS: Versionen



**Mai 1998:** CSS Level 2 (CSS2) *Recommendation*

Bis heute wird diese Empfehlung von **keinem** verbreiteten Browsern **vollständig** implementiert.

Bei der Verwendung im Web ergeben sich daher oft **Schwierigkeiten**.



# CSS: Versionen



**Juni 2011:** CSS Level 2 Rev. 1 (CSS 2.1) *Recommendation*

Es sollten die **Erfahrungen** mit **CSS2** berücksichtigt werden.

CSS 2.1 bringt jedoch selbst **keine** grundlegend **neuen** Fähigkeiten mit sich.

# CSS: Versionen

**Seit 2000: CSS Level 3 (CSS3) Drafts** *(Entwurfsphase)*

**CSS3** ist im Gegensatz zu den Vorgängern **modular** aufgebaut: einzelne **Teiltechniken** werden nach eigenem/n **Rhythmus/Versionsschritten** entwickelt.

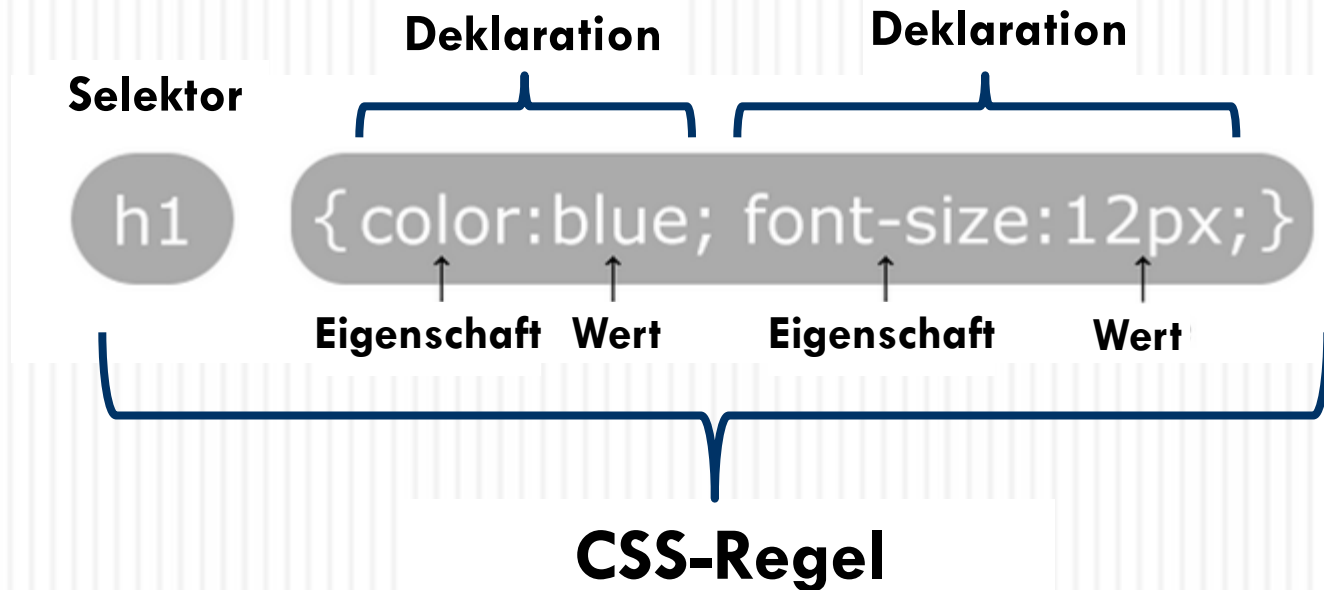
Die Browser haben bisher nur Teile von CSS3 implementiert.

# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- **Syntax und Kommentare in CSS**
- class- und id-Selektoren
- Gruppierte, verschachtelte, attributabhängige Selektoren
- Möglichkeiten der Einbindung und Priorität
- Gestaltungs-Möglichkeiten mit CSS
- Das CSS-Boxmodell
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Syntax (Aufbau eines Styles)



Eine **CSS-Deklaration** endet immer mit einem **Semikolon**, eine oder mehrere Deklarationen werden in **geschweifte** Klammern eingeschlossen.

# CSS: Syntax (Aufbau eines Styles)

```
<head>
<style type="text/css">
  p
  {
    color:red;
    text-align:center;
  }
</style>
</head>
```

Für **Anfänger** und für **Lehrzwecke** ist diese Quellcode-Formatierung **übersichtlicher**



Von "**Profis**" wird diese Quellcode-Formatierung **bevorzugt**:



```
p {color:red;text-align:center;}
```

# CSS: Selektoren gruppieren

**Kommentare** helfen anderen und auch dem Autor selbst, den Quellcode besser zu **verstehen**, zu **gliedern** und ggf. **Metadaten** (Autor, Version etc.) anzugeben.

## ***Achtung:***

Die Syntax für HTML- und CSS-Kommentare ist unterschiedlich!

```
/*  
Version: 1.0  
erstellt: gestern  
von: mir  
*/
```

```
/* typo css */  
html {  
    font-family: verdana;  
    color: #666;  
    font-size: 75%;  
}
```

```
/* reset css */  
body, h1, h2 {  
    margin: 0;  
}
```

# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- **class- und id-Selektoren**
- Gruppierte, verschachtelte, attributabhängige Selektoren
- Möglichkeiten der Einbindung und Priorität
- Gestaltungs-Möglichkeiten mit CSS
- Das CSS-Boxmodell
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: class- und id-Selektoren

## id-Selektor

#myid

```
{ color:blue; font-size:12px; }
```

## class-Selektor

.myclass

```
{ color:blue; font-size:12px; }
```

Zusätzlich zum Typ-Selektor, der ein **bestimmtes** HTML-Element formatiert, kann man mit CSS **eigene** Selektoren spezifizieren: **class-** und **id-**Selektoren



# CSS: class- und id-Selektoren

## id-Selektor

#myid

{color:blue; font-size:12px;}

Der **id-Selektor** wird benutzt, um ein Format für ein **einzelnes Element** zu spezifizieren, das im Dokument nur **einmal** vorkommen darf.

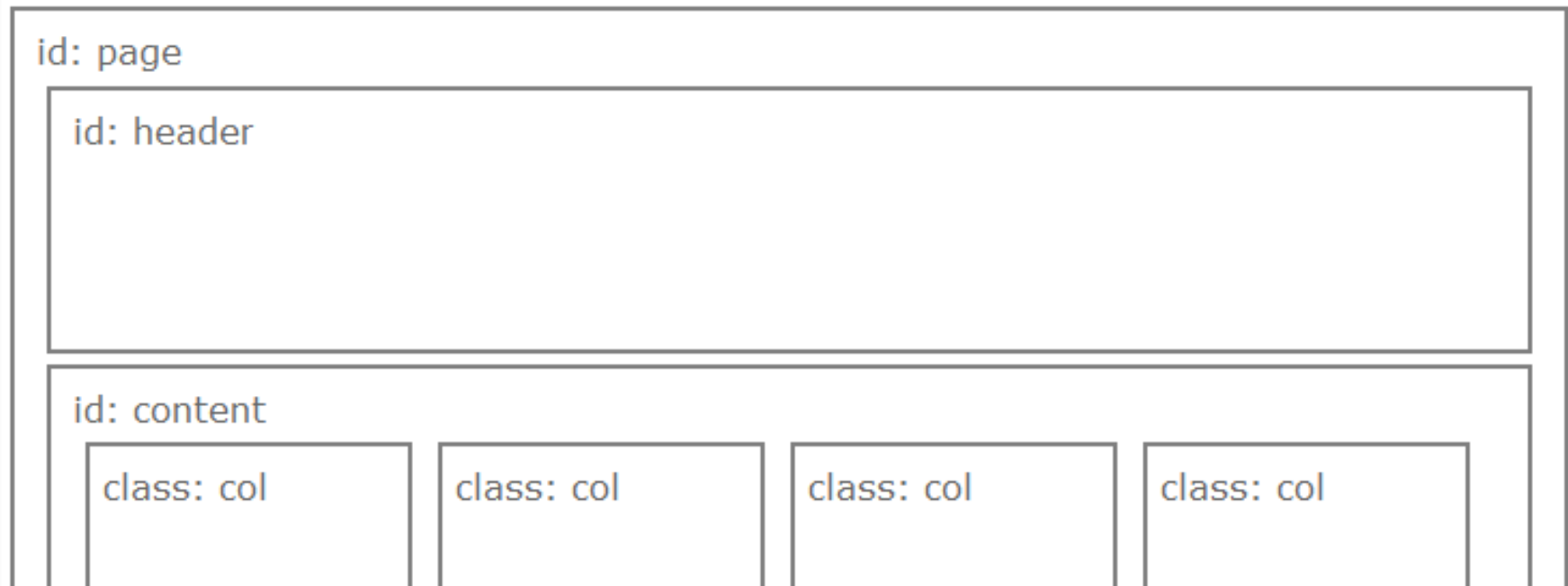
Das Element im body, welches mit dem Format verknüpft wird, muss wie folgt codiert werden:

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

# CSS: class- und id-Selektoren

Sinnvolle Einsatzgebiete des id-Selektors:

- ❑ Im Zusammenhang mit Javascript
- ❑ Um eindeutige Bereiche eines Layouts zu kennzeichnen



# CSS: class- und id-Selektoren

## class-Selektor

.myclass

{ color:blue; font-size:12px; }

Der **class-Selektor** wird benutzt, um das Format für eine **Gruppe** von Elementen zu spezifizieren, die im Dokument meist **häufiger** vorkommen.

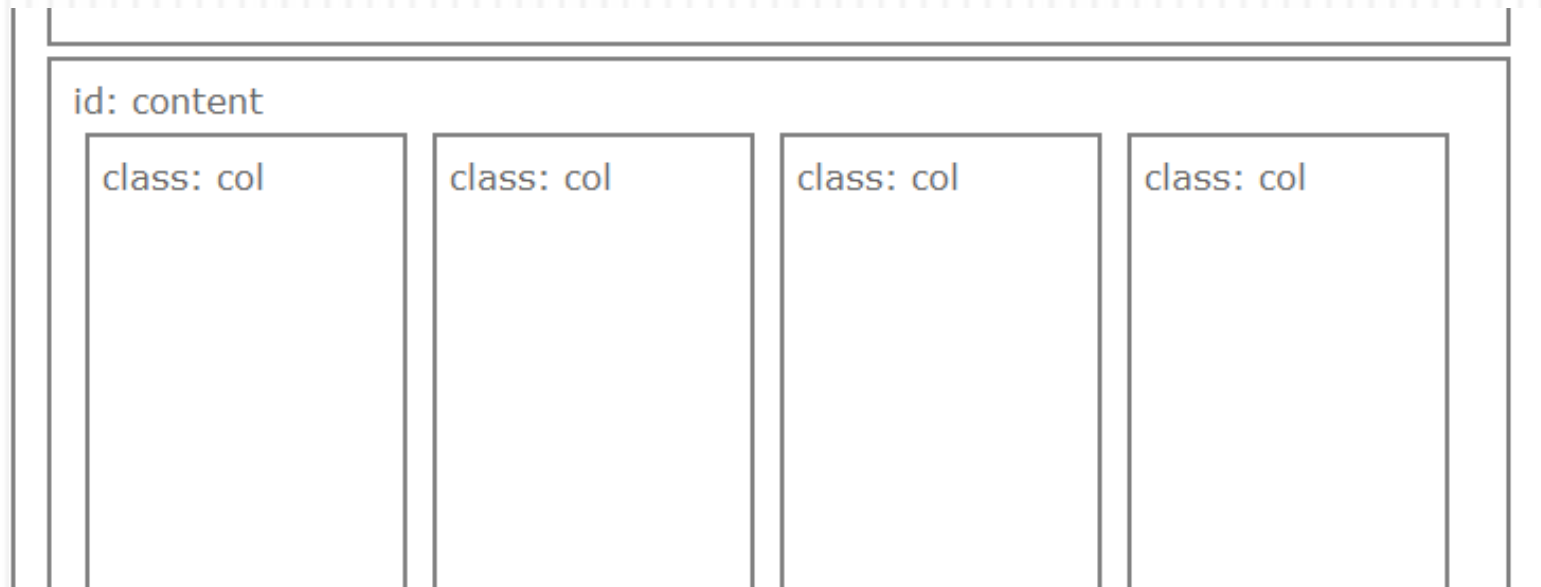
Das Element im body, welches mit dem Format verknüpft wird, muss wie folgt codiert werden:

```
<p class= "myclass">Hello World!</p>
```

# CSS: class- und id-Selektoren

Sinnvolle Einsatzgebiete des class-Selektors:

- Wird dort eingesetzt, wo verschiedenen Elementen dasselbe Format zugeordnet werden soll (z.B. Spalten)



# CSS: class- und id-Selektoren

Beide "Universal-Attribute" dürfen in einem Element **gleichzeitig** benutzt werden. Das **id-Attribut** hat jedoch **Vorrang** bei sich widersprechenden Attribut-Werten.

```
<p class= "myclass" id = "myid">Hello</p>
```

# CSS: class- und id-Selektoren

Man darf **einem** Element **mehrere** class-Selektoren **gleichzeitig** zuweisen (durch Leerzeichen getrennt).

```
<p class= "myclass1 myclass2">Hello</p>
```

Die **Eigenschaften aller** zugewiesenen Klassen werden dann **kombiniert**.

(Bei **id's** macht das **keinen** Sinn, weil sie ein Element eindeutig kennzeichnen sollen!)

# CSS: class- und id-Selektoren

h1.myclass

```
{ color:blue; font-size:12px; }
```

h1#myid

```
{ color:blue; font-size:12px; }
```

Durch die Kombination von Typ- und id- bzw. class-Selektoren kann bewirkt werden, dass die Klasse/Id nur **einem bestimmten** Element zugewiesen werden kann.

```
<p class="myclass">Hello</p>
```

Keine Wirkung

```
<h1 id="myclass">Hello</h1>
```

Wirkung

# CSS: class- und id-Selektoren

\*

```
{color:blue; font-size:12px;}
```

Zu erwähnen ist noch der "**Universal-Selektor**", der jedoch **so gut wie nie** zum Einsatz kommt.

Damit wird angegeben, dass **jedes** Element im Dokument ein bestimmtes Format bekommt.

Im praktischen Einsatz gibt jedoch eher den äußersten Elementen wie body oder html eine Eigenschaft, die 'dokumentweit' gelten soll.



# CSS: class- und id-Selektoren

Sinn machen würde der Universal-Selektor in Verbindung mit Klassen.....aber

`*.myclass`

`{color:blue; font-size:12px;}`

**entspricht exakt**

`.myclass`

`{color:blue; font-size:12px;}`

und wir sind wieder beim **class-Selektor**

# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- class- und id-Selektoren
- **Gruppierte, verschachtelte, attributabhängige Selektoren**
- Möglichkeiten der Einbindung und Priorität
- Gestaltungs-Möglichkeiten mit CSS
- Das CSS-Boxmodell
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Selektoren gruppieren

In CSS gibt es oft Elemente mit **identischem** Stil.

Um den Code **schlank** zu halten, können Selektoren (auch class-/id-Selektoren) **gruppiert** werden.

Dabei wird jeder Selektor mit einem **Komma** separiert.

```
h1 {color:red;}  
h2 {color:red;}  
h3 {color:red;}  
.myClass {color:red;}
```

*...ist identisch mit...*

```
h1, h2, h3, .myClass  
{  
  color:red;  
}
```

# CSS: Selektoren verschachteln (nesting)

Man kann einen Selektor mit Stil-Angaben verknüpfen, wenn dieser in einen **anderen** Selektor **verschachtelt** ist (auch mehrere Ebenen tiefer). Dabei wird jeder Selektor mit einem **Leerzeichen** separiert.

## Wir lernen Stylesheets

Wir lernen *Stylesheets*

```
<style type="text/css">
h1 {color:red;}
h1 i {color:blue; font-style:normal;}
</style>
```

...

```
<h1>Wir lernen <i>Stylesheets</i></h1>
<p>Wir lernen <i>Stylesheets</i></p>
```

# CSS: Selektoren verschachteln (nesting)

Wenn Sie nichts anderes angeben, übernimmt ein HTML-Element, das innerhalb eines anderen HTML-Elements vorkommt, dessen Eigenschaften und fügt seine eigenen Eigenschaften nur hinzu (kaskadiert).

## Wir lernen Stylesheets

Wir lernen *Stylesheets*

```
<style type="text/css">
h1 {color:red;}
h1 i {color:blue; font-style:normal;}
</style>
...
```

```
<h1>Wir lernen <i>Stylesheets</i></h1>
<p>Wir lernen <i>Stylesheets</i></p>
```

# CSS: Selektoren verschachteln (nesting)

Es gibt seit CSS2 beim nesting noch spezifischer Angaben, die jedoch eher selten Einsatz finden:

```
<style type="text/css">
div i { color:red; }
div * b { color:violet; }
div > p { color:blue; }
div + p { margin-top:5em; }
</style>
```

Bei Bedarf findet man schnell im Internet oder Literatur Erläuterungen hierzu.

# CSS: attributabhängige Selektoren

Mit Hilfe **attributbedingter** Formate können Sie in Selektoren angeben, dass Formatdefinitionen **nur** für Elemente mit **bestimmten Attributen** oder sogar **nur** für Elemente mit **bestimmten Wertzuweisungen** an Attribute gelten sollen.

# CSS: attributabhängige Selektoren

```
p[align] {color:red;}
/* p-Elemente die das Attribut align haben */
p[align=center] {color:blue;}
/* p-Elemente die ein align="center" im
   einleitenden Tag haben */
td[abbr~=Berlin] {color:#FFFF00;}
/* p-Elemente bei denen in der Wertzuweisung
   an dieses Attribut das Wort Berlin vorkommt */
*[lang|=en] {color:#FFFFFF;}
/* alle Elemente mit Attribut lang mit einem Wert,
   der mit en beginnt und danach eventuell einen
   Bindestrich enthalten (wie en-US) */
```




# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- class- und id-Selektoren
- Gruppierte, verschachtelte, attributabhängige Selektoren
- **Möglichkeiten der Einbindung und Priorität**
- Gestaltungs-Möglichkeiten mit CSS
- Das CSS-Boxmodell
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Einbindung und Priorität

Einbindung	Gültigkeit
1. Inline-Style	gilt nur für diesen Befehl
2. Internes Stylesheet	gilt für das ganze Dokument
3. Externes Stylesheet	gilt für alle einbindenden HTML-Seiten



# CSS: Einbindung und Priorität

## Einbindung

## Anwendungsfall

### 1. Inline-Style

Sollte nur in **Ausnahmefällen** eingesetzt werden, da es nahezu alle Vorteile von Stylesheets verhindert.

### 2. Internes Stylesheet

Sollte eingesetzt werden, wenn eine **einzelne** Seite eine **spezielle** Gestaltung aufweisen soll.

### 3. Externes Stylesheet

Geeignet, wenn die Gestaltung für **mehrere** Seiten angewandt werden soll. Das Erscheinungsbild einer **gesamten** Website kann so mit nur einer Datei beeinflusst werden.

# CSS: Einbindung und Priorität

## Einbindung

### 1. Inline-Style

## Gültigkeit

gilt nur für diesen Befehl

Einbindung in einem einzelnen HTML-Tag:

```
<h1 style="{font-size:18px;font-weight:bold;}">  
    Überschrift  
</h1>
```

# CSS: Einbindung und Priorität

## Einbindung

### 2. Internes Stylesheet

Einbindung in den HEAD-Bereich eines Dokumentes :

## Gültigkeit

gilt für das ganze Dokument

```
<style type="text/css">
<!--
    /* ich bin ein CSS-Kommentar*/
    p
    {
        font-family: arial;
        color: red;
        font-weight: bold;
    }
-->
</style>
```

# CSS: Einbindung und Priorität

## Einbindung

3. Externes Stylesheet

## Gültigkeit

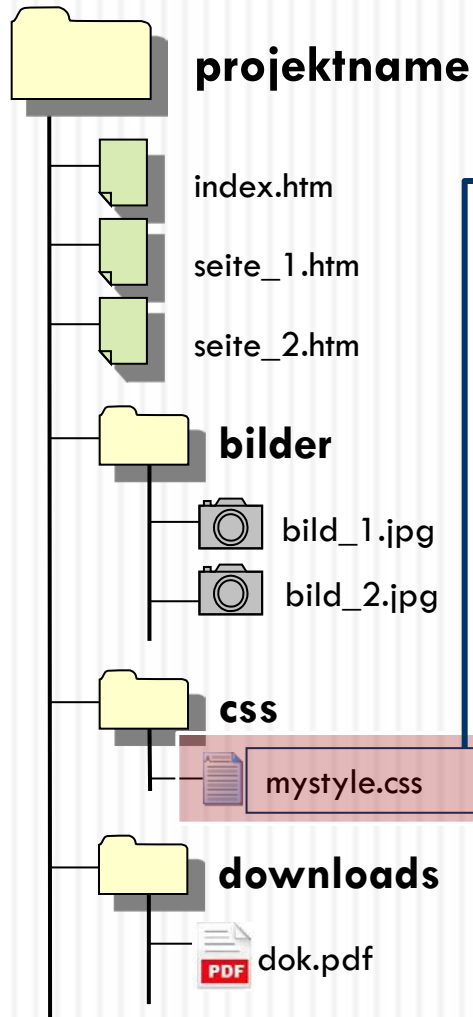
gilt für alle einbindenden HTML-Seiten

Einbindung in einer externen Datei:

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

```
<html>
<head>
<title>Stylesheets für unterschiedliche Ausgabemedien</title>
<link rel="stylesheet" media="screen" href="website.css">
<link rel="stylesheet" media="print, embossed" href="druck.css">
<link rel="stylesheet" media="aural" href="speaker.css">
</head>
```

# CSS: Einbindung und Priorität



```
hr {color:sienna;}  
p {margin-left:20px;}  
body {background-image:url("images/back40.gif");}
```

Eine **Textdatei** mit der Endung **.css** enthält die **Stylesheet-Angaben**.

Die Angaben dort werden **nicht** mit dem **<style>** - Tag eingeschlossen


# CSS: Einbindung und Priorität

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

 **projektname**

 index.htm

 seite\_1.htm


 seite\_2.htm

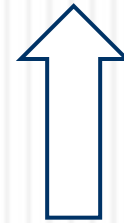
 **bilder**

 bild\_1.jpg

 bild\_2.jpg

 **css**

 mystyle.css



Das an einer beliebigen Stelle im **<head>** - Bereich eingefügte **<link>** - Tag verweist mittels des Attributs **href** auf die **externe** Datei (auch komplette URL-Adressen möglich).



# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- class- und id-Selektoren
- Gruppierte, verschachtelte, attributabhängige Selektoren
- Möglichkeiten der Einbindung und Priorität
- **Gestaltungs-Möglichkeiten mit CSS**
- Das CSS-Boxmodell
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Gestaltungs-Möglichkeiten mit CSS

Es gibt zahlreiche CSS-Eigenschaften, mit denen man HTML-Elemente formatieren kann.

Grobe Gliederung in einzelne Kategorien:

- Schriftformatierung
- Textformatierung
- Listenformatierung
- Tabellenformatierung

# CSS: Gestaltungs-Möglichkeiten mit CSS

Hierzu gibt es genügend Beispiele und Erläuterungen in Literatur und im Internet (siehe Linkempfehlungen).

Weitere **wichtige** Eigenschaften werden im Zusammenhang mit dem **Boxmodell** und der **Positionierung / Anzeige** in dieser und in der nächsten Präsentation näher erläutert.

# 4. Veranstaltung

## Inhalt CSS:

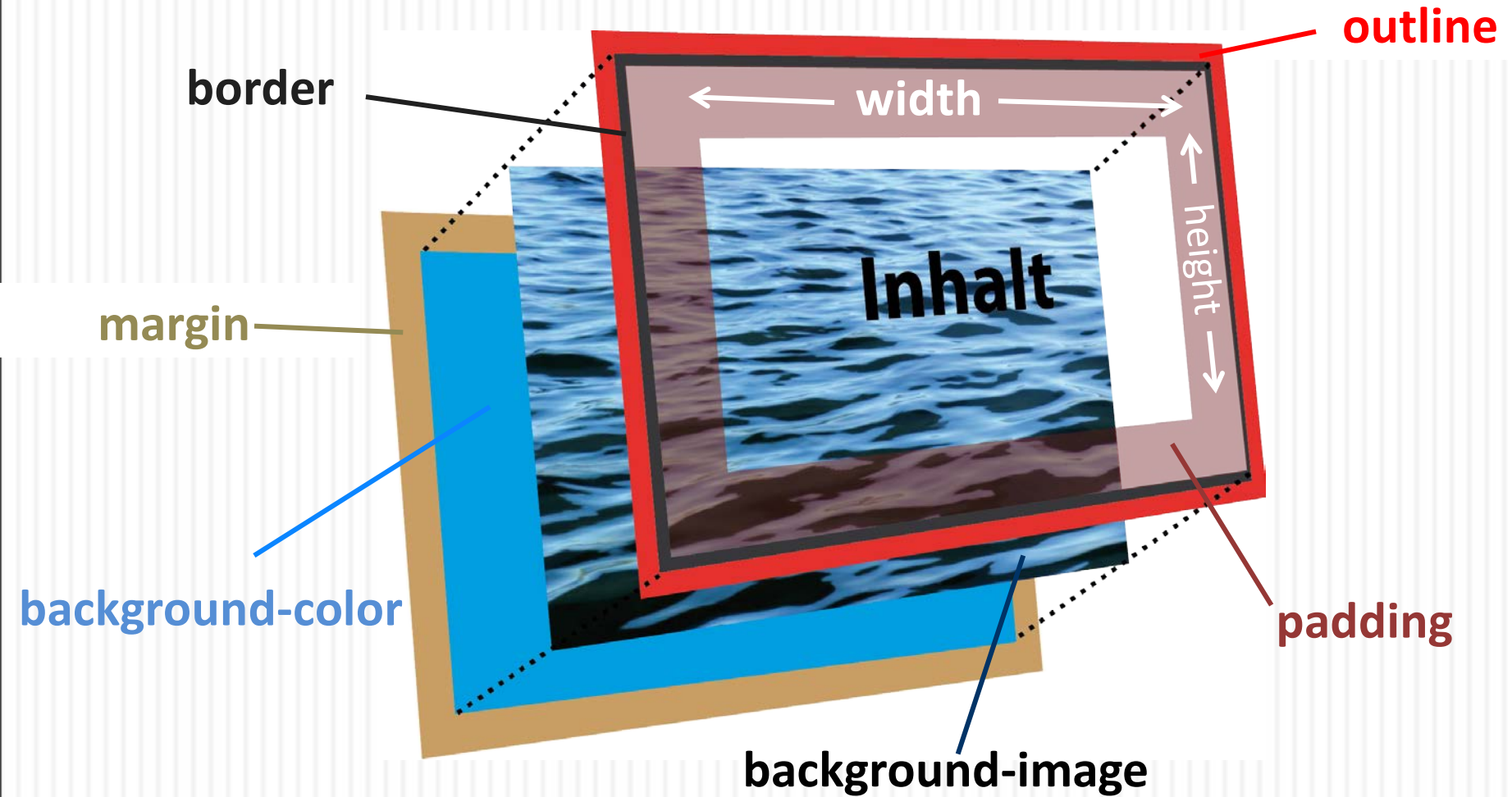
- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- class- und id-Selektoren
- Gruppierte, verschachtelte, attributabhängige Selektoren
- Möglichkeiten der Einbindung und Priorität
- Gestaltungs-Möglichkeiten mit CSS
- **Das CSS-Boxmodell**
- Pseudoklassen
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# CSS: Das Boxmodell

Mit dem Boxmodell sollen die folgenden CSS-Eigenschaften kompakt und einprägsam dargestellt werden:

- Dimensionen
- Hintergrund
- Rahmen
- Innenabstand
- Außenabstand
- Kontur

# CSS: Das Boxmodell



# CSS: Das Boxmodell

Der HTML-Code:

```
<div>  
  Inhalt in Form  
  von ...  
</div>
```



**Inhalt in Form  
von Text, Link  
oder Bildern**

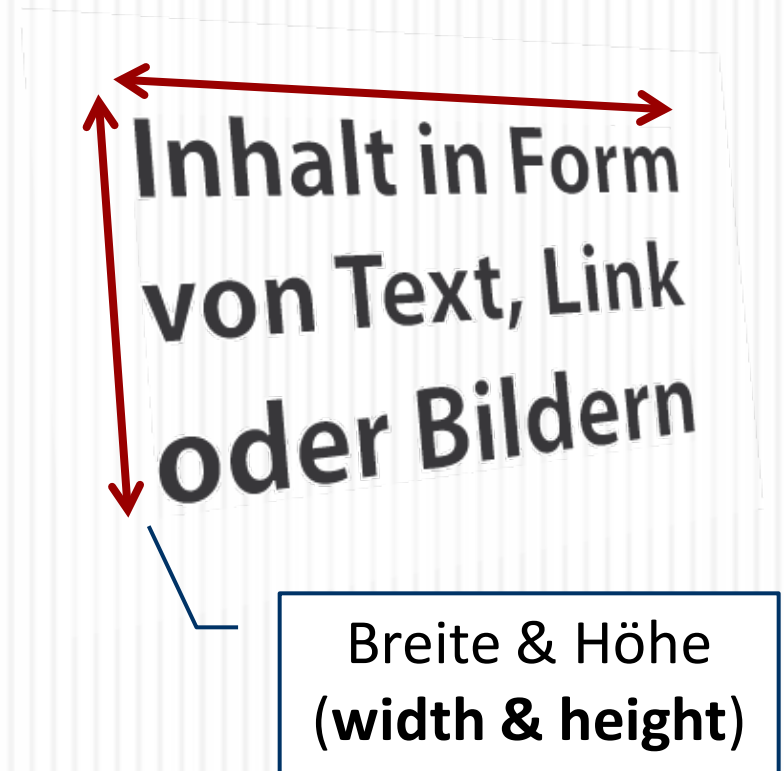
**Höhe** der Box wird vom **Inhalt** bestimmt  
**Breite** bei Blockelementen: **100%**

# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width: 300px;  
  height: 240px;  
}
```

**Dimensionen**





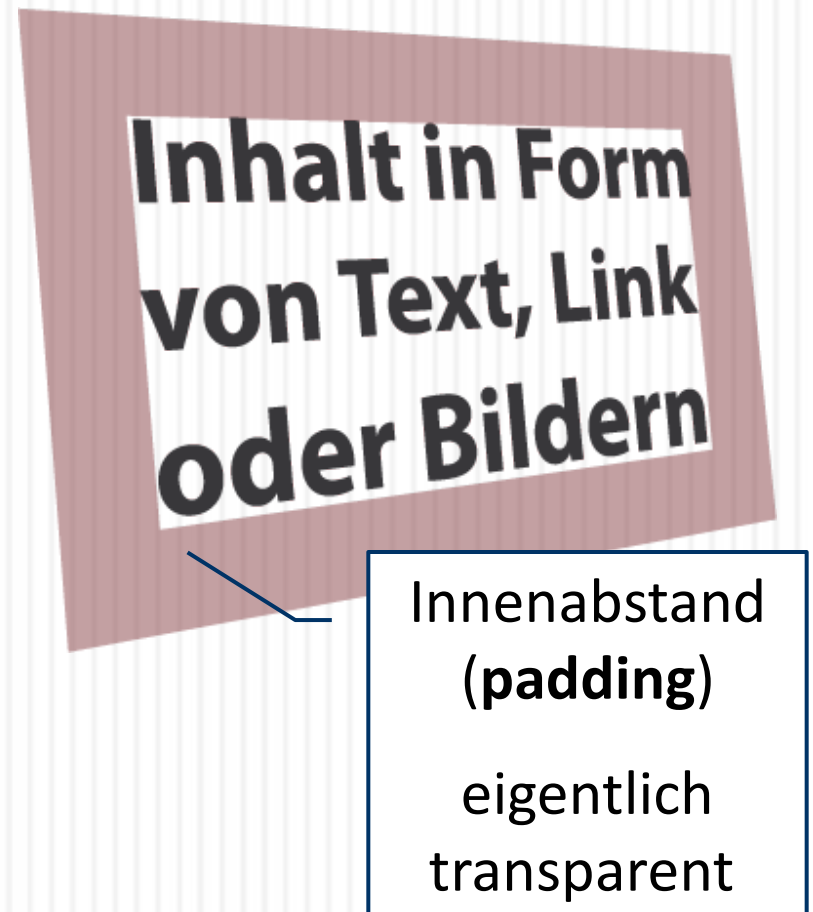
# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width: 300px;  
  height: 240px;  
  padding: 30px;  
}
```

**Innenabstand**

Gesamtbreite	x	Gesamthöhe
330	x	270



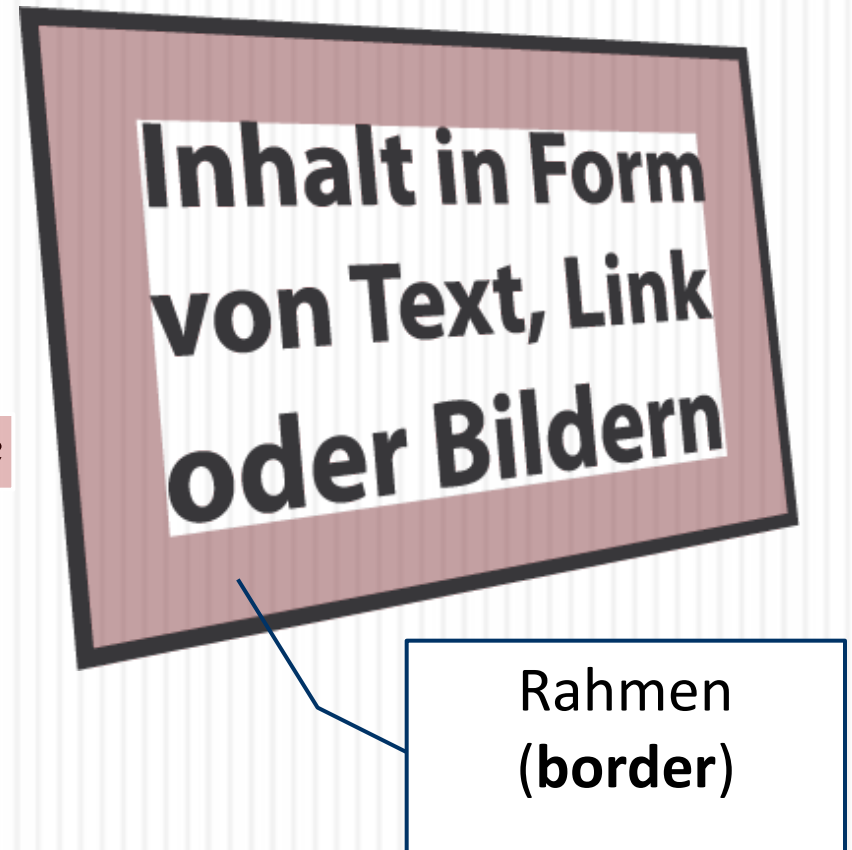
# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width:300px;  
  height:240px;  
  padding:30px;  
  border:3px solid brown;  
}
```

## Rahmen

Gesamtbreite	x	Gesamthöhe
333	x	273

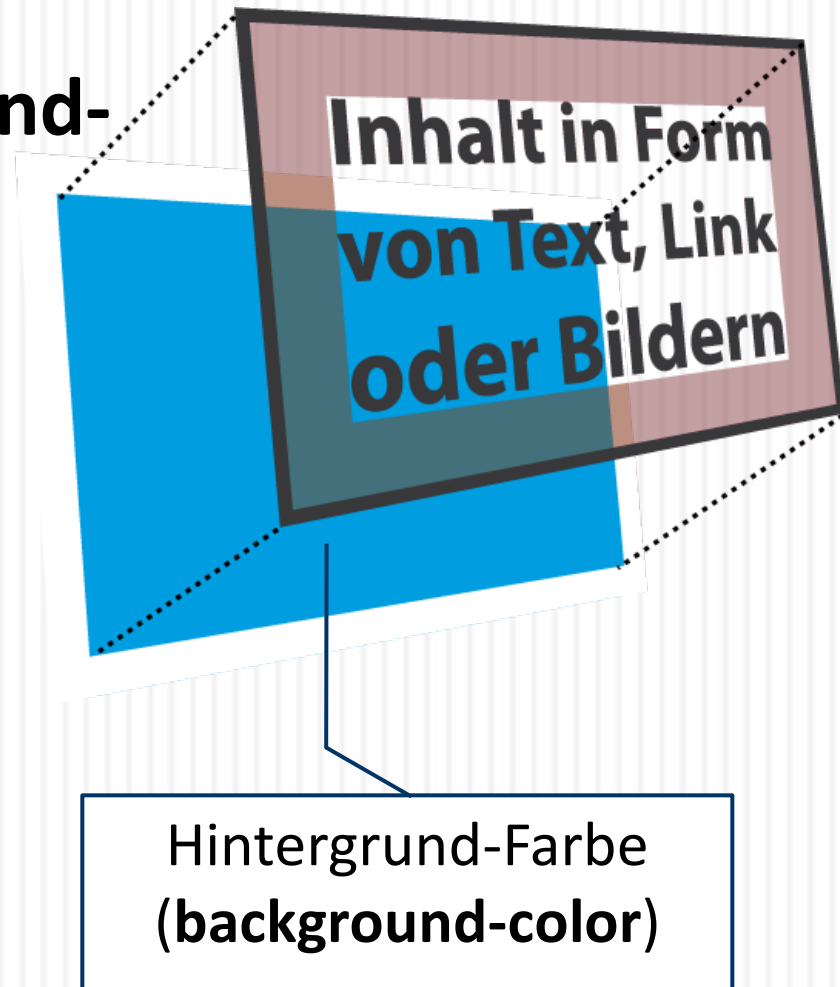


# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width:300px;  
  height:240px;  
  padding:30px;  
  border:3px solid brown;  
  background-color:blue;  
}
```

**Hintergrund-  
Farbe**



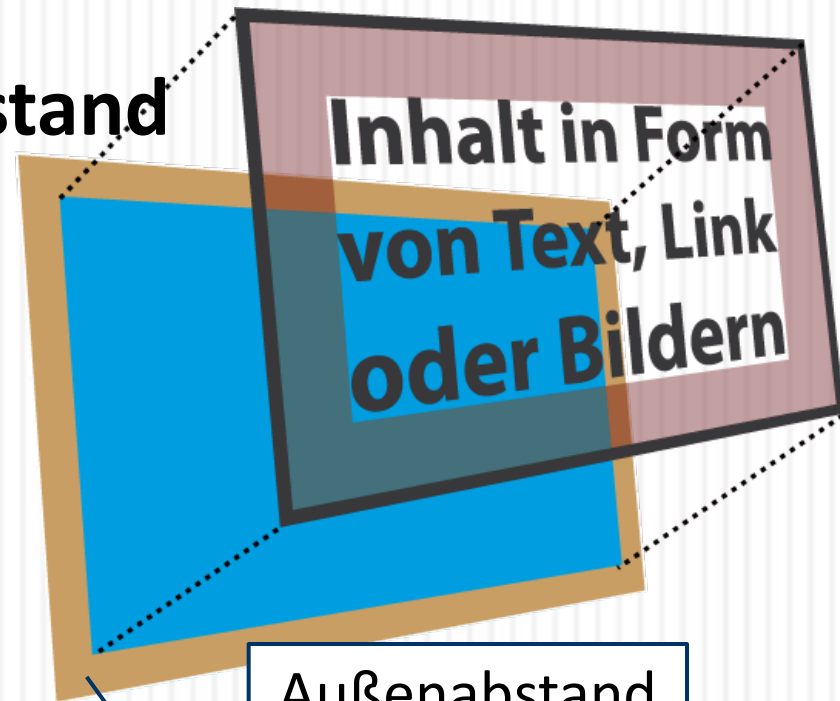
# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width:300px;  
  height:240px;  
  padding:30px;  
  border:3px solid brown;  
  background-color:blue;  
  margin:15px;  
}
```

Gesamtbreite	x	Gesamthöhe
333	x	273

**Außenabstand**



**Außenabstand  
(margin)**

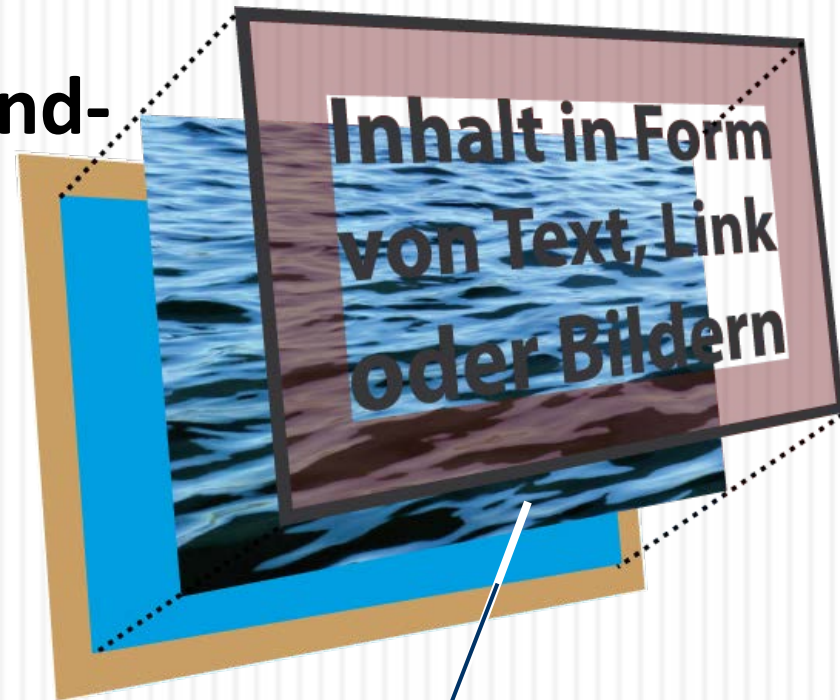
eigentlich  
transparent

# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width:300px;  
  height:240px;  
  padding:30px;  
  border:3px solid brown;  
  background-color:blue;  
  margin:15px;  
  background-image: url(meer.jpg);  
}
```

**Hintergrund-  
Bild**



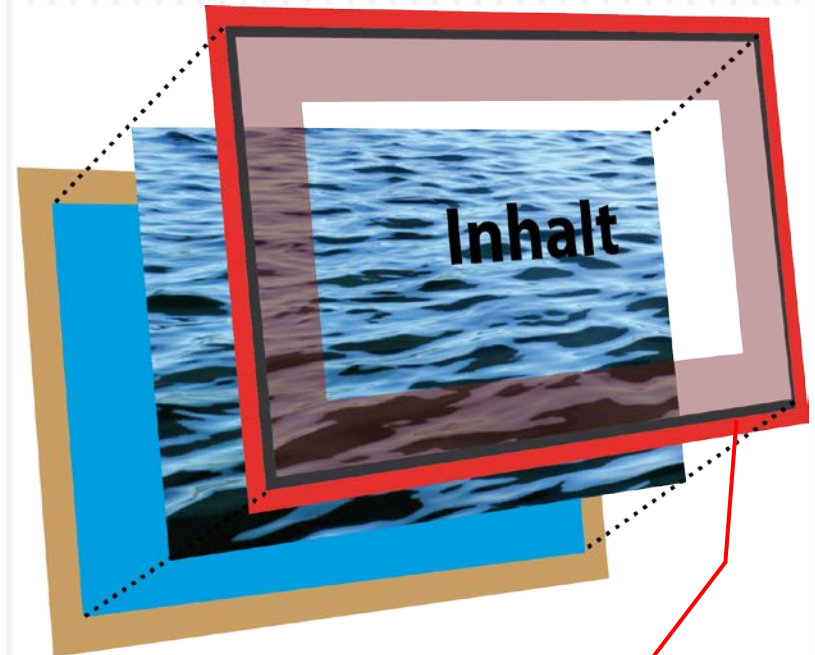
Hintergrund-Bild  
(background-image)

# CSS: Das Boxmodell

Der CSS-Code:

```
div {  
  width:300px;  
  height:240px;  
  padding:30px;  
  border:3px solid brown;  
  background-color:blue;  
  margin:15px;  
  background-image: url(meer.jpg);  
  outline:2px solid red;  
}
```

**Kontur**

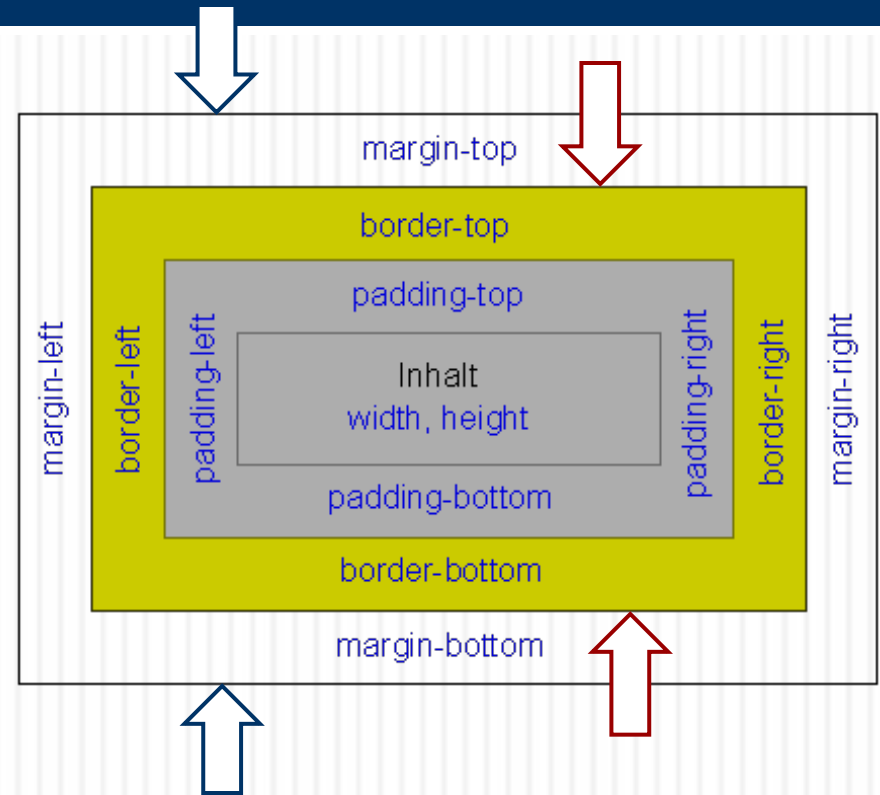


Kontur  
(outline)

# CSS: Das Boxmodell

## Abmessungen der Box

Bei der **Höhe** eines Elements wird zwischen "in **Anspruch genommener** Höhe" und der "**sichtbaren** Höhe" unterschieden. Diese setzen sich laut der CSS-Regel wie folgt zusammen:

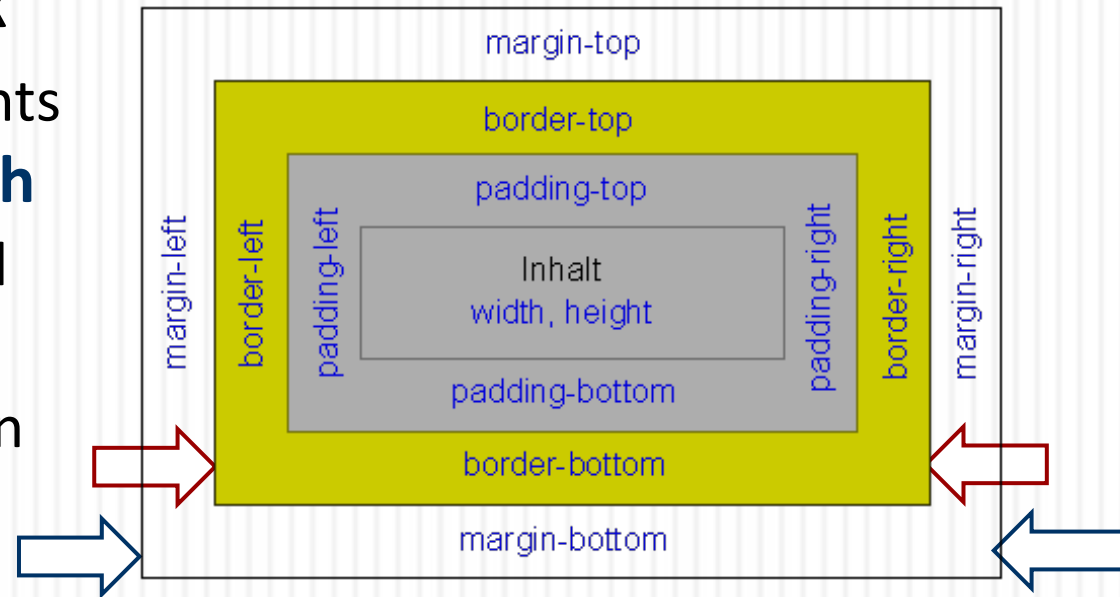


**In Anspruch genommene Höhe** = oberer Außenabstand + obere Rahmenbreite + oberer Innenabstand + Höhe des Inhalts + unterer Innenabstand + untere Rahmenbreite + unterer Außenabstand (ohne outline)

# CSS: Das Boxmodell

## Abmessungen der Box

Bei der **Breite** eines Elements wird zwischen "in **Anspruch genommener** Breite" und der "**sichtbaren** Breite" unterschieden. Diese setzen sich laut der CSS-Regel wie folgt zusammen:



**In Anspruch genommene Höhe** = linker Außenabstand + linke Rahmenbreite + linker Innenabstand + **Breite des Inhalts** + rechter Innenabstand + rechte Rahmenbreite + rechter Außenabstand (ohne outline).



# CSS: Das Boxmodell

Für jede **Seite** sind für die Eigenschaften

- margin und
- padding

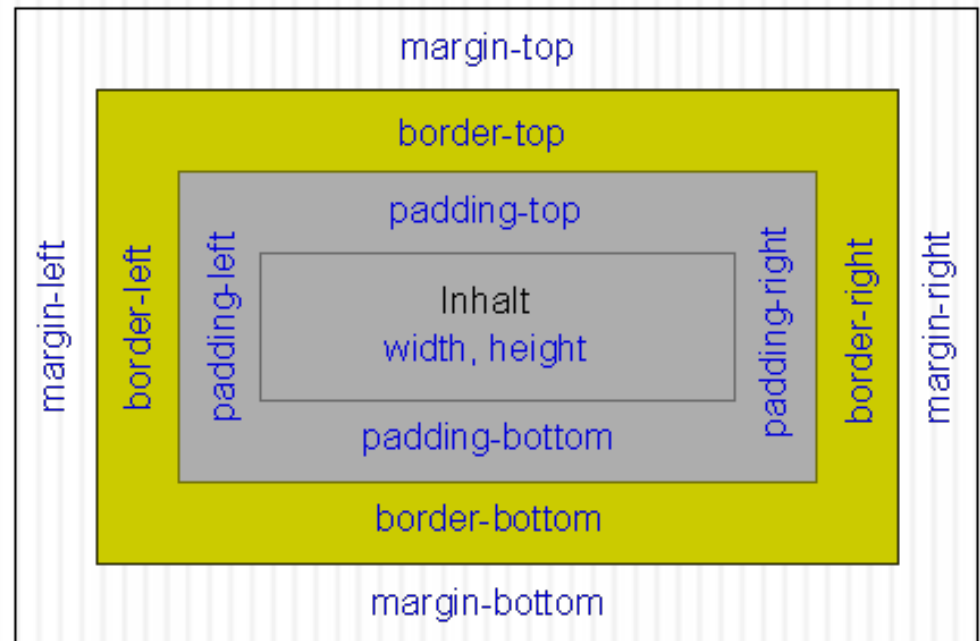
vier **separate** Eigenschaften vorgesehen:

*attribut-left*,  
*attribut-right*

*attribut-top*

*attribut-bottom*

*attribut-left*,  
*attribut-right*



# CSS: Das Boxmodell

Man kann auch für die Eigenschaften **margin** und **padding** die allgemeine Form in **Kurzschreibweise** verwenden und ein oder mehrere Werte in **bestimmter Reihenfolge** durch **Leerzeichen** getrennt angeben:

```
margin: 10px;  
padding: 10px;
```

Eine Angabe bedeutet: **alle** vier Seiten des Elements erhalten den gleichen Außenabstand.

# CSS: Das Boxmodell

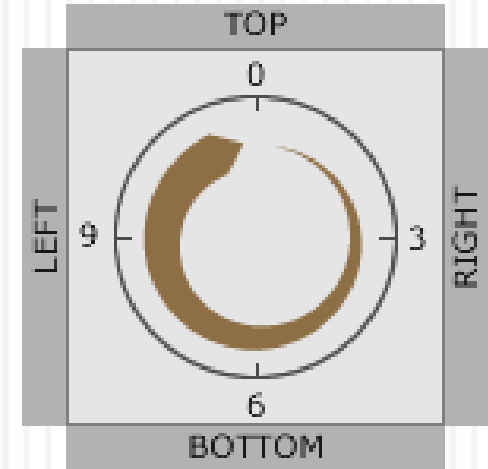
Bei vier Angaben:

```
margin: 10px 20px 30px 40px;
```

```
padding: 10px 20px 30px 40px;
```

Die Reihenfolge ist wie in der Grafik rechts dargestellt festgelegt:

*top – right – bottom – left*



- |         |          |
|---------|----------|
| 1 Top   | 3 Bottom |
| 2 Right | 4 Left   |

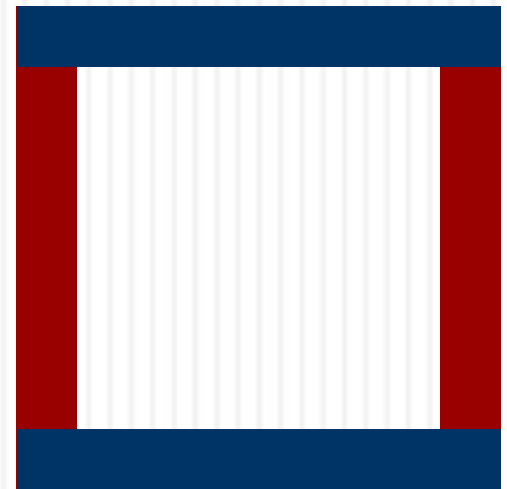
# CSS: Das Boxmodell

Bei zwei Angaben:

```
margin: 10px 20px;  
padding: 10px 20px;
```

Die erste Angabe bedeutet den Abstand für oben und unten, die zweite den Abstand für rechts und links.

*top/bottom* – *right/left*



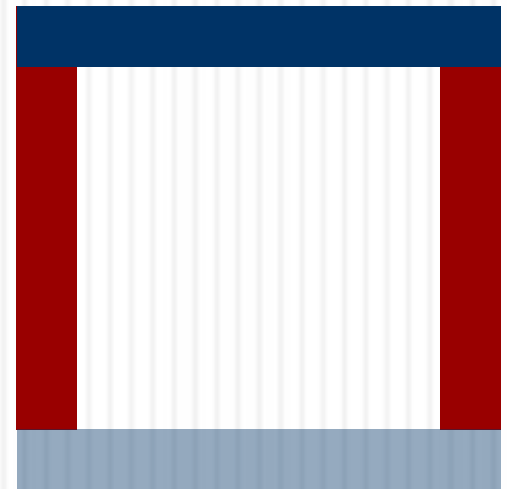
# CSS: Das Boxmodell

Bei drei Angaben:

```
margin: 10px 20px 30px;  
padding: 10px 20px 30px;
```

die erste Angabe bedeutet den Abstand für oben, die zweite den Abstand für rechts und links und die dritte den Abstand für unten.

*top – right/left – bottom*



# CSS: Das Boxmodell

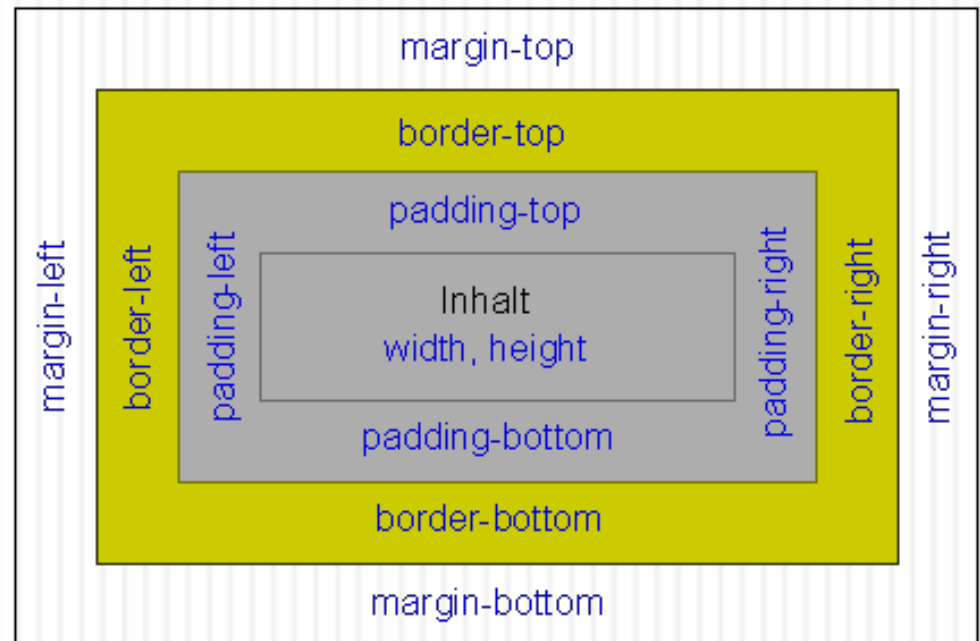
Auch bei **border** kann man jede **Seite** separat gestalten:

*border-left*

*border-right*

*border-top*

*border-bottom*



# CSS: Das Boxmodell

Ein **Rahmen** benötigt jedoch **3 Angaben**. Erst wenn diese vorhanden sind, wird der Rahmen auch **sichtbar**:

*border-**width*** (Rahmenstärke)

*border-**style*** (Rahmenart)

*border-**color*** (Rahmenfarbe)

## CSS Border Properties

The CSS border properties allow you to specify the style and color of an element's border.

# CSS: Das Boxmodell

*border-**width*** (Rahmenstärke)

Die **Rahmenstärke** wird durch einen entsprechenden **Pixel-Wert** angegeben oder durch einen der drei **vordefinierten** Werte **thin**, **medium**, oder **thick**.

```
border-width: medium;
```

```
border-width: 16px;
```



# CSS: Das Boxmodell

*border-color* (Rahmenfarbe)

Der Wert der **Rahmenfarbe** kann wie folgt gesetzt werden:

Farbname: z.B. "**red**"

Dezimaler RGB – Wert: z.B. "**rgb( 255 , 0 , 0 )**"

Hexadezimaler RGB – Wert: z.B. "**#ff0000**"

# CSS: Das Boxmodell

Für die Rahmenart gibt es die Werte:

- ❑ solid = durchgezogen
- ❑ double = doppelt
- ❑ none = kein Rahmen (unsichtbar)
- ❑ hidden = kein Rahmen (unsichtbar)
- ❑ dotted = gepunktet
- ❑ dashed = gestrichelt
- ❑ groove = 3D-Effekt
- ❑ ridge = 3D-Effekt
- ❑ inset = 3D-Effekt
- ❑ outset = 3D-Effekt

*border-**style** (Rahmenart)*

# CSS: Das Boxmodell

Jede Seite kann individuell gestaltet werden:

```
border-top-width: 5px;  
border-top-style: dotted;  
border-top-color: black;
```

```
border-right-width: 3px;  
border-right-style: solid;  
border-right-color: green;
```

```
border-bottom-width: 7px;  
border-bottom-style: dashed;  
border-bottom-color: red;
```

```
border-left-width: 44px;  
border-left-style: groove;  
border-left-color: blue;
```

# CSS: Das Boxmodell

## Kurzschreibweise (Shorthand Property)

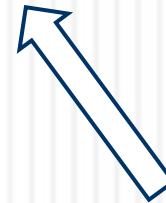
**border: 5px solid red;**

Reihenfolge der Attribute:

**border-width**

**border-style**

**border-color**



# CSS: Das Boxmodell

Oder für jeder Seite separat:

```
border-top: 5px dotted black;
```

=

```
border-top-width: 5px;  
border-top-style: dotted;  
border-top-color: black;
```

# CSS: Das Boxmodell

Auch das hier ist möglich:

```
border-width: 5px;
```

```
border-width: 5px 6px;
```

```
border-width: 5px 6px 7px;
```

```
border-width: 5px 6px 7px 8px;
```

Wobei dieselben Regeln gelten wie bei padding/margin

# CSS: Das Boxmodell

Auch das hier ist möglich:

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

```
border-style:solid dotted;
```

```
border-style:solid dotted dashed;
```

```
border-style:solid dotted dashed groove;
```

Wobei dieselben Regeln gelten wie bei padding/margin

# CSS: Das Boxmodell

Auch das hier ist möglich:

```
border-color:red;
```

```
border-color:red blue;
```

```
border-color:red blue green;
```

```
border-color:red blue green black;
```

Wobei dieselben Regeln gelten wie bei padding/margin



# CSS: Das Boxmodell

Um den **Hintergrund** einer "Box" zu gestalten, stehen die folgenden Attribute zur Verfügung:

*background-color* (Hintergrundfarbe)

*background-image* (Hintergrundbild)

*background-repeat* (Wiederholungs-Effekt)

*background-attachment* (Wasserzeichen-Effekt)

*background-position* (Hintergrundposition)

# CSS: Das Boxmodell

*background-color* (Hintergrundfarbe)

Der Wert der **Hintergrundfarbe** kann wie folgt gesetzt werden:

**Farbname:** z.B. "red"

**Dezimaler RGB – Wert:** z.B. "rgb(255,0,0)"

**Hexadezimaler RGB – Wert:** z.B. "#ff0000"

# CSS: Das Boxmodell

*background-image* (Hintergrundbild)

Legt für ein Element ein **Hintergrundbild** fest. Dieses werden immer über den **gesamten** Bereich des **Inhalts** und des **Innenabstands** angezeigt und vertikal und horizontal **wiederholt** (gekachelt).

```
background-image: url( 'paper.gif' );
```

# CSS: Das Boxmodell

*background-repeat* (Wiederholungs-Effekt)

Das **Wiederholungsverhalten** einer Hintergrundgrafik kann kontrolliert werden. Erlaubt ist eine der folgenden Angaben:

**repeat** = wiederholen (Voreinstellung)

**repeat-x** = nur "eine Zeile lang" waagerecht wiederholen

**repeat-y** = nur "eine Spalte lang" senkrecht wiederholen

**no-repeat** = nicht wiederholen (Einzelbild)

**background-repeat: repeat-x;**

# CSS: Das Boxmodell

## *background-attachement* (Wasserzeichen-Effekt)

Man kann erzwingen, dass der Hintergrund beim Scrollen stehen bleibt (**Wasserzeichen-Effekt**) und nicht mit scrollt. Erlaubt sind die Angaben:

**scroll** = mitscrollen (Voreinstellung), orientiert sich an der Position des jeweiligen Elements

**fixed** = Hintergrundbild bleibt stehen, orientiert sich am Viewport

**background-attachement: fixed;**

# CSS: Das Boxmodell

*background-**position*** (Hintergrundposition)

Man kann die **Position** der **linken oberen** Ecke der Hintergrundgrafik festlegen. Der **erste** Wert steht für die **horizontale**, der **zweite** für die **vertikale** Position. Bezugspunkt ist das HTML-Element, für das die Hintergrundgrafik definiert wird.

**background-*position*: 20px 30px;**

# CSS: Das Boxmodell

*background-**position*** (Hintergrundposition)

Erlaubt sind **numerische** Werte (z.B. px) und folgende Angaben:

**left** = horizontal linksbündig

**right** = horizontal rechtsbündig

**top** = vertikal oben bündig

**bottom** = vertikal unten bündig

**center** = zentriert (horizontal/vertikal)

**background-**position**: *right bottom*;**

# CSS: Das Boxmodell

## Kurzschreibweise (Shorthand Property)

`background: red url('bild.png') no-repeat fixed right top`

Reihenfolge  
der Attribute:

**color**

**-image**

**repeat**

**attachment**

**position**

background-



# 4. Veranstaltung

## Inhalt CSS:

- Definition, Ziel, Versionen
- Syntax und Kommentare in CSS
- class- und id-Selektoren
- Gruppierte, verschachtelte, attributabhängige Selektoren
- Möglichkeiten der Einbindung und Priorität
- Gestaltungs-Möglichkeiten mit CSS
- Das CSS-Boxmodell
- **Pseudoklassen**
- Die tags 'div' und 'span' im Zusammenhang mit CSS

# Pseudoklassen

Sie können das Erscheinungsbild von **Verweisen**

- zu noch **nicht besuchten** Seiten (**:link**),
- zu **bereits besuchten** Seiten (**:visited**)

und zu **Elementen**, die

- per Tastatur **selektiert** (**:focus**),
- gerade mit der **Maus überfahren** (**:hover**) oder
- **angeklickt** (**:active**) werden,

bestimmen.

# Pseudoklassen

- ❑ **Pseudoklassen** werden **zentral** in einem style-Bereich notiert.
- ❑ **Pseudoklassen** gelten teilweise **nur für das a-Element** in HTML, daher wird vor dem Doppelpunkt das a notiert.
- ❑ Die Pseudoklassen :focus, :hover und :active gelten auch für **andere** Elemente als Verweise.

# Pseudoklassen

```
<style type="text/css">
a:link { font-weight:bold; color:blue;}
a:visited { font-weight:bold; color:silver;}
a:focus { font-weight:bold; color:red;}
a:hover { font-weight:bold; color:green;}
a:active { font-weight:bold; color:lime;}

h1:focus { background-color:red;}
h1:hover { background-color:silver;}
h1:active { background-color:green;}
</style>
```

Yahoo! (Verzeichnis)  
Google (Suchmaschine)  
Web.de (Verzeichnis)  
Multimeta (Meta-Suchmaschine)

# Die tags 'div' und 'span'

Die beiden HTML-Elemente `<div>` und `<span>` haben eine **besondere** Bedeutung im Zusammenhang mit CSS: Sie sind nahezu **eigenschaftslos** ("tabula rasa")

Einziger Unterschied:

- ❑ `<div>` **erzwingt** eine **neue Zeile** im Textfluss
- ❑ `<span>` kann **innerhalb** eines Textes verwendet werden und erzeugt **keinen** Absatz



# Die tags 'div' und 'span'

Häufig werden diese Tags benutzt, um vermeintliche Probleme mit **voreingestellten Eigenschaften** bestimmter Elemente zu vermeiden (z.B. <h1> ).

Dies ist jedoch eine **Zweckentfremdung**, da dadurch die **Semantik** von HTML **nicht** mehr gewährleistet ist und somit **keine Barrierefreiheit** mehr besteht.

Am meisten leidet aber **der wohl bekannteste, blinde Besucher** unter fehlender Semantik: **die Suchmaschine!**

# Die tags 'div' und 'span'

---

**Semantisch** haben 'div' und 'span' nahezu **keine** Relevanz. Das div-tag für steht aber in diesem Zusammenhang für

Bereich/Teilung/Abtrennung/Aufgliederung

und sollte auch **nur** für diesen **Zweck** eingesetzt werden (z.B. Navigationsbereich).

# Die tags 'div' und 'span'

```
<head>
  <meta charset = "utf-8" />
  <title>CSS-Beispiel 1</title>
  <style type="text/css">
```

```
    div.ueberschrift
```

```
    {
      font-size:30px;
      font-weight: bold;
      color:#990000;
      margin: 20px 0px;
    }
```

```
  </style>
```

```
</head>
```

```
<body>
```

```
  <div class = "ueberschrift">Dies ist eine Überschrift</div>
```

```
  <p>Dies ist ein Abschnitt 1</p>
```

```
  <p>Dies ist ein Abschnitt 2</p>
```

```
</body>
```

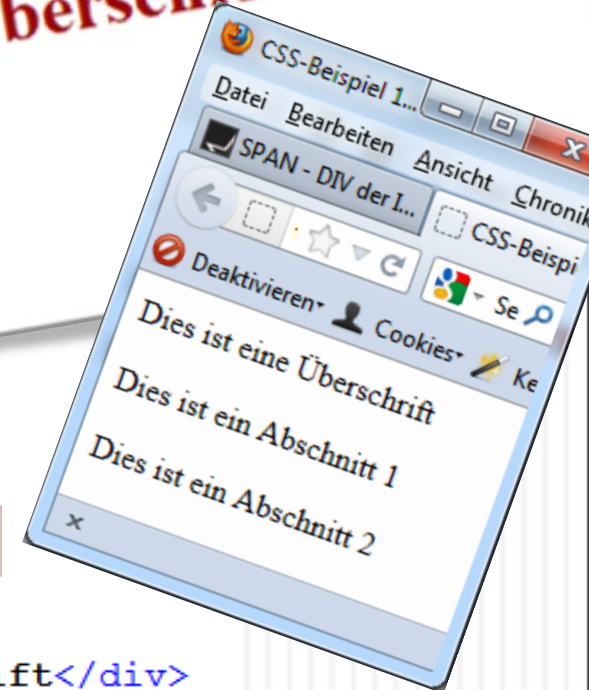
```
</html>
```

**Dies ist eine Überschrift**

Dies ist ein Abschnitt 1

Dies ist ein Abschnitt 2

So sieht die Suchmaschine die Seite:





# Die tags 'div' und 'span'

```
<head>
  <meta charset = "utf-8" />
  <title>CSS-Beispiel 1</title>
  <style type="text/css">
```

h1.ueberschrift

```
{
  font-size:30px;
  font-weight: bold;
  color:#990000;
  margin: 20px 0px;
}
```

```
</style>
```

```
</head>
```

```
<body>
```

```
<h1 class = "ueberschrift">Dies ist eine Überschrift</h1>
```

```
<p>Dies ist ein Abschnitt 1</p>
```

```
<p>Dies ist ein Abschnitt 2</p>
```

```
</body>
```

```
</html>
```

**Dies ist eine Überschrift**

Dies ist ein Abschnitt 1  
Dies ist ein Abschnitt 2

So sieht die Suchmaschine die Seite:



# Die tags 'div' und 'span'

- span hat nur noch **wenig Bedeutung**, so dass es teilweise kategorisch vom Einsatz auf einer "sauberen" Website ausgeschlossen wurde
- **Nicht sinnvoll** ist z.B. der Einsatz zur Darstellung von **Zitaten etc.** mit entsprechend gestalteten span-Elementen:

`<p>Ein Zitat:`

`<cite>`

Hier steht etwas  
Gescheites

`</cite></p>`



`<p>Hier folgt ein Zitat`

`<span style="font-style:italic;">`

Hier steht etwas  
Gescheites

`</span></p>`



# Die tags 'div' und 'span'

## Fazit:

span-tags dürfen eingesetzt werden, aber das **oberste Gebot**, wie auch bei den div-tags, lautet:

**Zuerst** über zur Verfügung stehende **Elemente** nachdenken. Sieht man dann **keine andere** Lösung, darf über den **sparsamen** Einsatz von SPAN nachgedacht werden.