

Verteilte Systeme 1

Technologien des World Wide Web

christian.zirpins@hs-karlsruhe.de

Gestaltung von Webapps mit CSS3



Hochschule Karlsruhe
Technik und Wirtschaft

UNIVERSITY OF APPLIED SCIENCES



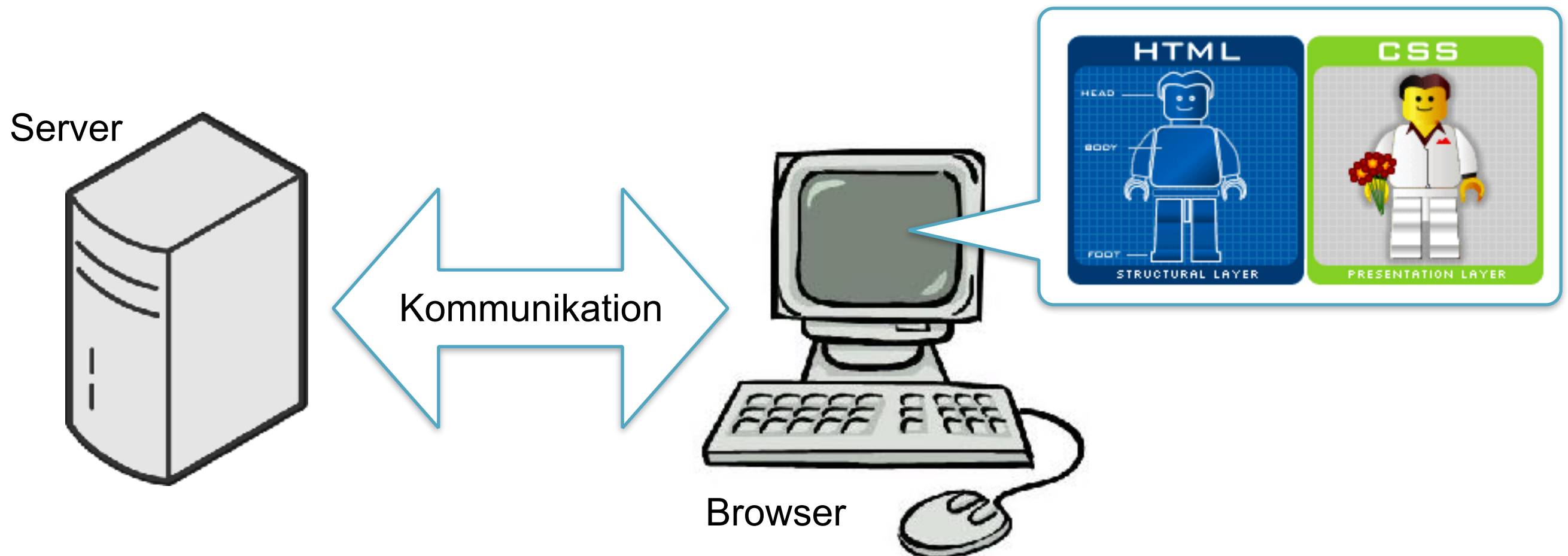
VS1 Termine im Sommer 2017

Termin (ca.)	Thema	Vorbereitung (Begleitbuch)	Raum
KW11: 13.03, 15.03	HTTP , die Sprache des Web		E 301/304
KW12: 20.03, 22.03	Web Apps mit HTML5	Web App Development Kapitel 2	E 301/304
KW13: 27.03, 29.03	Gestaltung von Web Apps mit CSS3	Web App Development Kapitel 3	E 301/304
KW14: 03.04, 05.04	<i>Übung: Webseite mit HTML5/CSS3 erstellen</i>		LI 137
KW15: 10.04, 12.04	Browser Interaktion mit JavaScript	Web App Development Kapitel 4	E 301/304
KW16	Ostern		
KW17: 24.04, 26.04	<i>Übung: Formulare mit JavaScript / HTML5 APIs</i>		LI 137
KW18	Maifeiertag		
KW19: 08.05, 10.05	JavaScript auf dem Server mit Node.js	Web App Development Kapitel 6	E 301/304
KW20: 15.05, 17.05	<i>Übung: Node.js / Express Web App erstellen</i>		LI 137
KW21: 22.05, 24.05	Web Entwicklung mit Ajax & Co	Web App Development Kapitel 5	E 301/304
KW22: 29.05, 31.05	<i>Übung: Web App mit REST und AJAX erweitern</i>		LI 137
KW23	Pfingsten		
KW24: 12.06, 14.06	Web Apps Personalisieren	Web App Development Kapitel 9	E 301/304
KW25: 19.06, 21.06	Web App Sicherheit		E 301/304
KW26: 26.06, 28.06	Klausurvorbereitung, Q&A		E 301/304

Heutige Lernziele

Nach dieser Vorlesung können Sie...

- ...HTML-Elemente nach einem **Design** positionieren und darstellen
- ...das **Boxmodell** erklären
- ...**Pseudoklassen** und **Pseudoelemente** verwenden
- ...CSS Mechanismen für **Datenzugriff/Datenerstellung** nutzen



Etwas **Kontext**

Geschichte von CSS

- **CSS 1:** W3C Empfehlung 1996
 - Unterstützt Schriften, Farben, Ausrichtung, Ränder, IDs und Klassen
- **CSS 2:** W3C Empfehlung 1998
 - Unterstützung für Media Queries, Positionierung von Elementen
- **CSS 2.1:** W3C Empfehlung 2011
 - Fehlerbehebung, Unterstützung für neue Funktionen, die in den verbreiteten Browsern implementiert wurden
- **CSS 3:** momentan in Entwicklung; Spezifikation geteilt
 - In Module; Fortschritt variiert zwischen Modulen
- **CSS 4:** einige Module haben “Level 4” Status erreicht

<http://www.w3.org/Style/CSS/current-work>

Module gibt es viele...

Mittlere Priorität	Aktuell	Zukünftig
CSS Backgrounds and Borders Level 4		WD
CSS Device Adaptation	WD	WD
CSS Exclusions	WD	WD
Filter Effects	WD	WD
CSS Generated Content for Paged Media	WD	WD
CSS Page Floats	FPWD	WD
CSS Template Layout	NOTE	NOTE
CSS Line Grid	WD	WD
CSS Lists Level 3	WD	WD
CSS Positioned Layout Level 3	WD	WD
CSS Regions	WD	WD
CSS Ruby Level 1	WD	WD
CSS Tables Level 3		WD
CSS Object Model	WD	WD
CSS Font Loading	LC	WD
CSS Scoping Level 1	FPWD	WD
Media Queries level 4	WD	WD
Non-element Selectors	FPWD	WD
CSS Inline Layout Module Level 3	WD	WD
Motion Path Module Level 1	FPWD	WD
CSS Round Display Level 1	WD	WD
CSS Basic User Interface Module Level 4	FPWD	WD
CSS Text Module Level 4	FPWD	WD
CSS Painting API Level 1	FPWD	WD
CSS Properties and Values API Level 1	FPWD	WD
CSS Typed OM Level 1	FPWD	WD
Worklets Level 1	FPWD	WD
CSS Color Module Level 4	FPWD	WD
CSS Fonts Level 4		WD

Codes, von am wenigsten bis am meisten stabil

Abkürzung	Bezeichnung
WD	Arbeitsentwurf
LC	Letzter Aufruf
CR	Veröffentlichungsvorschlag
PR	Geplante Veröffentlichung
REC	Standard

Abgeschlossene Werke	Aktuell	Zukünftig
CSS Snapshot 2010	NOTE	
CSS Snapshot 2007	NOTE	
CSS Color Level 3	REC	REC
CSS Namespaces	REC	REC
Selectors Level 3	REC	REC
CSS Level 2 Revision 1	REC	REC
CSS Level 1	REC	
CSS Print Profile	NOTE	
Media Queries	REC	REC
CSS Style Attributes	REC	REC

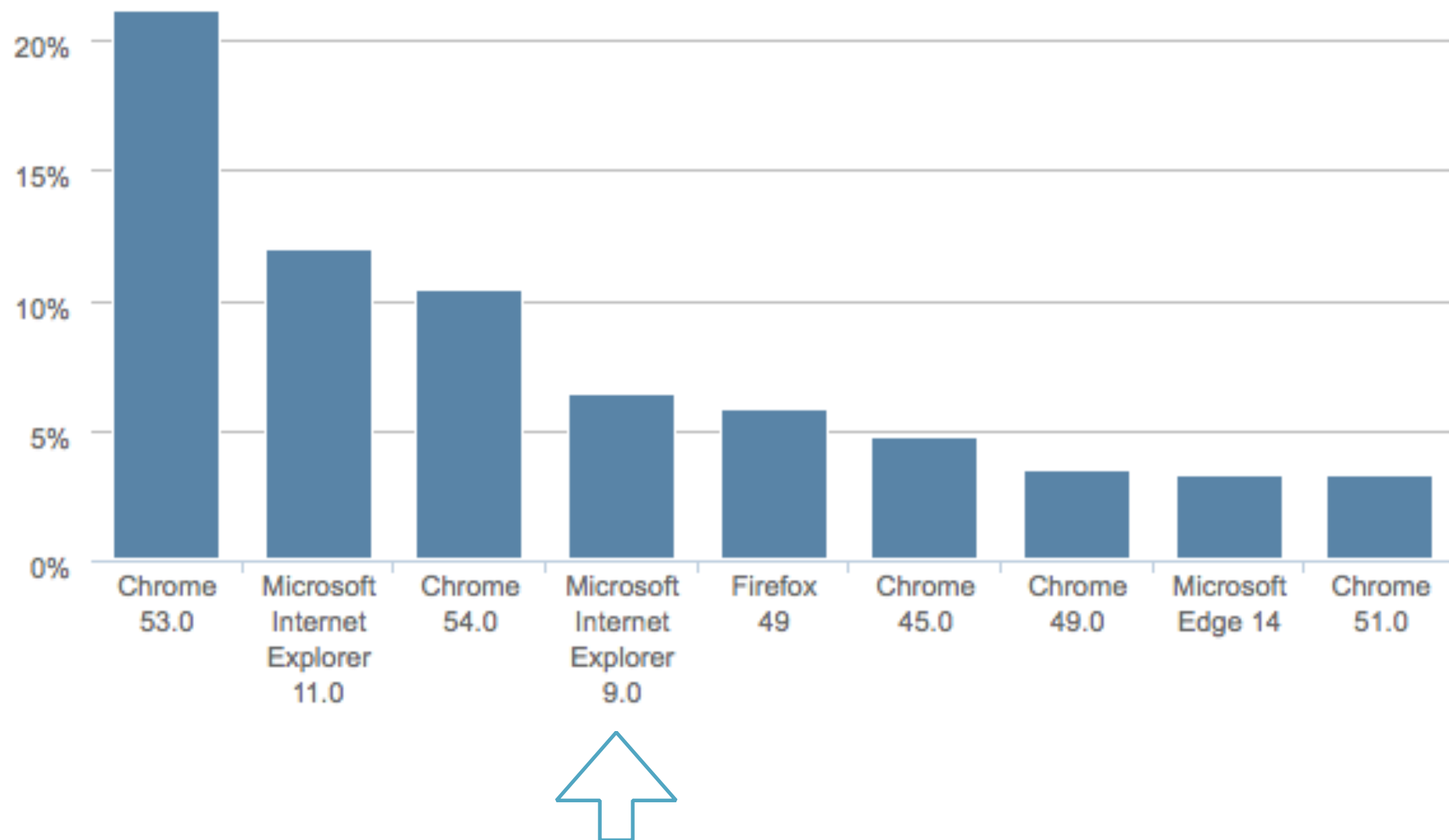
Hohe Priorität	Aktuell	Zukünftig
CSS Backgrounds and Borders Level 3	CR	PR
CSS Conditional Rules Level 3	CR	CR
CSS Multi-column Layout	CR	CR
CSS Values and Units Level 3	CR	PR
CSS Cascading and Inheritance Level 3	CR	PR
CSS Snapshot 2015	NOTE	
CSS Image Values and Replaced Content Level 3	CR	CR
CSS Speech	CR	CR
CSS Flexible Box Layout	CR	PR
CSS Text Decoration Module Level 3	CR	CR
CSS Fonts Level 3	CR	CR
CSS Writing Modes Level 3	CR	CR
CSS Shapes Level 1	CR	CR
CSS Masking Level 1	CR	CR
CSS Counter Styles Level 3	CR	PR
CSS Fragmentation Level 3	CR	PR
Compositing and Blending Level 1	CR	CR
CSS Syntax Level 3	CR	CR
CSS Grid Layout Level 1	CR	PR
CSS Basic User Interface Level 3	CR	CR
Geometry Interfaces Module Level 1	CR	CR
CSS Cascading and Inheritance Level 4	CR	PR
CSS Scroll Snap Module Level 1	CR	PR

CSS 3

- Unmöglich CSS zu schreiben, das moderne Funktionen nutzt und auf allen Browsern funktioniert...
- Die Verwendung von CSS 3 Features sollte sich gründen auf...
 - **Nutzergruppe** (hauptsächlich in den USA oder China oder ...?),
 - **Nutzungsmodus** (Smartphone, Touchscreen oder Desktop?),
 - **Anwendungstyp** (3D-Animationen für Aufgabenliste, echt jetzt?)
- Es gibt JavaScript Bibliotheken, die Front-End-Entwicklern bei der Umsetzung von Cross-Browser Apps helfen (z.B. Modernizr)

Voraussetzung für unsere GeoTag Web App: sie sollte auf **3 großen aktuellen Browsern** arbeiten.

Sind alte Browser verbreitet?



(IE9 vom 15. März 2011 für Windows Vista u.a.)

Rückblick: Kapitel 3

Rückblick: Kapitel 3

CSS beschreibt, wie Elemente im DOM **dargestellt** werden sollen

```
body {  
  background-color: #ffee22;  
  width: 800px;  
  margin: auto;  
}
```

```
#the-button {  
  color: maroon;  
}
```

```
ul li {  
  color: gray;  
  border: 1px solid gray;  
}
```

```
p.last {  
  color: green;  
}
```

Selektor

Eigenschaft

Wert

CSS Datei

Frage: was tun die gezeigten Selektoren?

Rückblick: Kapitel 3

CSS beschreibt, wie Elemente im DOM **dargestellt** werden sollen

```
body {
  background-color: #ffee22;
  width: 800px;
  margin: auto;
}

#the-button {
  color: maroon;
}

ul li {
  color: gray;
  border: 1px solid gray;
}

p.last {
  color: green;
}
```

Selektor

Eigenschaft

Wert

CSS Datei

- Drei Arten von Stylesheets:
(1) Browser-Stylesheet
(2) Autorenstylesheet
(3) Stylesheet des Benutzers
- Stylesheets werden in Reihenfolge verarbeitet; spätere Deklarationen überschreiben frühere
- **!important** überschreibt alle anderen Deklarationen (sollte möglichst nicht verwendet werden)

↑
überschreibt

Pseudo-Elemente **und** Pseudo-Klassen

Pseudoklasse

Pseudoklasse: ein Schlüsselwort, das zu einem Selektor hinzugefügt wird und einen bestimmten Zustand des entsprechenden Elements angibt

Pseudoklassen (und Pseudoelemente) erlauben das Styling nach **externen** Faktoren (z. B. Mausbewegungen, Browserverlauf).

```
selector:pseudo-class {  
    property: value;  
    property: value;  
}
```

Generelle Syntax

Pseudoklasse

- Mehr als 30 Pseudoklassen
- Unterstützung variiert je nach **Rendering Engine**

Eine **Rendering-Engine** (oder Browser-Engine, Layout-Engine) ist Verantwortlich für die Darstellung von HTML + CSS (u.a.) auf dem Bildschirm

Rendering Engine	Browser
Gecko	Firefox
Trident	Internet Explorer
WebKit	Safari, ältere Chrome
Blink	neuere Chrome, Opera

Gängige Pseudoklassen

:hover ein Zeigegerät (Maus) schwebt über dem Element

:active das Element ist derzeit aktiv (z. B. angeklickt)

```
button {  
  background: white;  
  color: darkgray;  
  width: 100px;  
  padding: 5px;  
  font-weight: bold;  
  text-align: center;  
  border: 1px solid darkgray;  
}
```

```
button:hover {  
  color: white;  
  background: darkgray;  
}  
  
button:active {  
  border: 1px dashed black;  
}
```



Gängige Pseudoklassen

:enabled Element, das selektiert (z.B. angeklickt) werden kann

:disabled Element, das nicht selektiert werden kann

CSS

```
button {  
    ...  
}  
  
button:enabled:hover {  
    ...  
}  
  
button:enabled:active {  
    ...  
}
```

HTML

```
<button id="addyesterday" disabled>  
    ADD TODO  
</button>
```

- Enabled/Disabled Buttons sehen gleich aus
- Enabled Buttons ändern ihr aussehen wenn sie aktiviert werden oder beim darüber streichen

Gängige Pseudoklassen

`:nth-child(x)` alle n-ten Kind Elemente ihrer Eltern

`:nth-of-type(x)` alle n-ten Geschwisterelemente eines Typs

```
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```
p:nth-child(2) {
  color: red;
}

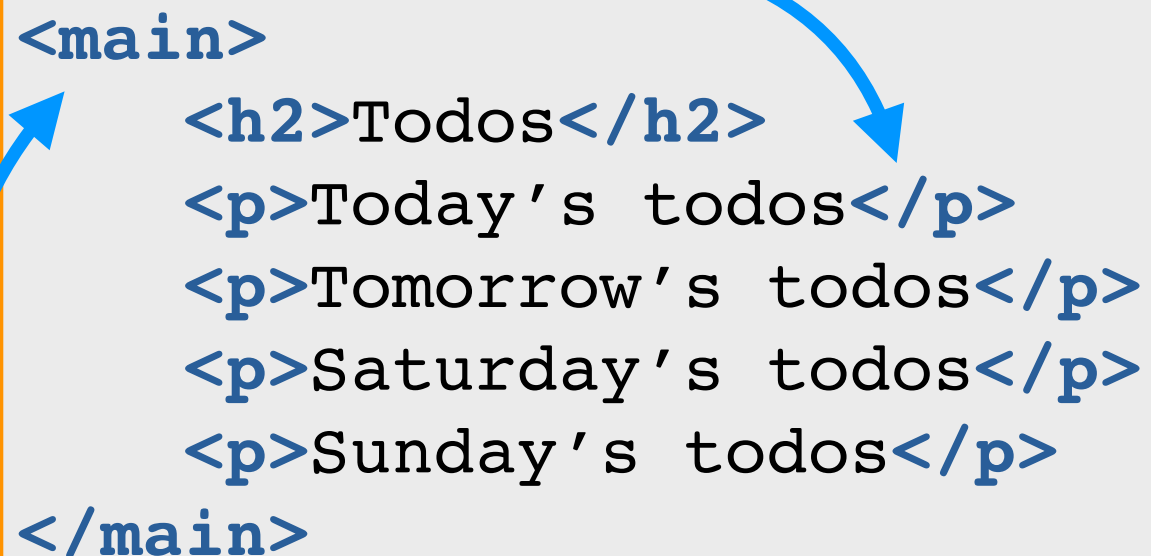
p:nth-of-type(2) {
  background-color: green;
}
```

Gängige Pseudoklassen

`:nth-child(x)` alle n-ten Kind Elemente ihrer Eltern

`:nth-of-type(x)` alle n-ten Geschwisterelemente eines Typs

2. Child



```
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

Parent

```
p:nth-child(2) {
  color: red;
}
```

```
p:nth-of-type(2) {
  background-color: green;
}
```

Gängige Pseudoklassen

:nth-child(x) alle n-ten Kind Elemente ihrer Eltern

:nth-of-type(x) alle n-ten Geschwisterelemente eines Typs

```
<main>
  <h2>Todos</h2>
  {
    <p>Today's todos</p>
    <p>Tomorrow's todos</p>
    <p>Saturday's todos</p>
    <p>Sunday's todos</p>
  }
</main>
```



Geschwister

```
p:nth-child(2) {
  color: red;
}

p:nth-of-type(2) {
  background-color: green;
}
```

Gängige Pseudoklassen

`:nth-child(x)` alle n-ten

`:nth-of-type(x)` alle n-ten

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

s Typs

```
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```
p:nth-child(2) {
  color: red;
}

p:nth-of-type(2) {
  background-color: green;
}
```

Gängige Pseudoklassen

`:nth-child(x)` alle n-ten Kind Element

`:nth-of-type(x)` alle n-ten Geschwister

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

s Typs

```
<main>
  <h2>Todos</h2>
  <p>Today's todos</p>
  <p>Tomorrow's todos</p>
  <p>Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```
p:nth-child(3) {
  color: red;
}

p:nth-of-type(4) {
  background-color: green;
}
```


Gängige Pseudoklassen

`:first-child` entspricht `:nth-child(1)`

`:last-child` entspricht `:nth-last-child(1)`

`:first-of-type` entspricht `:nth-of-type(1)`

`:last-of-type` entspricht `:nth-last-of-type(x)`

Gängige Pseudoklassen

:not(x) alle Elemente die nicht Selektor x entsprechen

```
<main>
  <h2>Todos</h2>
  <p id="firsttodo">Today's todos</p>
  <p class="todo">Tomorrow's todos</p>
  <p class="todo">Saturday's todos</p>
  <p>Sunday's todos</p>
</main>
```

```
main :not(.todo) {
  color: orange;
}
```

Todos

Today's todos

Tomorrow's todos

Saturday's todos

Sunday's todos

Gängige Pseudoklassen

`:in-range` `:out-of-range`

selektiert Elemente mit eingeschränktem Wertebereich

```
<main>
  <input type="text" placeholder="add your todo" />
  <input id="d1" name="d1" type="number" min="0"
    max="30" placeholder="Days to deadline" required />
  <label for="deadline1"> </label>
</main>
```

Geschwisterselektor

```
input[type=text] {
  border: 0px;
  width: 150px;
}
```

```
input[type=number] {
  width: 100px;
}
```

Attribut Selektor

```
input:valid + label::after {
  content: " \2714";
  color: rgba(0, 100, 0, 0.7);
}
```

Unicode

```
input:invalid + label::after {
  content: " (invalid)";
  color: rgba(255, 0, 0, 0.7);
}
```

RGB & Alpha

Gängige Pseudoklassen

`:in-range` `:out-of-range`

selektiert Elemente mit eingeschränktem Wertebereich

add your todo	Days to deadline	⌵	(invalid)
Lese Buch	12fdsdf	⌵	(invalid)
Lese Buch	123	⌵	(invalid)
Lese Buch	12	⌵	✓

Pseudoelemente

`::first-letter`

`::first-line`

- Typisches Beispiel: Erstes Zeichen/Zeile im Paragraph

```
p::first-line {  
    color: gray;  
    font-size: 125%;  
}  
  
p::first-letter {  
    font-size: 200%;  
}
```

`<p>`

To be, or not to be,
that is the question—

`</p>`

`<p>`

Whether 'tis Nobler in
the mind to suffer The
Slings and Arrows of
outrageous Fortune,...

`</p>`

Pseudoelemente

`::first-letter`

`::first-line`

- Typisches Beispiel: Erstes Zeichen/Zeile im Paragraph

```
p::first-line {  
    color: gray;  
    font-size: 125%;  
}  
  
p::first-letter {  
    font-size: 200%;  
}
```

To be, or not to be, that is the question—

Whether 'tis Nobler in the mind to suffer The Slings and Arrows of outrageous Fortune,...

`<p>`

To be, or not to be,
that is the question—

`</p>`

`<p>`

Whether 'tis Nobler in
the mind to suffer The
Slings and Arrows of
outrageous Fortune,...

`</p>`

Pseudoelemente

::after Inhalt hinter einem Element hinzufügen

::before Inhalt vor einem Element hinzufügen

```
<cite>
  To be, or not
  to be ...
</cite>
```

- Typisches Beispiel: Füge **Anführungszeichen** zu einem Zitat hinzu

```
cite::before {
  content: "\201C";
}

cite::after {
  content: "\201D";
}
```

“ To be, or not to be ... ”

Pseudoelemente

::after Inhalt hinter einem Element hinzufügen

::before Inhalt vor einem Element hinzufügen

- Auch das geht, die Ausgabe ist genau gleich

```
<cite>  
</cite>
```

CSS arbeitet mit Daten!?

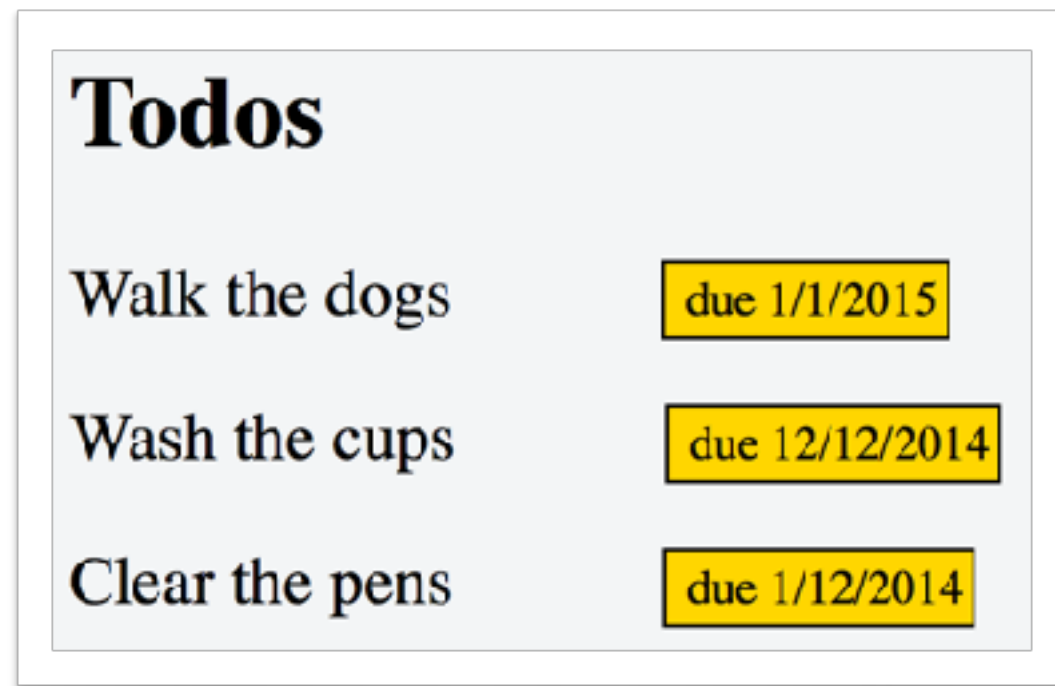
```
cite::before {  
    content: "\201CTo be, or ";  
}  
  
cite::after {  
    content: "not to be ... \201D";  
}
```

Daten in CSS

CSS & Daten (eine Möglichkeit)

CSS beschreibt nicht nur den Stil, es kann auch **Daten** enthalten

```
<main>
  <h2>Todos</h2>
  <p id="t1">Walk the dogs</p>
  <p id="t2">Wash the cups</p>
  <p id="t3">Clear the pens</p>
</main>
```



```
p::after {
  background-color: gold;
  border: 1px solid;
  font-size: 70%;
  padding: 2px;
  margin-left: 50px;
}

p#t1::after {
  content: " due 1/1/2015";
}

p#t2::after {
  content: " due 12/12/2014";
}

p#t3::after {
  content: " due 1/12/2014";
}
```

CSS & Daten (eine Möglichkeit)

CSS beschreibt nicht nur den Stil, es kann auch **Daten** enthalten

```
<main>
  <h2>Todos</h2>
  <p id="t1">Walk the dogs</p>
  <p id="t2">Wash the cups</p>
  <p id="t3">Clear the pens</p>
</main>
```

```
p::after {
  background-color: gold;
  border: 1px solid;
  font-size: 70%;
  padding: 2px;
  margin-left: 50px;
}
```

Probleme:

1. Daten verteilt über HTML und CSS Dateien
2. Nicht die Rolle von CSS, Daten zu speichern
3. Inhalt wird nicht im DOM repräsentiert (Zugriffsproblem!)

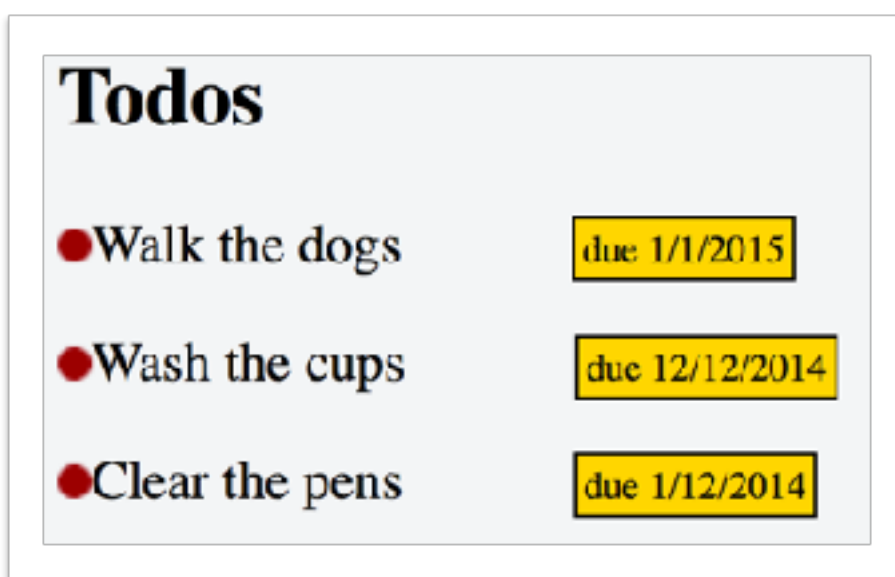
Versuch: jsfiddle öffnen und Datumsteile im Browser mit Entwicklertools untersuchen: <https://jsfiddle.net/zich0001/jL597c16/>

CSS & Data-* (besserer Weg)

CSS nutzt Daten, die in HTML Elementen gespeichert sind

Erinnerung: HTML Elemente können data-* Attribute definieren

```
<main>
  <h2>Todos</h2>
  <p id="t1" data-due="1/1/2015"> Walk the dogs</p>
  <p id="t2" data-due="12/12/2014"> Wash the cups</p>
  <p id="t3" data-due="1/12/2014"> Clear the pens</p>
</main>
```



```
p::after {
  background-color: gold;
  border: 1px solid;
  font-size: 70%;
  padding: 2px;
  margin-left: 50px;
  content: "due " attr(data-due);
}
p::before {
  content: url(https://upload.wikimedia.org/
wikipedia/commons/thumb/e/e9/Locator_Dot.svg/
10px-Locator_Dot.svg.png);
}
```


attr() liest den Inhalt
eines Attributes

content: auch URL möglich

CSS & Data-* (besserer Weg)

Noch ein Beispiel: einfacher Tooltip

```
<ul>
  <li data-name="Cascading Style Sheets">CSS</li>
  <li data-name="HyperText Markup Language">HTML</li>
  <li data-name="Hypertext Transfer Protocol">http</li>
  <li data-name="Hypertext Transfer Protocol Secure">https</li>
</ul>
```

- CSS
 - HTML
 - http
 - https
- 

```
li {
  cursor: help;
}

li:hover::after {
  background-color: rgba(10, 10, 10, 0.7);
  color: gold;
  border: 1px dashed;
  padding: 5px;
  font-size: 70%;
  content: attr(data-name);
  position: relative;
  bottom: 15px;
  left: 5px;
}
```

Anderer Cursor Typ

CSS Counters

CSS Counter können die Anzahl von Aufrufen eines Regelsatzes zählen. Counter werden von CSS initialisiert und verwaltet.

```
<main>
  <h2>Todos</h2>
  <p id="t1" data-due="1/1/2015"> Walk the dogs</p>
  <p id="t2" data-due="12/12/2014"> Wash the cups</p>
  <p id="t3" data-due="1/12/2014"> Clear the pens</p>
</main>
```

Todos

Todo 1: Walk the dogs

Todo 2: Wash the cups

Todo 3: Clear the pens

```
body {
  /* initialisiere counter auf 0 */
  counter-reset: countTodo;
}

p::before {
  /* erhöhe bei jedem <p> */
  counter-increment: countTodo;
  /* counter wird ausgegeben */
  content: " Todo " counter(countTodo) ": ";
}
```


Verschachtelte CSS Counters

Kindenelemente bekommen ihre eigene Counter Instanz
Verschiedene Instanzen werden mit **counters()** kombiniert

```
<ul>
  <li>Today's todos
    <ul>
      <li>Walk the dogs</li>
      <li>Wash the cups</li>
      <li>Clear the pens</li>
    </ul>
  </li>
  <li>Tomorrow's todos
    <ul>
      <li>Walk the dogs</li>
      <li>Wash the dishes</li>
    </ul>
  </li>
</ul>
```

```
ul {
  counter-reset: cli;
  list-style-type: none;
}

li::before {
  counter-increment: cli;
  content: counters(cli, ".") ": ";
}
```

```
1: Today's todos
  1.1: Walk the dogs
  1.2: Wash the cups
  1.3: Clear the pens
2: Tomorrow's todos
  2.1: Walk the dogs
  2.2: Wash the dishes
```

Die richtigen **CSS Features** Auswählen

Kann ich `attr()` benutzen?

Ist `attr()` ein etablierter (akzeptierter) Teil der CSS Spezifikation?

1. W3C CSS Spezifikation

- Candidate Recommendation oder Recommendation?
- CSS2 or CSS3?
- Erschöpfender Überblick aller Aspekte

2. Mozilla Developer Network

- Fokussiert die wichtigsten Aspekte einer Technik (nicht erschöpfend)
- Aktuelle Information
- Einfach einen Überblick zu bekommen

Kann ich attr() benutzen?

Specifications

Specification	Status	Comment
CSS Values and Units Module Level 3 The definition of 'attr()' in that specification.	Candidate Recommendation	Added two optional parameters; can be used on all properties; may return other values than <code><string></code> . These changes are experimental and may be dropped during the CR phase if browser support is too small.
CSS Level 2 (Revision 1) The definition of 'attr()' in that specification.	Recommendation	Limited to the <code>content</code> property; always return a <code><string></code> .

Prima, nicht nur für content!

Für content ok!

Achtung, experimentell!

Nur Strings!

Browser compatibility

Mobile Browser unterstützen attr() für content.

Feature	Android	Firefox Mobile (Gecko)	IE Mobile	Opera Mobile	Safari Mobile
Basic support	2.1	1.0 (1.0)	8	10.0	3.1
Usage in other properties than <code>content</code> and with non-string values	?	No support [1]	No support	?	?

Browser-spezifische Prefixe

CSS ist in *aktiver Entwicklung*, viele Funktionen sind nicht stabil,

- werden oft mit Browser-spezifischen Prefixes genutzt
- ändern sich vielleicht noch (in neuen Versionen der Spezifikation)

```
main:-webkit-full-screen {}  
/* Chrome */
```

```
main:-moz-full-screen {}  
/* Firefox */
```

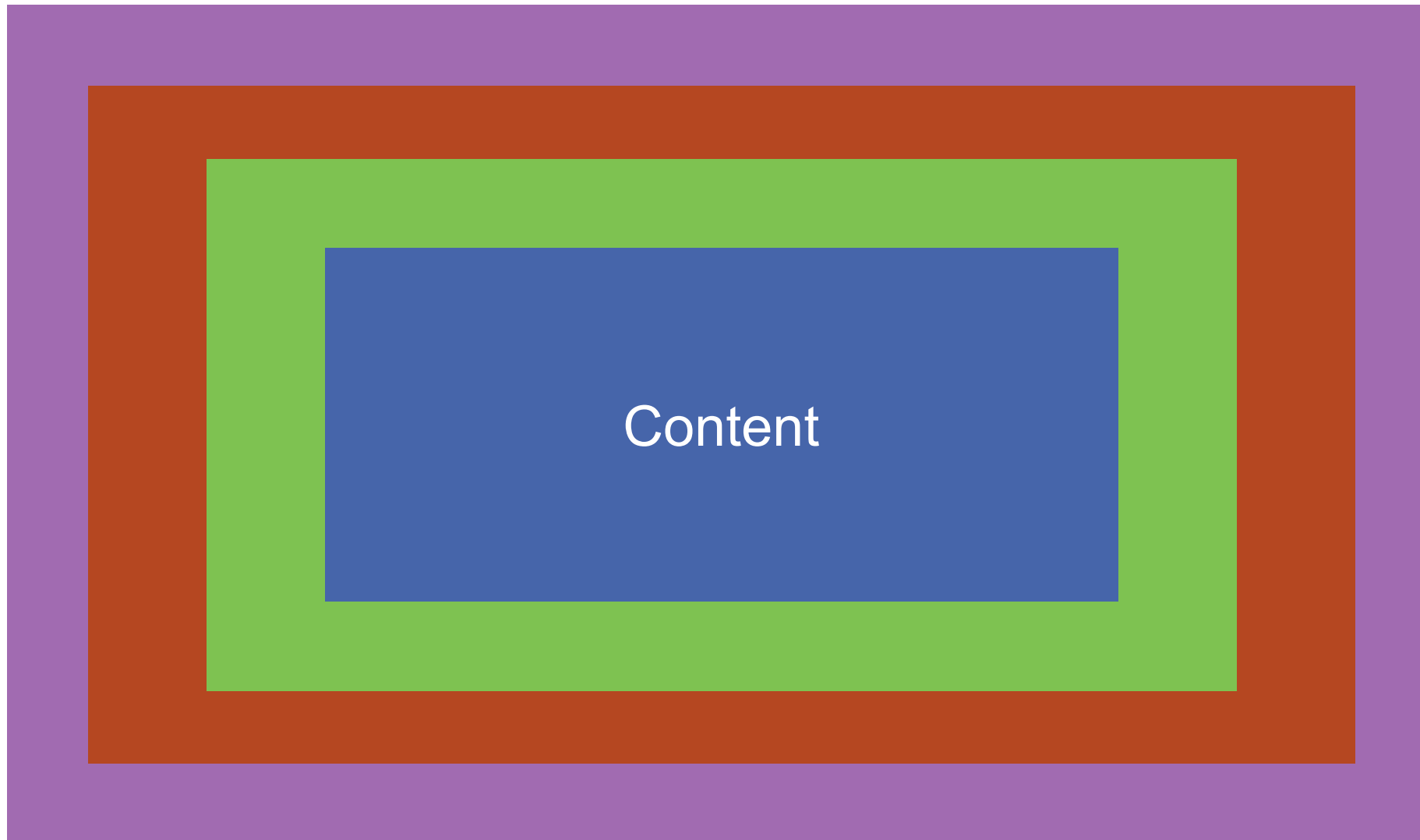
```
main:-ms-fullscreen {}  
/* Internet Explorer */
```

```
main:fullscreen {}  
/* W3C proposal */
```

- **Vorteil:** tolle neue Funktionen können schon früh genutzt werden
- **Nachteil:** neues Browser Release könnte die CSS Implementierung beeinträchtigen

Das **Box** Modell

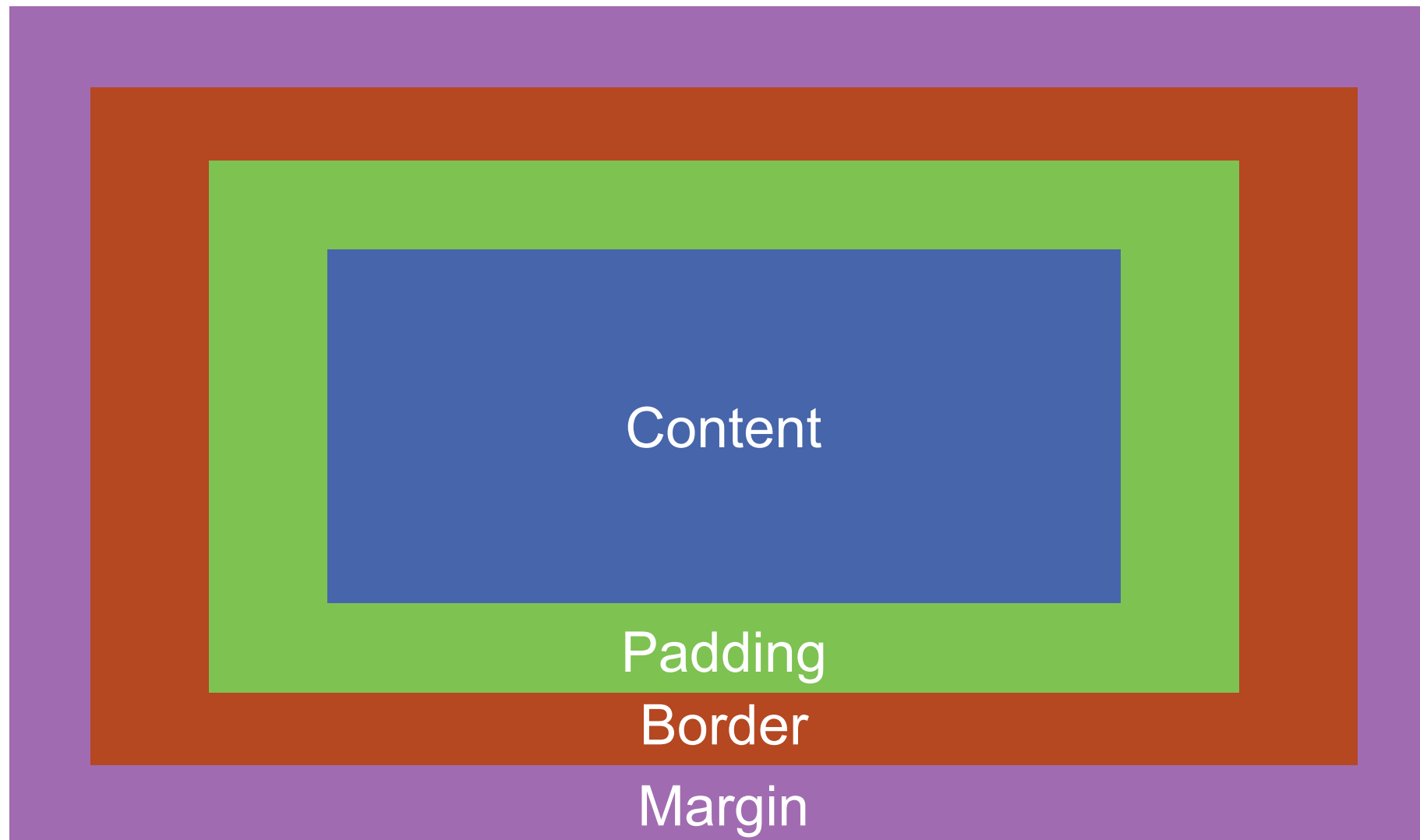
Das Box Modell



- **Padding**
- **Border**
- **Margin**

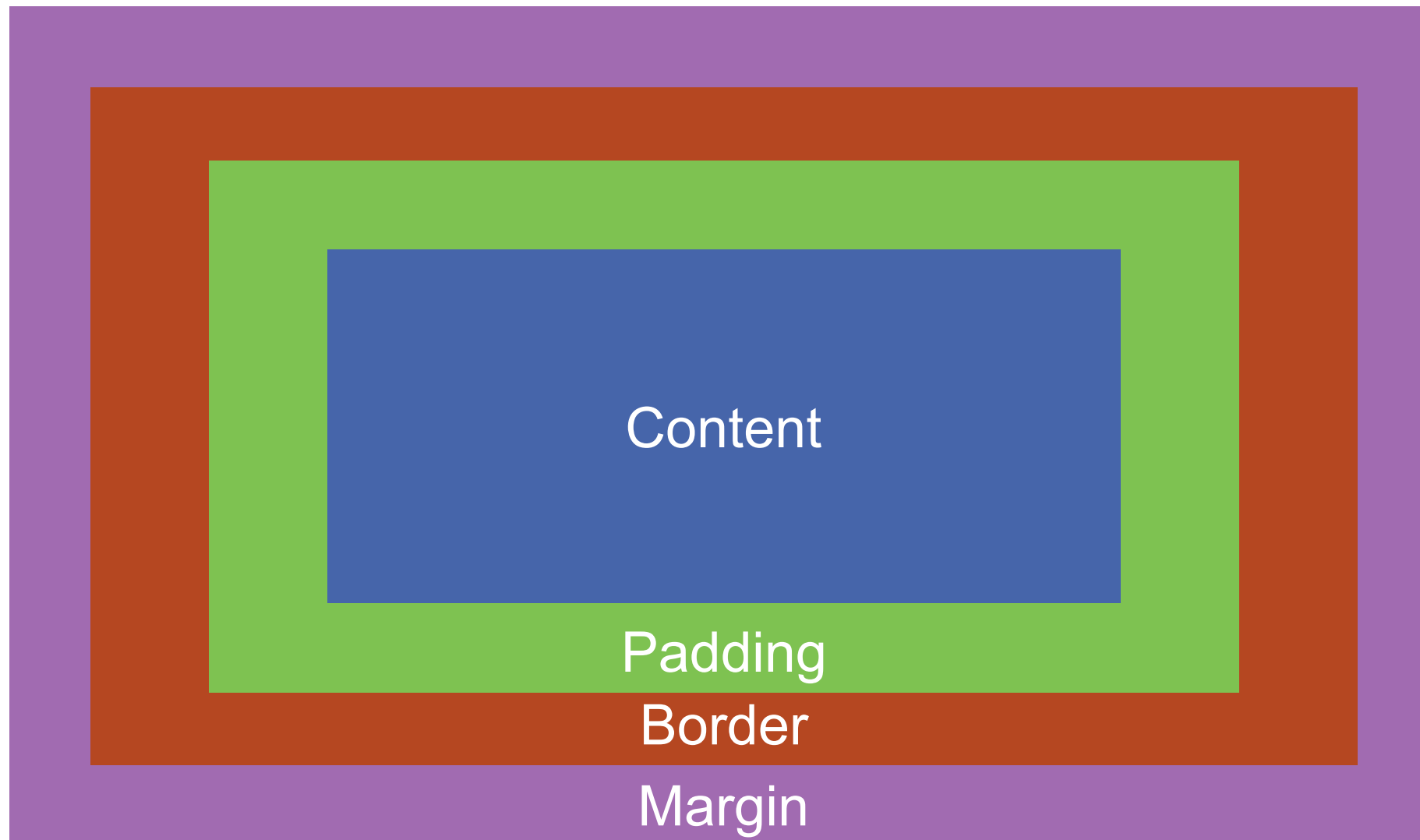
Frage: welche Fläche ist das jeweils (Farbe)?

Das Box Modell



- **Padding**: leere Fläche zwischen Content und Border
- **Border**: Fläche der Border
- **Margin**: leere Fläche zwischen dem Element und seinen Nachbarn

Das Box Modell



Experiment: man kann die CSS Stylesheets einer Seite im Browser mit den Entwicklertools ansehen und live ändern!
Laden Sie eine Seite ihrer Wahl und experimentieren Sie mit den Werten der (z.B. Box) Eigenschaften.

Elemente **positionieren** mit `float`, `position` und `display`

Elemente “fließen” im Normalfall

Blockelemente sind von Zeilenumbrüchen umschlossen. Sie können Block- oder Inline-Elemente enthalten. Die Breite wird von dem beinhalteten Element bestimmt.

z.B. `<main>` oder `<p>`

HTML 4 Terminologie

Inline-Elemente können in Block- oder Inline-Elemente platziert werden. Sie können wiederum Inline-Elemente enthalten. Die Breite wird vom Inhalt bestimmt.

z.B. `` oder `<a>`

```
<main>
  <p>
    This is a paragraph containing <a href="#">a link</a>
  </p>
  <p>
    This is another paragraph
    <span> with a span and <a href="#">a link in the span</a></span>
  </p>
</main>
```

Elemente “fließen” im Normalfall

Blockelemente sind von *Zeilenumbrüchen* umschlossen. Sie können Block- oder Inline-Elemente enthalten. Die Breite wird von dem beinhalteten Element bestimmt.

z.B. `<main>` oder `<p>`

HTML 4 Terminologie

Inline-Elemente können in Block- oder Inline-Elemente platziert werden. Sie können wiederum Inline-Elemente enthalten. Die Breite wird vom Inhalt bestimmt.

z.B. `` oder `<a>`

This is a paragraph containing [a link](#)

This is another paragraph with a span and [a link in the span](#)

```
main {width: auto;}
```

Elemente “fließen” im Normalfall

Blockelemente sind von Zeilenumbrüchen umschlossen. Sie können Block- oder Inline-Elemente enthalten. Die Breite wird von dem beinhalteten Element bestimmt.

z.B. `<main>` oder `<p>`

HTML 4 Terminologie

Inline-Elemente können in Block- oder Inline-Elemente platziert werden. Sie können wiederum Inline-Elemente enthalten. Die Breite wird vom Inhalt bestimmt.

z.B. `` oder `<a>`

This is a paragraph containing [a link](#)

This is another paragraph with a span and [a link in the span](#)

```
main {width: 275;}
```

Elemente aus dem Fluß nehmen

`float:left` (oder `:right`) nimmt ein Element aus dem Fluß; es wird zur am weitesten links (oder rechts) liegenden Position des umschließenden Elements bewegt - entweder dessen Kante oder ein anderes Float Element.

This is a paragraph containing [a link](#)

This is another paragraph with a span and [a link in the span](#)

```
a {float: none;}
```

Elemente aus dem Fluß nehmen

`float:left` (oder `:right`) nimmt ein Element aus dem Fluß; es wird zur am weitesten links (oder rechts) liegenden Position des umschließenden Elements bewegt - entweder dessen Kante oder ein anderes Float Element.

This is a paragraph containing

[a link](#)

This is another paragraph with a span and

[a link in the span](#)

```
a {float: right;}
```

[a link](#) This is a paragraph containing

[a link in the span](#) This is another paragraph with a span and

```
a {float: left;}
```

Den Fluß mit `clear` zurücksetzen

- Typisches Beispiel: Sidebars zu einem Layout hinzufügen

- Go to page 1
- Go to page 2
- Go to page 3
- Go to page 4
- Go to page 5
- Go to page 6
- Go to page 7
- Go to page 8
- Go to page 9

nav#nav1

- January 2014
- February 2014
- March 2014
- April 2014

nav#nav2

This is the 1. paragraph
This is the 2. paragraph

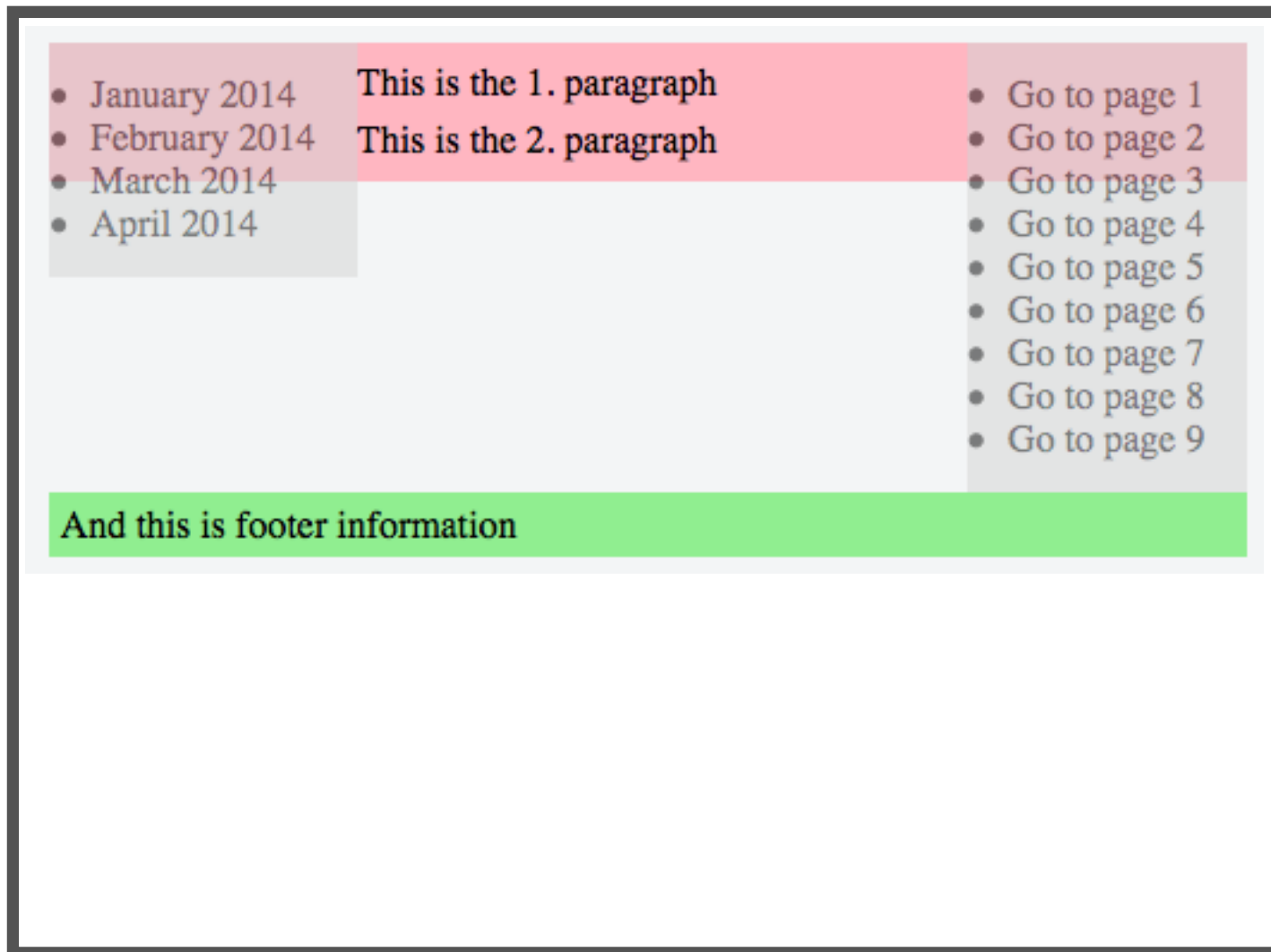
main

And this is footer information

footer

Den Fluß mit `clear` zurücksetzen

- Typisches Beispiel: Sidebars zu einem Layout hinzufügen



```
#nav1 {float: right;}
#nav2 {float: left;}
footer {clear: left;}
footer {clear: right;}
/* entspricht */
footer {clear: both;}
```

Feingranulare Bewegung von Elementen: **position**

position erlaubt es, Elemente in jede Richtung zu **verschieben** (up/down/left/right) in absoluten oder relativen Einheiten

position:static	Normalfall
position:relative	Element wird spontan justiert; andere Elemente sind nicht betroffen
position:absolute	Element wird aus dem normalen Fluß genommen (keine Platzreservierung)
position:fixed	wie absolut aber fixiert zum Viewport
position:sticky	zwischen relativ und fixed

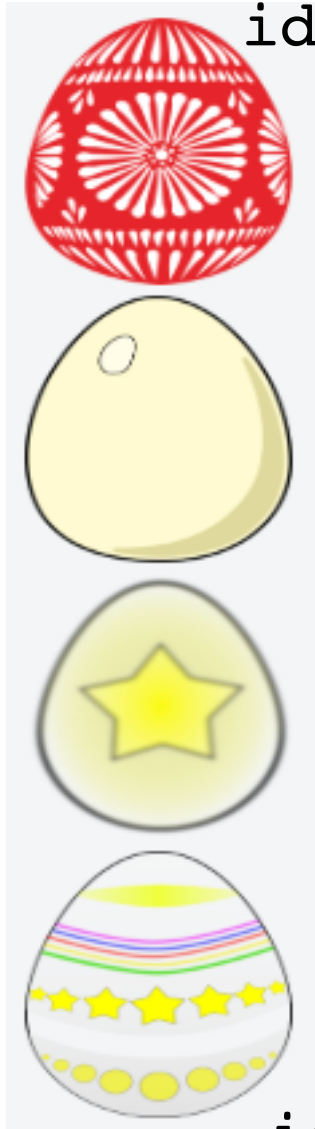
position: relative

<https://jsfiddle.net/zich0001/ftqd1ycL/>

Element wird spontan justiert; andere Elemente sind nicht betroffen

Bewegung relativ zur Originalposition

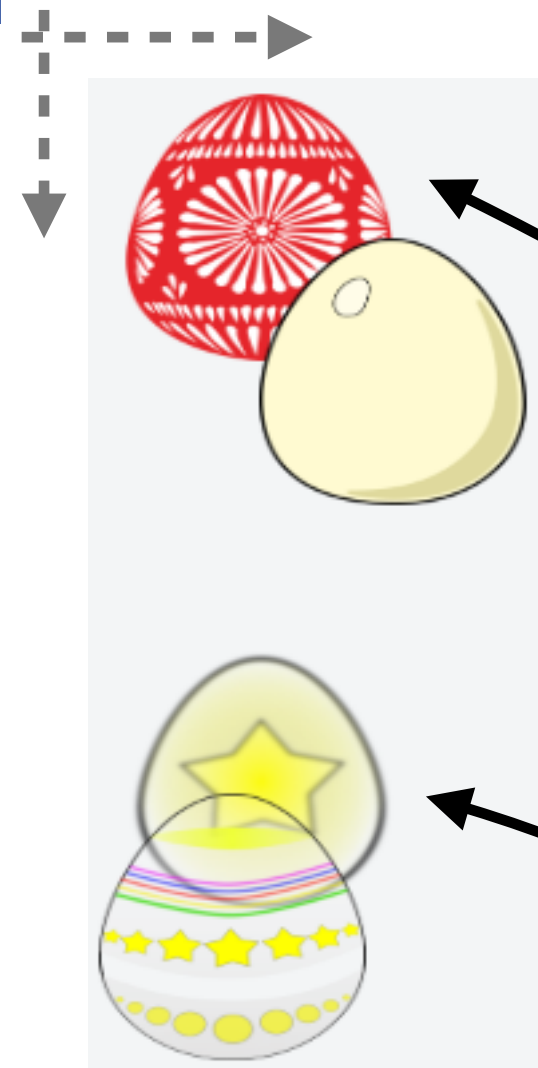
id="egg1"



```
#egg2 {  
  position: relative;  
  bottom: 50px;  
  left: 50px;  
}
```

```
#egg4 {  
  position: relative;  
  bottom: 50px;  
  right: 10px;  
}
```

id="egg4"



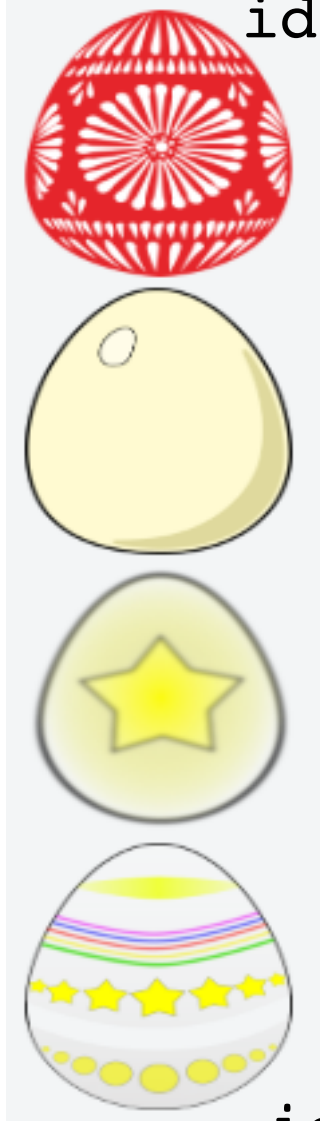
unverändert

position: absolute

Element wird aus dem Fluß genommen (keine Platzreservierung)

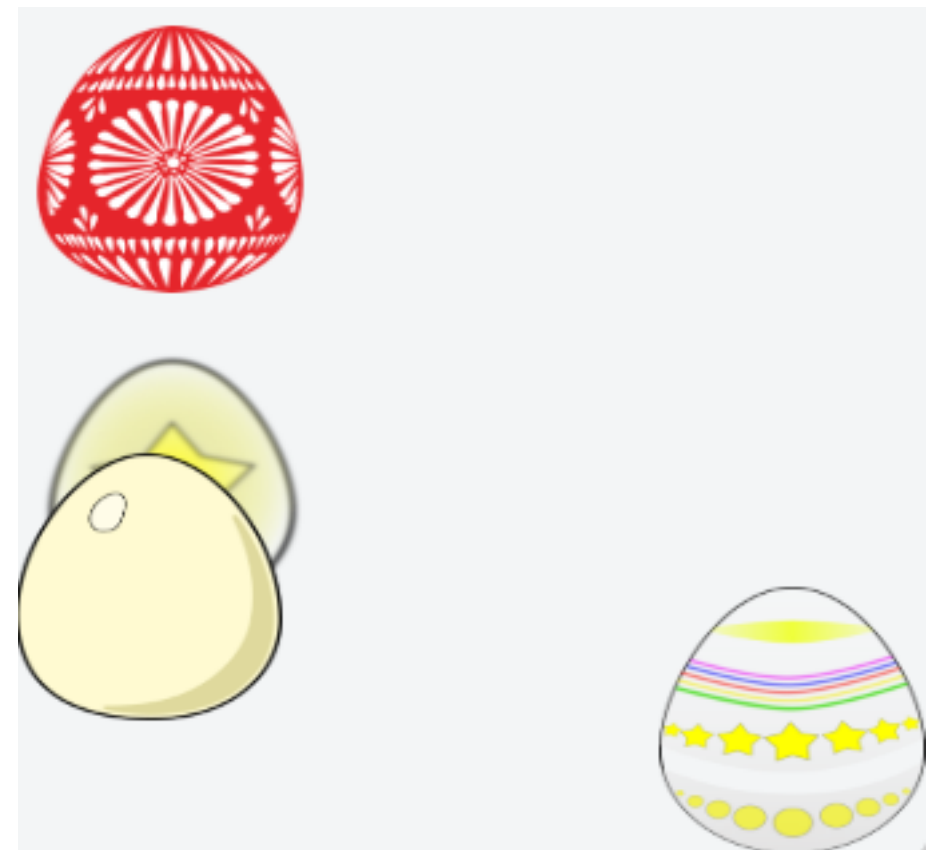
Position **relativ** zum nächsten **Vorfahren** oder **Fenster**

id="egg1"



```
#egg2 {  
  position: absolute;  
  bottom: 50px;  
  left: 0px;  
}
```

```
#egg4 {  
  position: absolute;  
  bottom: 0px;  
  right: 0px;  
}
```



id="egg4"

Literatur

- **Learning Web App Development, Kap. 3**
- Auch gut:
 - Andy Budd, Emil Björklund, "CSS Mastery", 3rd Edition, Apress, 2016 (Online im Hochschulnetz)
 - Peter Gasston, "The Book of CSS3", 2nd Edition, No Starch Press, 2014
- Die nächste Vorlesung setzt JavaScript Grundkenntnisse voraus!
- **Bitte lesen Sie vorab (bis zum 10/12.04):**
Learning Web App Development, Kapitel 4

