Verteilte Systeme 1

Technologien des World Wide Web

christian.zirpins@hs-karlsruhe.de

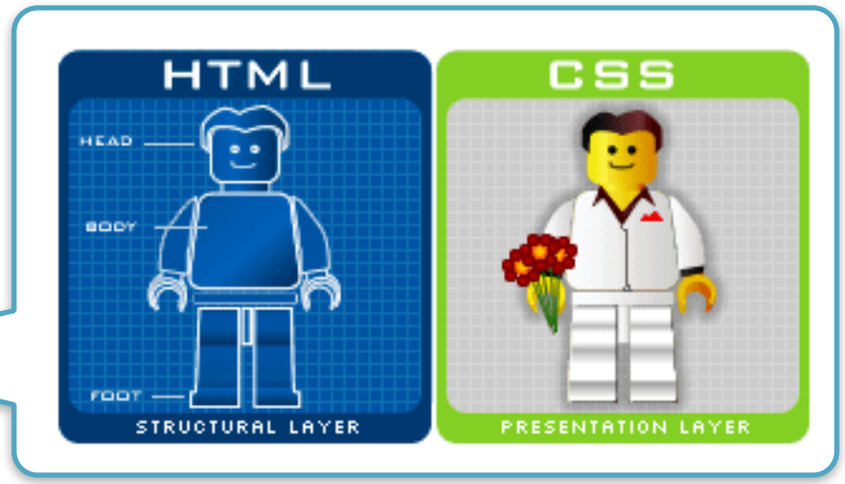
Gestaltung von Webapps mit CSS3

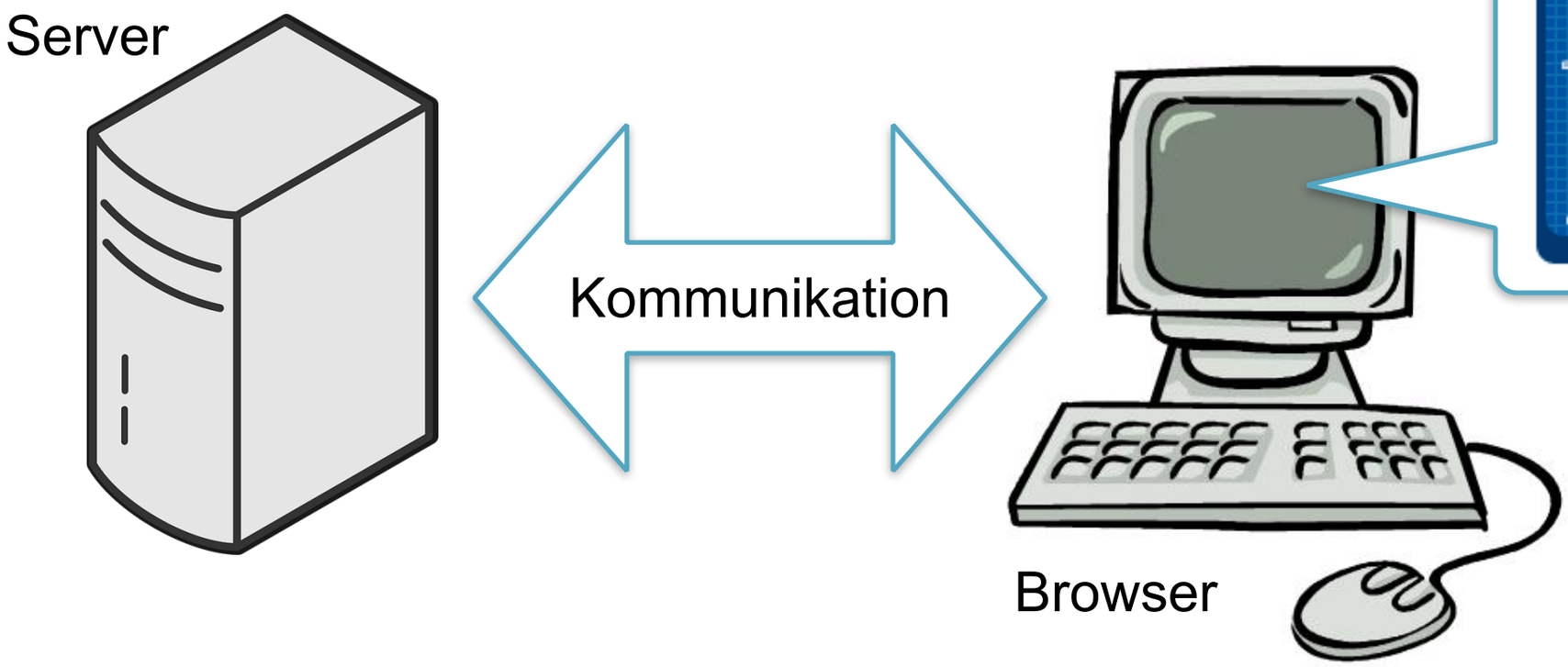
Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

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
- …CSS **Media Queries** schreiben





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…

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

Mittlere Priorität	Aktuell	Zukünftig
CSS Paged Media Level 3	WD	WD
CSSOM View	WD	WD
CSS Intrinsic & Extrinsic Sizing Level 3	WD	CR
CSS Ruby Level 1	WD	WD
CSS Overflow Level 3	WD	WD
CSS Box Model Level 3	WD	WD
CSS Pseudo-Elements Level 4	WD	WD
CSS Scrollbars Level 1	FPWD	WD
CSS Backgrounds and Borders Level 4		FPWD
CSS Device Adaptation	WD	WD
CSS Exclusions	WD	WD
Filter Effects Level 1	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 Positioned Layout Level 3	WD	WD
CSS Regions	WD	WD

…

Geringe Priorität	Aktuell	Zukünftig
CSS Generated Content Level 3	WD	WD
CSS Level 1	SPSD	
CSS Print Profile	NOTE	
CSS Mobile Profile 2.0	NOTE	
Non-element Selectors	NOTE	
The CSS 'Reader' Media Type	NOTE	
CSS Presentation Levels	NOTE	
CSS TV Profile 1.0	NOTE	

…

Abgeschlossene Werke	Aktuell	Zukünftig
CSS Snapshot 2018	NOTE	
CSS Snapshot 2017	NOTE	
CSS Snapshot 2015	NOTE	
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
Media Queries	REC	REC
CSS Style Attributes	REC	REC
CSS Fonts Level 3	REC	REC
CSS Basic User Interface Level 3	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 Level 1	WD	CR
CSS Values and Units Level 3	CR	PR
CSS Flexible Box Layout Level 1	CR	PR
CSS Cascading and Inheritance Level 3	CR	PR
CSS Writing Modes Level 3	PR	REC
CSS Counter Styles Level 3	CR	PR
CSS Containment Level 1	PR	REC
CSS Images Level 3	CR	CR
CSS Speech	CR	CR
CSS Text Decoration Level 3	CR	CR
CSS Shapes Level 1	CR	CR

…

First Public Working Draft (FPWD)
Working Draft (Arbeitsentwurf) (WD)
Candidate Recommendation (Veröffentlichungsvorschlag) (CR)
Proposed Recommendation (Geplante Veröffentlichung) (PR)
Recommendation (Standard) (REC)
Superseded Recommendation (SPSD)

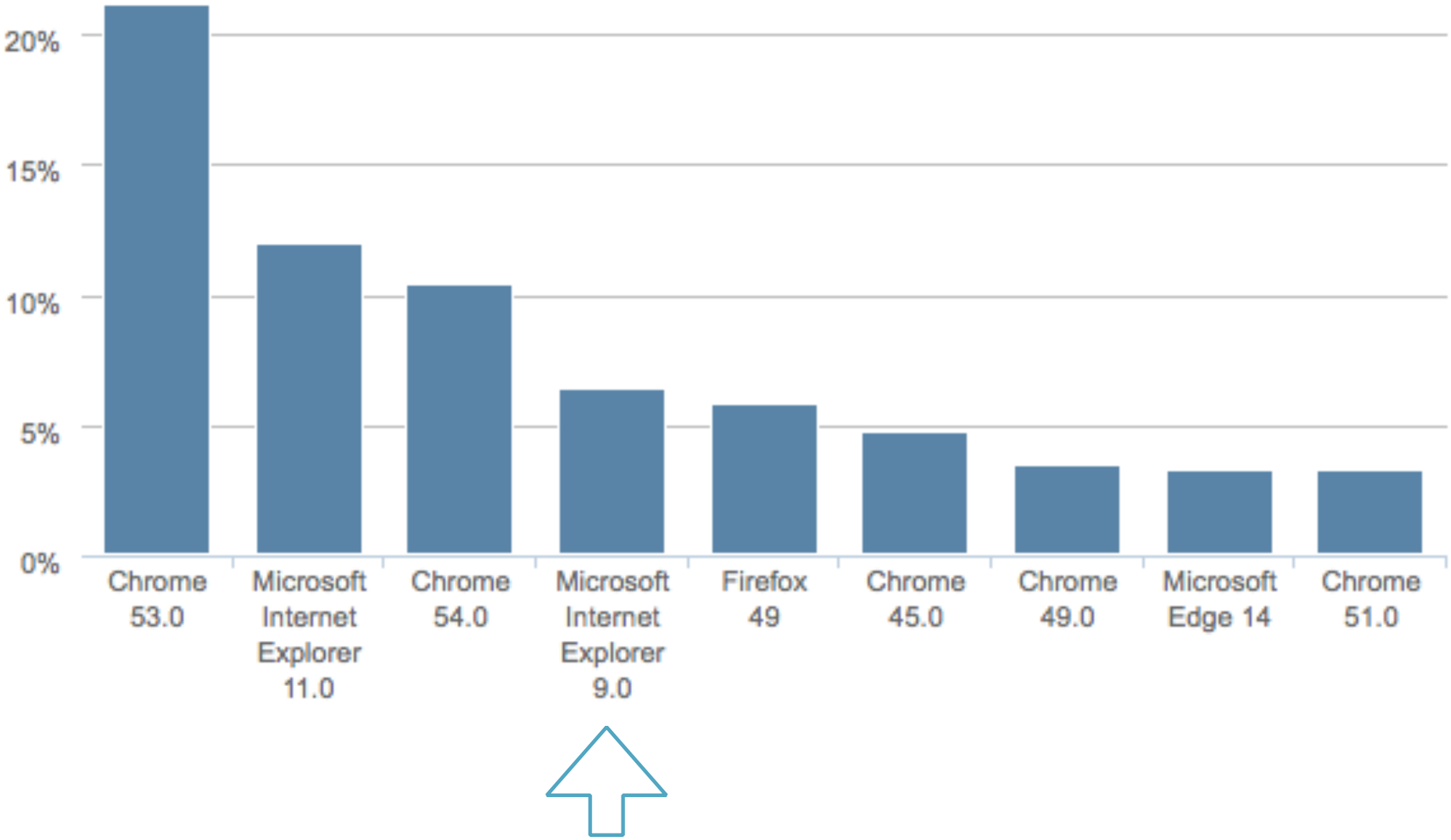
November 2019, https://www.w3.org/Style/CSS/current-work

CSS 3

- Unmöglich CSS zu schreiben, das moderne Funktionen nutzt und auf allen Browsern funktioniert…
- Verwendung von CSS 3 Features abhängig von…
  - **Nutzergruppe** (hauptsächlich in den USA oder China oder ...?),
  - **Nutzungsmodus** (Smartphone, Touchscreen oder Desktop?),
  - **Anwendungstyp** (keine 3D-Animationen für Aufgabenliste)
- 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.)

November 2016, http://www.netmarketshare.com

Rückblick: Kapitel 3

Rückblick: ein einfaches HTML Dokument

- Das Dokument beschreibt die Struktur einer Seite, die im Browser dargestellt werden soll.

```
<!doctype html>
<html>
<head>
  <title>My Web App</title>
</head>

<body>
  <h1>Hello World!</h1>
  <p align="center">
    Nice to meet you.
  </p>
  <ol>
    <li> This is the first page
  </ol>
  <ul>
    <li> but not the last.
  </ul>

  <p class="last">
    Press the button.
  </p>

  <button id="the-button">
    Next page
  </button>

</body>
</html>
```

Datei **page1.html**

Rückblick: Kapitel 3

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

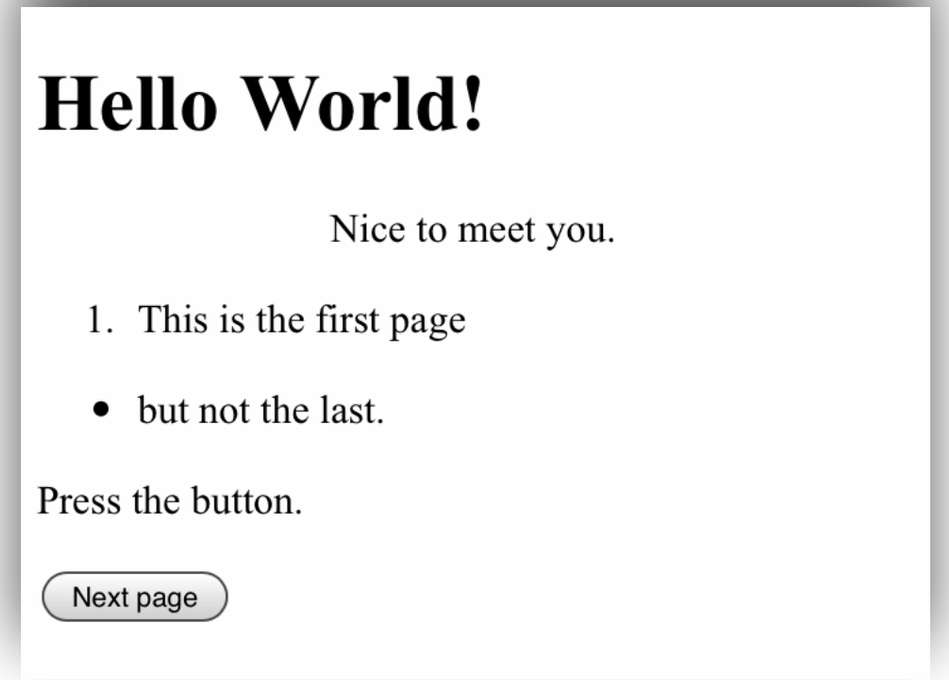
```
body {
    background-color: #ffee22;
    width: 200px;
    margin: auto;
}

#the-button {
    color: maroon;
}

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

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

Selektor

Eigenschaft

Wert

CSS Datei



Darstellung von `page1.html` noch ohne CSS

Frage: was tun die gezeigten Selektoren?

Rückblick: Kapitel 3

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

```
body {
    background-color: #ffee22;
    width: 200px;
    margin: auto;
}

#the-button {
    color: maroon;
}

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

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

Selektor

Eigenschaft

Wert

CSS Datei

Darstellung von `page1.html` nun mit CSS

Frage: was tun die gezeigten Selektoren?

Rückblick: Kapitel 3

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

```
body {
    background-color: #ffee22;
    width: 200px;
    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

überschreibt

- Stylesheets werden in Reihenfolge verarbeitet; spätere Deklarationen überschreiben frühere
- `!important` überschreibt alle anderen Deklarationen (sollte möglichst nicht verwendet werden)

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
EdgeHTML	Microsoft Edge

Gängige Pseudoklassen

`:hover`	ein Zeigegerät (Maus) schwebt über dem Element
`:active`	das Element ist derzeit aktiv (z. B. angeklickt)

CSS Code

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

```css
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

randlos = CSS

```
button {
    ...
}

button:enabled:hover {
    ...
}

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

gelber Rand = 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-te
`:nth-of-type(x)`	alle n-te

s Typs

```html
<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>
```

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

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

Gängige Pseudoklassen

`:nth-child(x)`	alle n-ten Kind Elemen
`:nth-of-type(x)`	alle n-ten Geschwistere s Typs


```html
<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>
```

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

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

Frage: was ist das Ergebnis?

`:nth-x` mit Offset und Wiederholung

`:nth-child(an+b)`	alle a-ten Kind Elemente ihrer Eltern ab b
`:nth-of-type(an+b)`	alle a-ten Geschwister eines Typs ab b

```
<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(2n) {
    color: red;
}

p:nth-of-type(2n+1) {
    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(1)`

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;
}
```

Frage: was ist das Ergebnis?

Gängige Pseudoklassen

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

selektiert Elemente mit eingeschränktem Wertebereich

```
<main>
    <input type="text" id="td" placeholder="add your todo" />
    <label for="td"> </label>
    <input id="dl" name="dl" type="number" min="0"
        max="30" placeholder="Days to deadline" required />
    <label for="dl"> </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);
}

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

Unicode

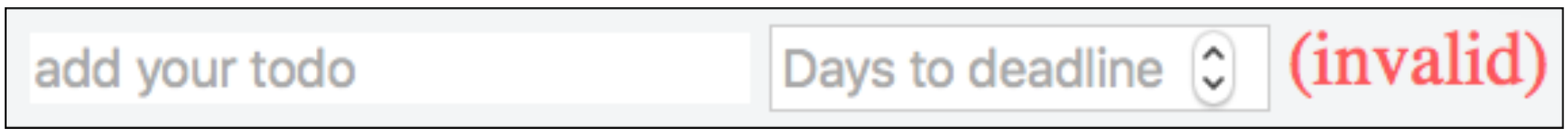
Pseudoelement

RGB & Alpha

Gängige Pseudoklassen

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

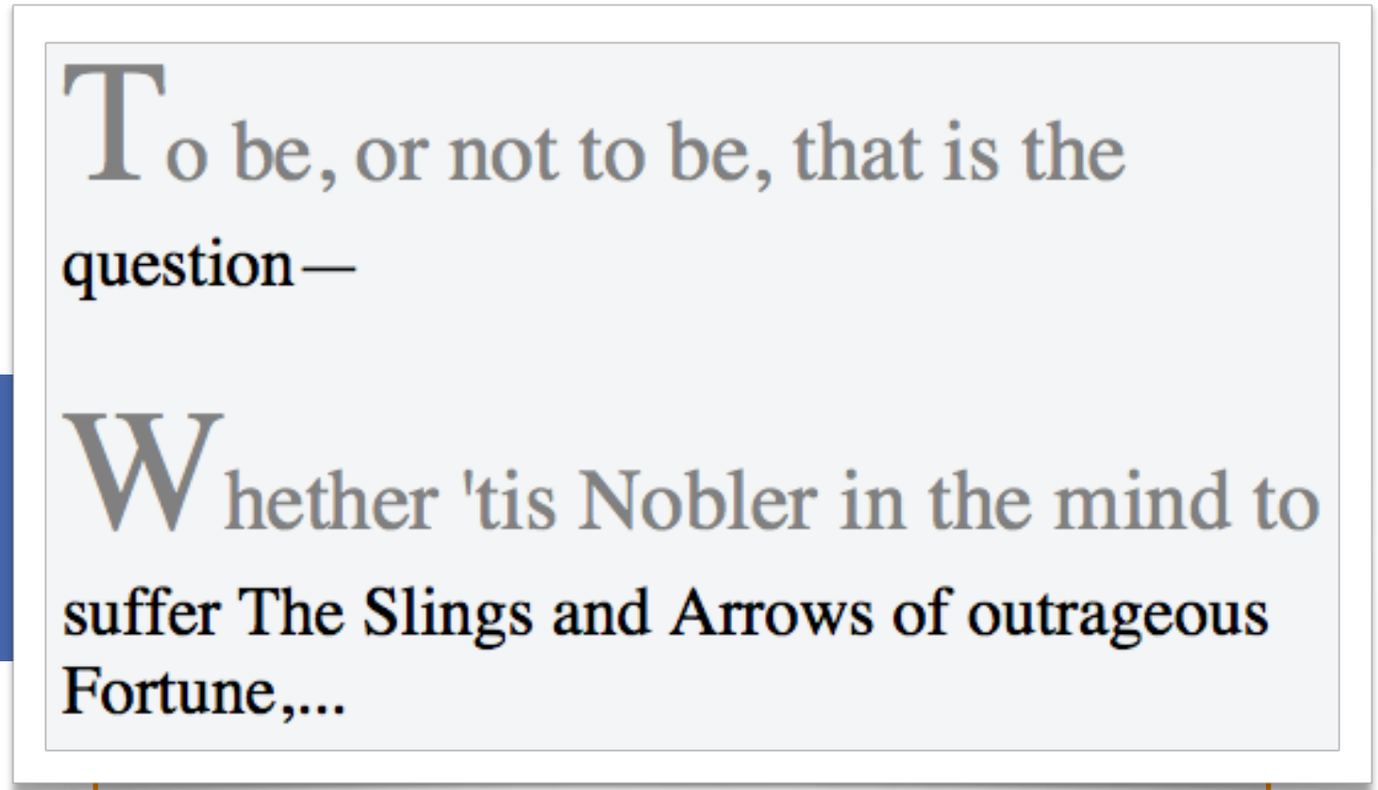
selektiert Elemente mit eingeschränktem Wertebereich



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

`::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 enthält Daten!?

```
cite::before {
    content: "\201CTo be, or ";
}

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

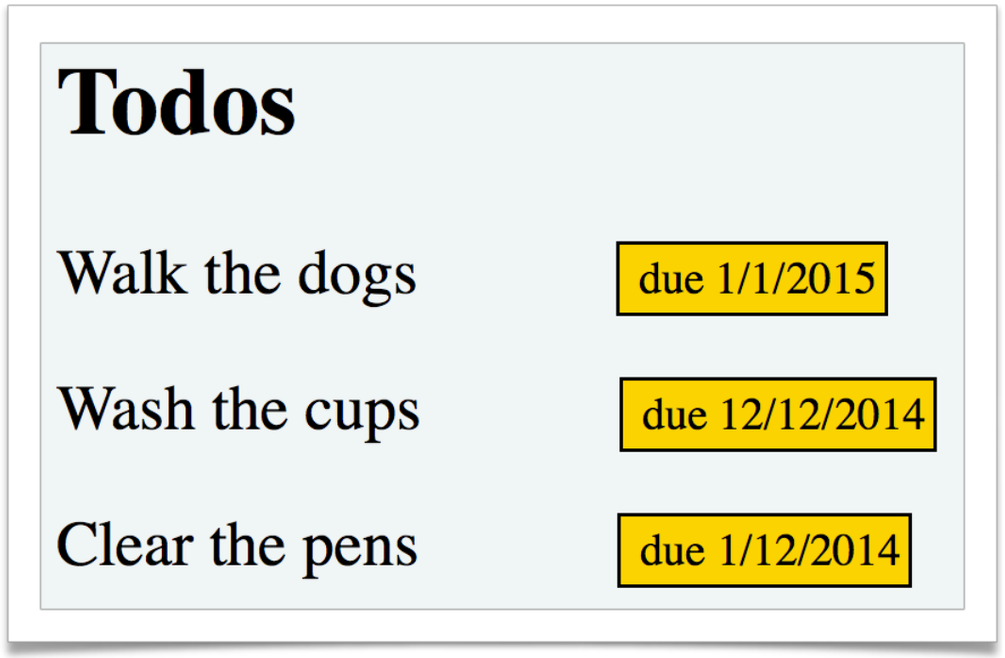
> *“ To be, or not to be ... ”*

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!)

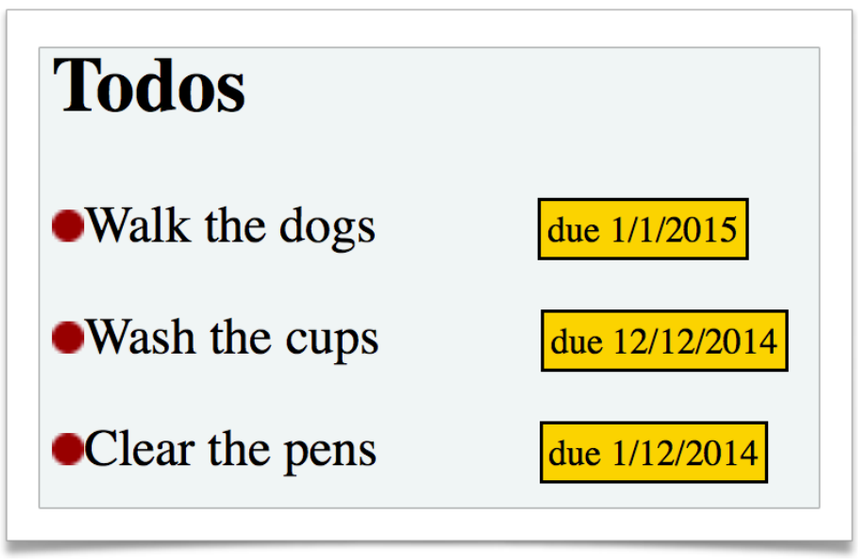
Versuch: jsfiddle öffnen und Datumsteile im Browser mit Entwicklertools untersuchen: https://jsfiddle.net/zich0001/jL597c16/

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

HyperText Markup Language

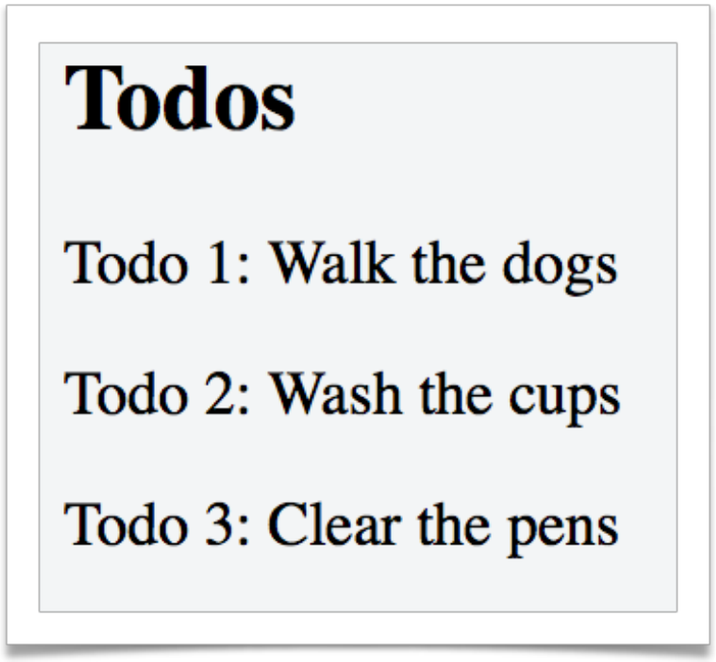
```
li {
    cursor: help;
}

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>
```




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

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

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
- Gut, um 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 `<string>`. 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 `content` property; always return a `<string>`.

Prima, nicht nur für `content`!

Achtung, experimentell!

Für `content` ok!

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 `content`and with non-string values	?	No support [1]	No support	?	?

Browser-spezifische Prefixe

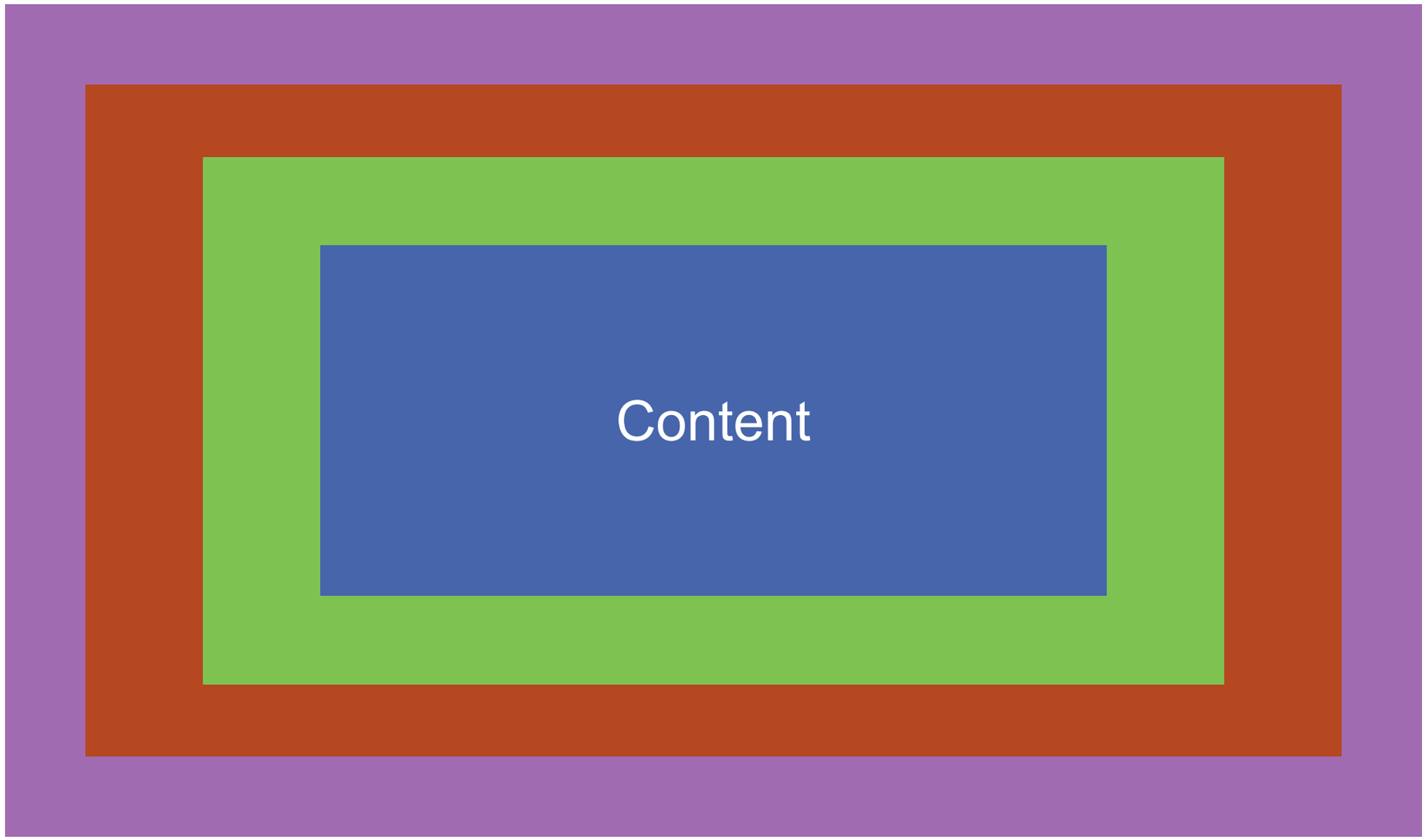
CSS ist in *aktiver Entwicklung*, viele Funktionen sind nicht stabil,
- werden oft mit Browser-spezifischen Prefixes genutzt
- ändern sich vielleicht noch (in neuen Versionen der Spezifikation)

```
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

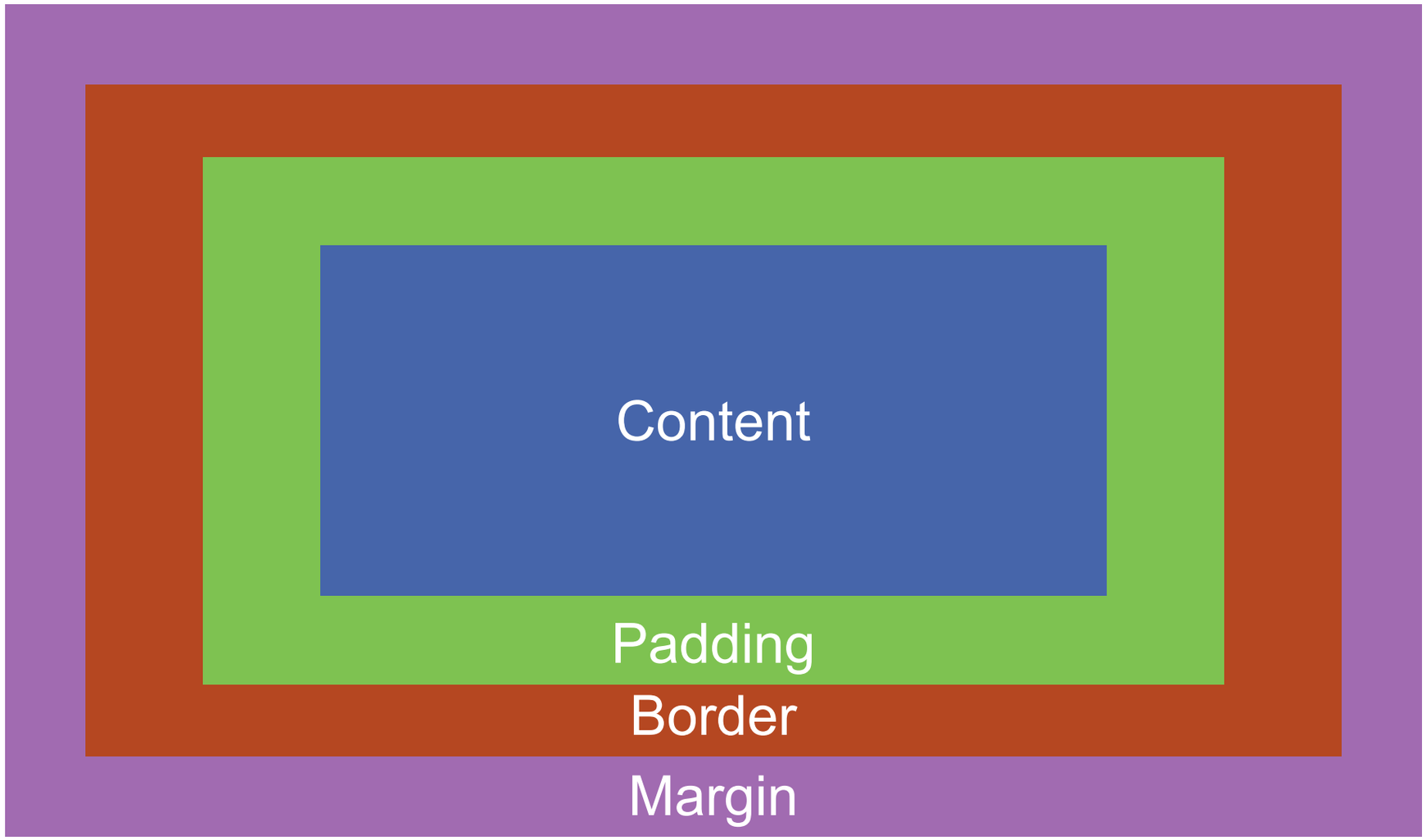
Das Box Modell



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

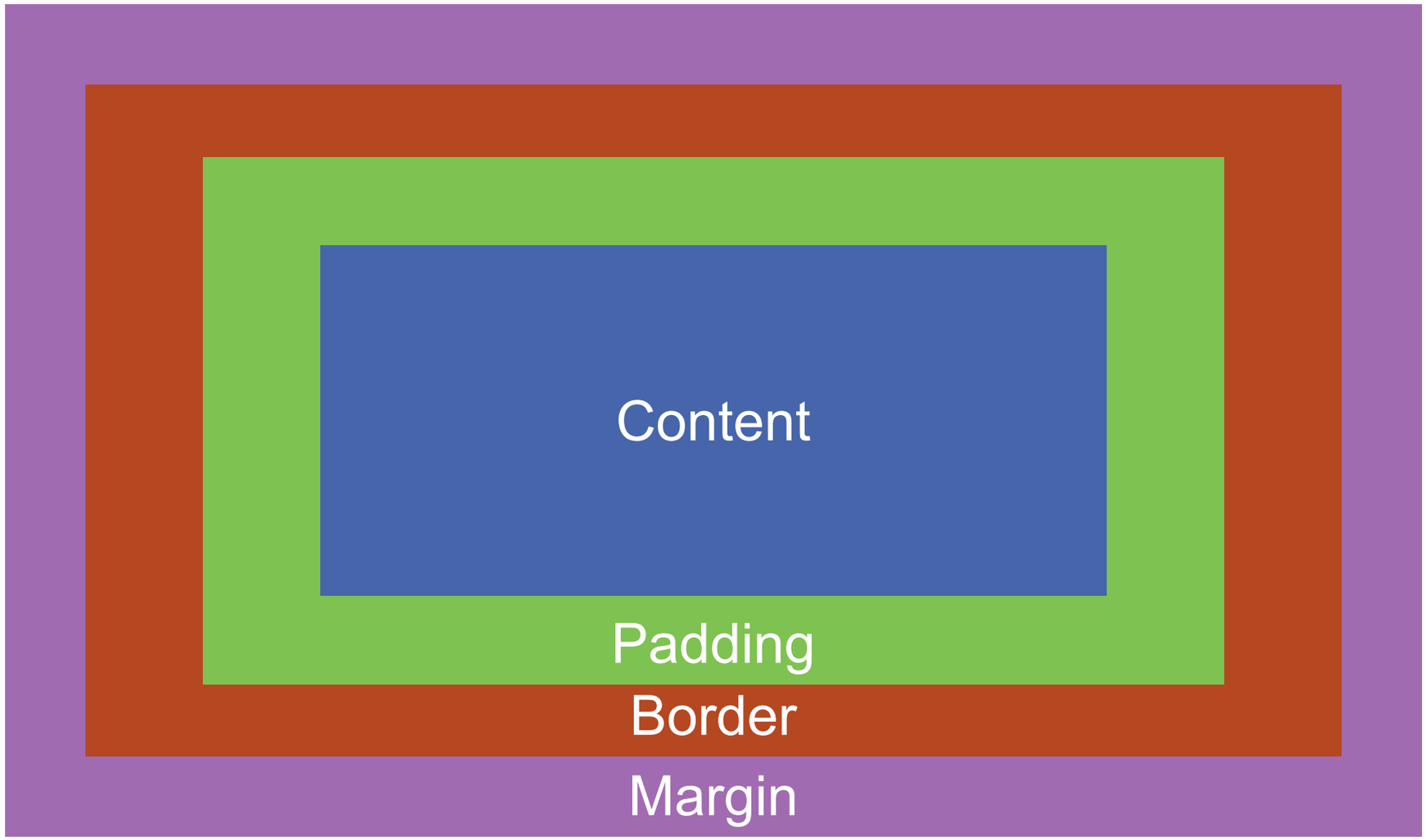
Frage: welche Fläche ist das jeweils (Farbe)?

Das Box Modell



- **Padding**: leere Fläche zwischen Content und Border
- **Border**: Fläche der Border (Grenze bzw. Rahmen)
- **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 Eigenschaften (z.B. Box).

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

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. `<span>` oder `<a>`

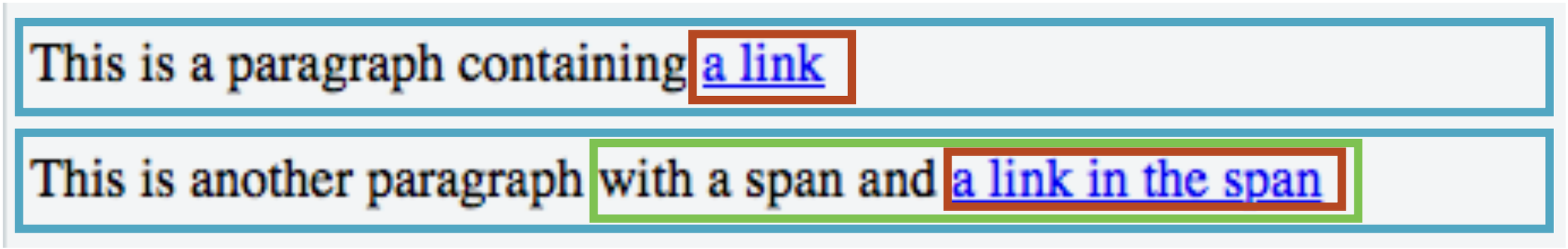
```
<main>
    <p>
        This is a paragraph containing <a href="#">a link</a>
    </p>
    <p>
        This is another paragraph
        <span> with a span and <a href="#">a link in the span</a></span>
    </p>
</main>
```

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. `<span>` oder `<a>`



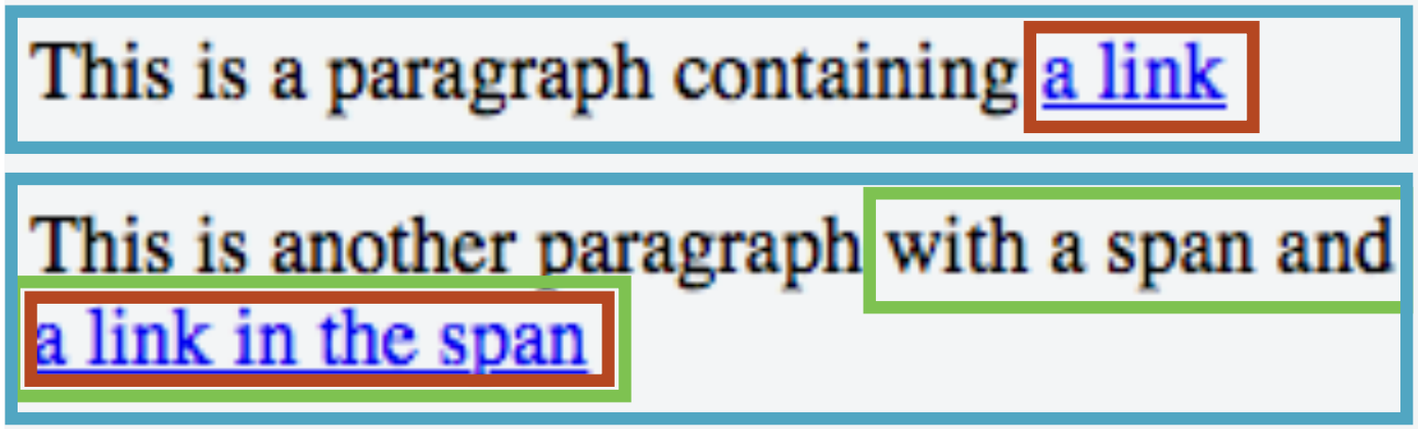

```
main {width: auto;}
```

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

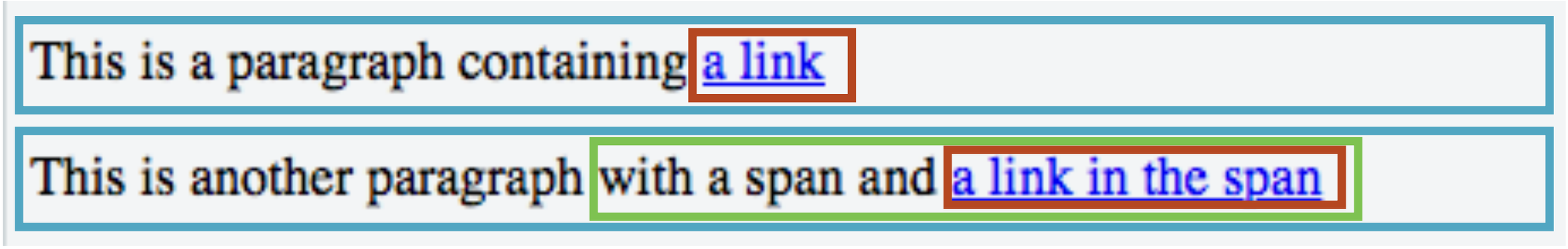
Inline-Elemente können in Block- oder Inline-Elemente platziert werden. Sie können wiederum Inline-Elemente enthalten. Die Breite wird vom Inhalt bestimmt.
z.B. `<span>` oder `<a>`




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

Elemente aus dem Fluß nehmen

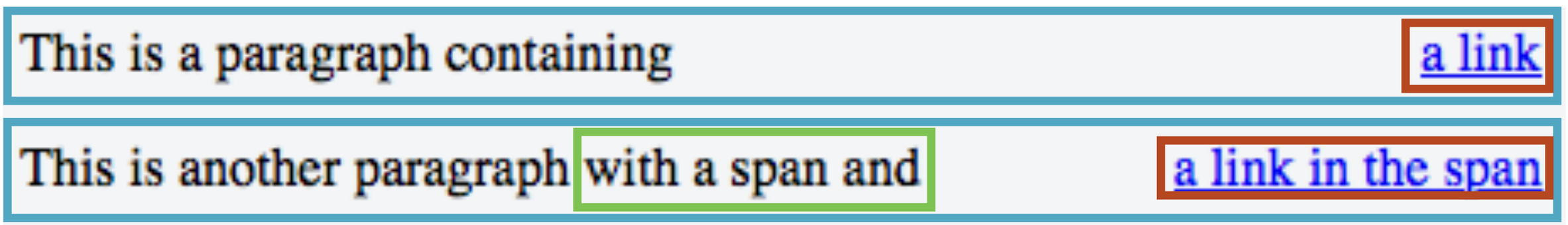
`float:left` (oder **`:right`**) nimmt ein Element aus dem Fluß; es wird zur am weitesten links (oder rechts) liegenden Position des umschließenden Elements bewegt - entweder dessen Kante oder ein anderes Float Element.



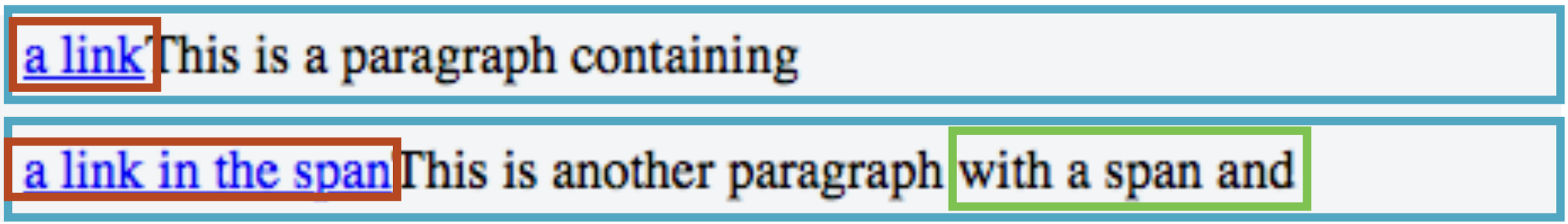

```
a {float: none;}
```

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.




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



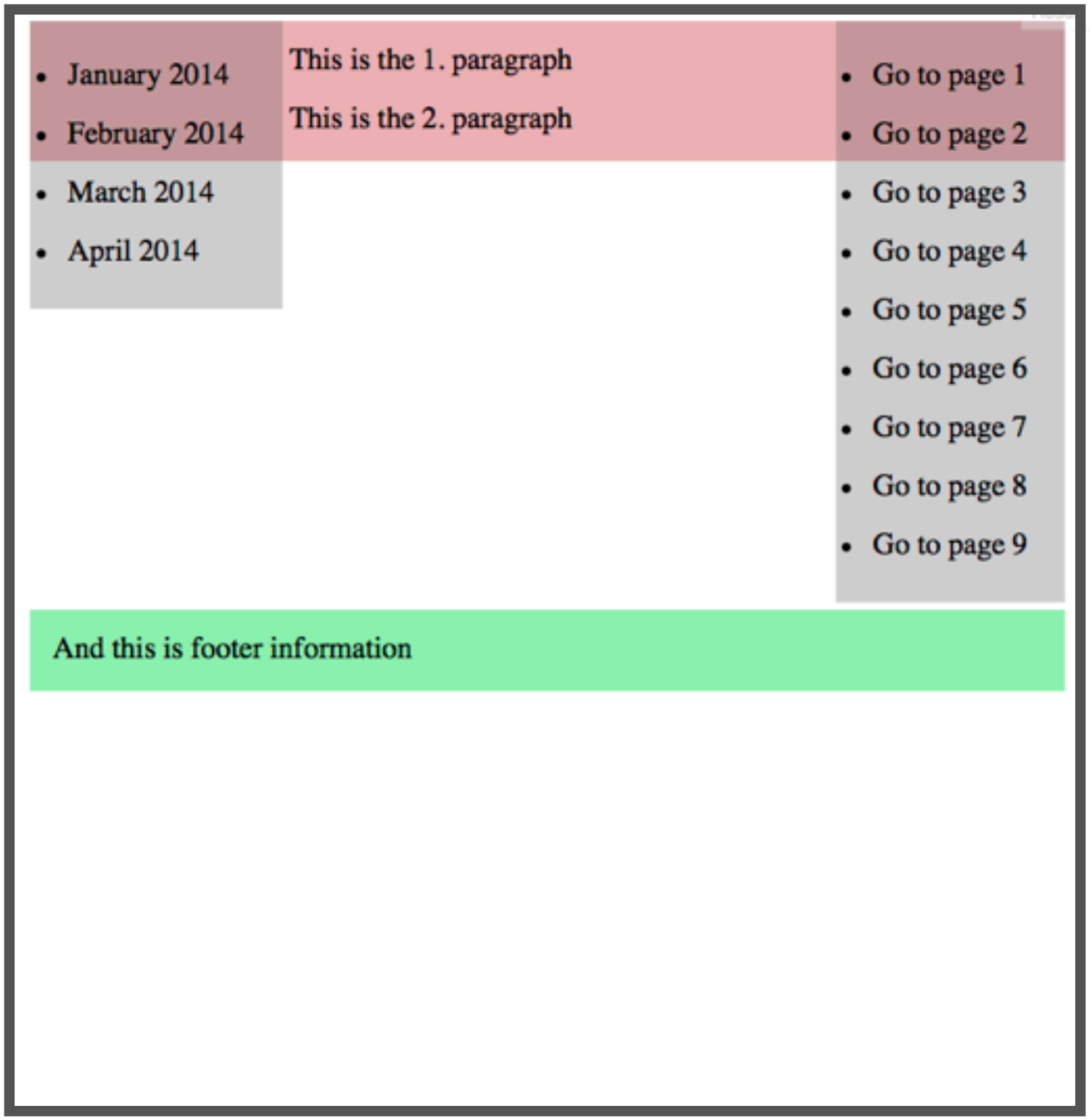

```
a {float: left;}
```

Den Fluß mit `clear` zurücksetzen

- Typisches Beispiel: Sidebars zu einem Layout hinzufügen

Den Fluß mit `clear` zurücksetzen

- Typisches Beispiel: Sidebars zu einem Layout hinzufügen




```
#nav1 {float: right;}
```

```
#nav2 {float: left;}
```

```
footer {clear: left;}
```

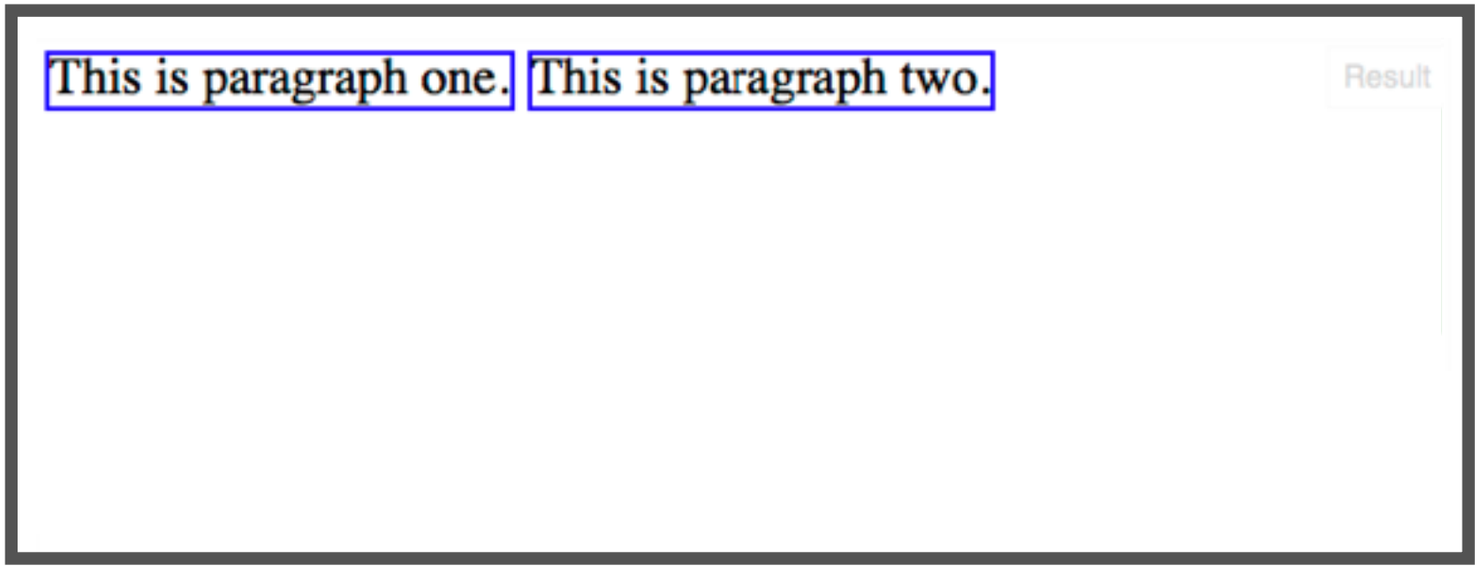
```
footer {clear: right;}
```

entspricht

```
footer {clear: both;}
```

display

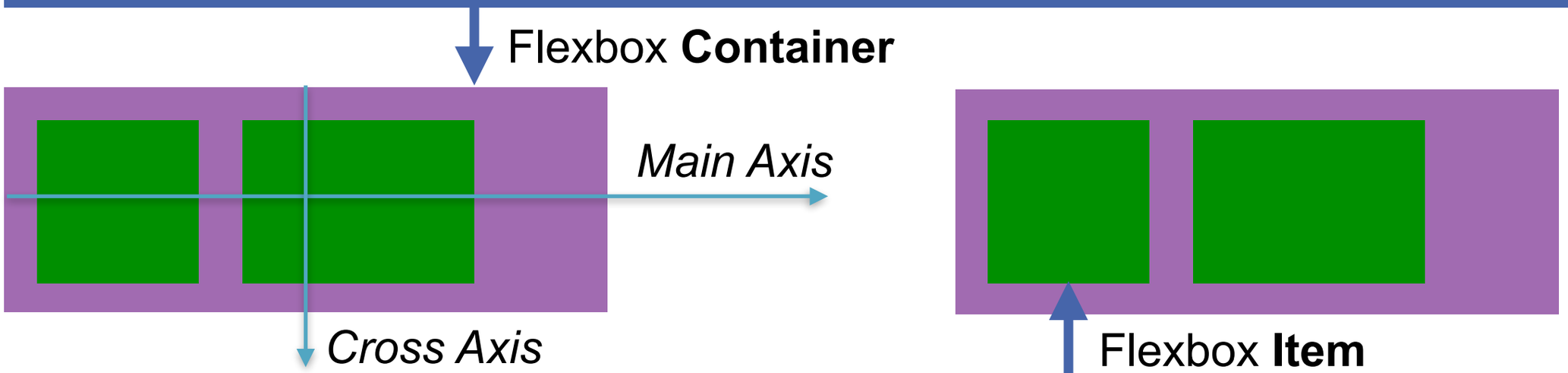
`display:inline`	Element wird mit **inline** Element-Box gerendert
`display:block`	Element wird mit **block** Element-Box gerendert
`display:none`	Element (und Nachfolger) werden **versteckt** und kein Platz wird im Layout reserviert




```
1 span {display: block;}
2 p {display: inline;}
3 span {display: none;}
```

Flexbox: Elemente flexibel & dynamisch verteilen

`display:flex`	Element ist **Flexbox** Container
`flex-direction:row`|`column`	**Richtung** der Ausbreitung
`justify-content:stretch`	Dehne Items auf Containerlänge
`justify-content:``space-between`	Gleiche Zwischenräume

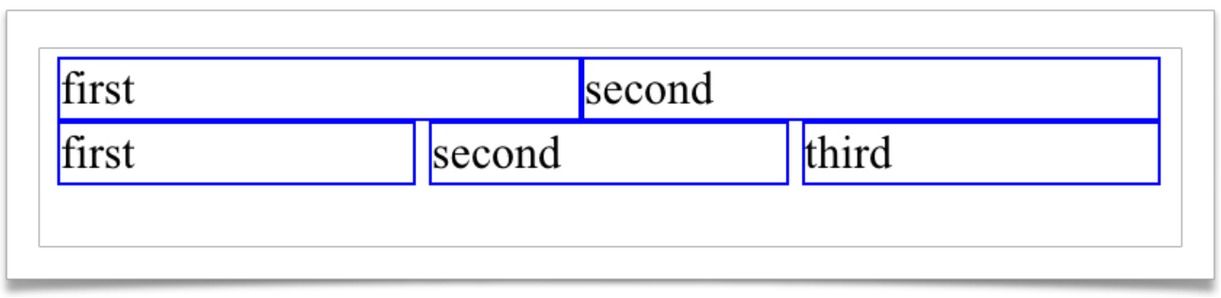


`order:<Zahl>`	Reihenfolge des Items im Container
`flex-grow:<Zahl>`	Wachstumsfaktor relativ zu anderen Items
`flex-shrink:<Zahl>`	Schrumpfung relativ zu anderen Items
`flex-basis:``<Länge>`	Raum des Items absolut festlegen
`flex-basis:auto`	Raum des Items ist Größe des Elements

Beispiel: einfaches Grid Layout mit Flexbox

Ein Grid Layout besteht aus flexiblen Reihen und Spalten

```
<div class="flex-grid">
  <div class="col">first</div>
  <div class="col">second</div>
</div>

<div class="flex-grid-thirds">
  <div class="col">first</div>
  <div class="col">second</div>
  <div class="col">third</div>
</div>
```




```
.flex-grid {
  display: flex;
}
.flex-grid .col {
  flex-grow: 1;
}

.flex-grid-thirds {
  display: flex;
  justify-content: space-between;
}
.flex-grid-thirds .col {
  width: 32%;
}

.col {
  border:  1px solid blue;
}
```

Feingranulare Bewegung von Elementen: `position`

Hochschule Karlsruhe
Technik und Wirtschaft
UNIVERSITY OF APPLIED SCIENCES

`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

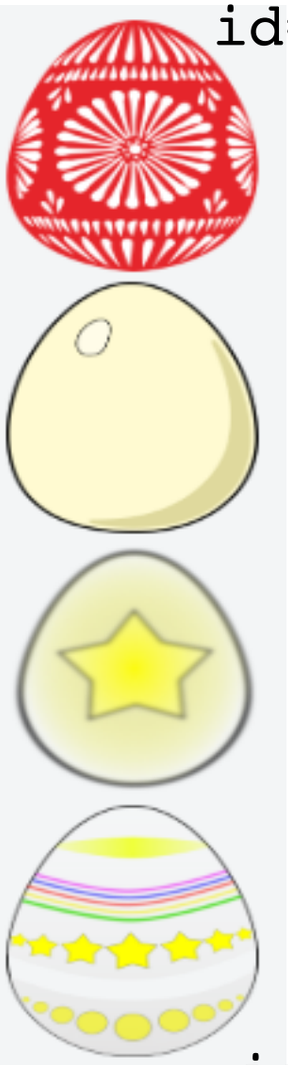
position:relative

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

Element wird spontan justiert; andere Elemente sind nicht betroffen

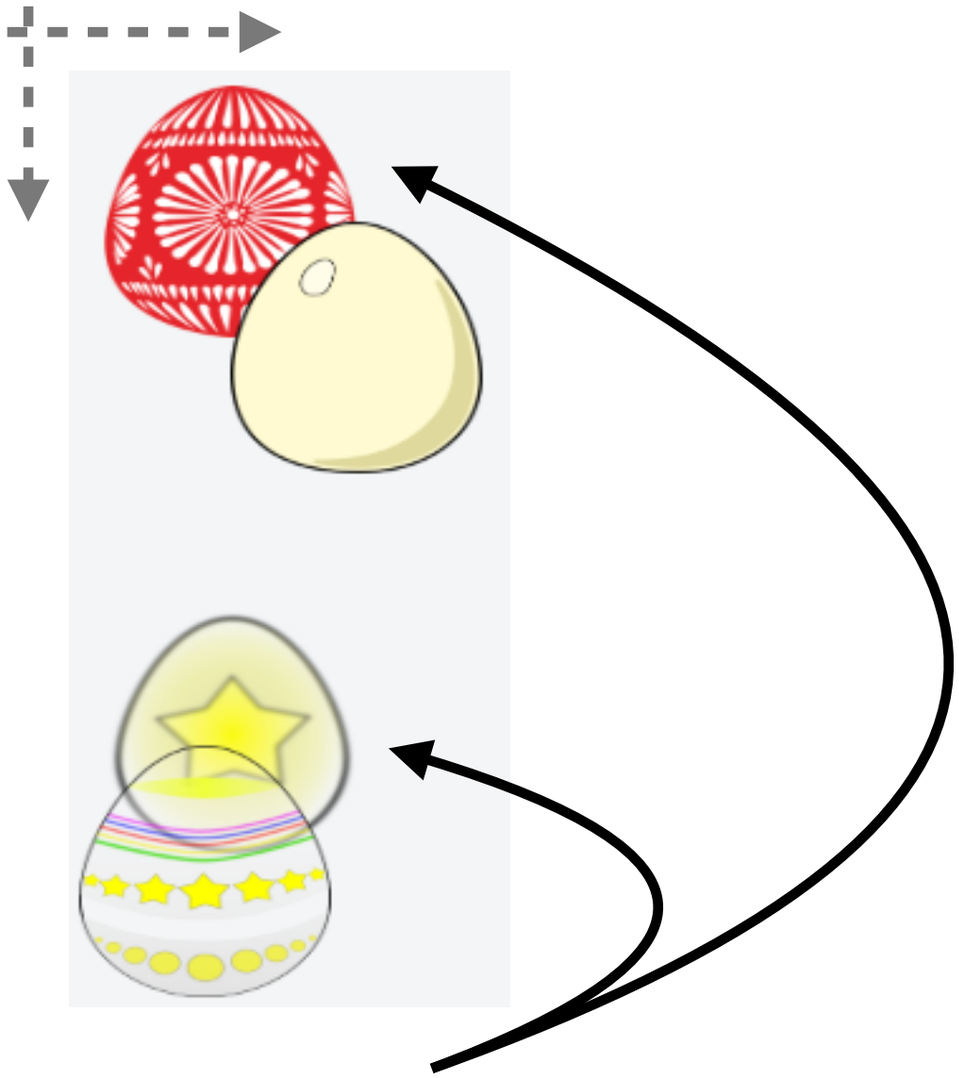
Bewegung **relativ** zur **Originalposition**

id="egg1"



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

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

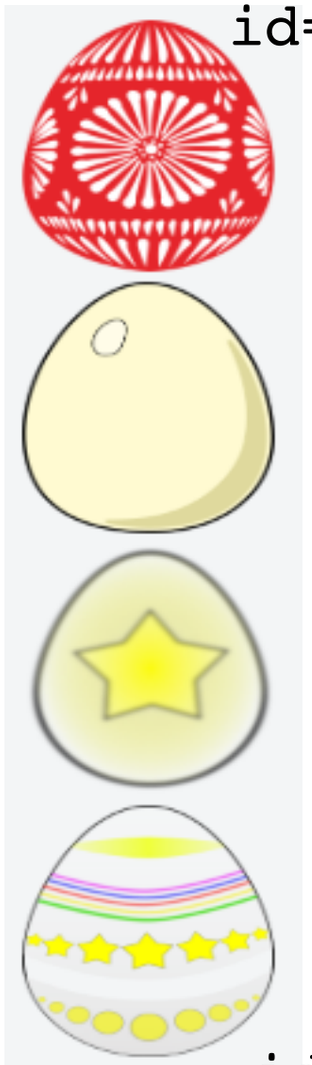
id="egg4"



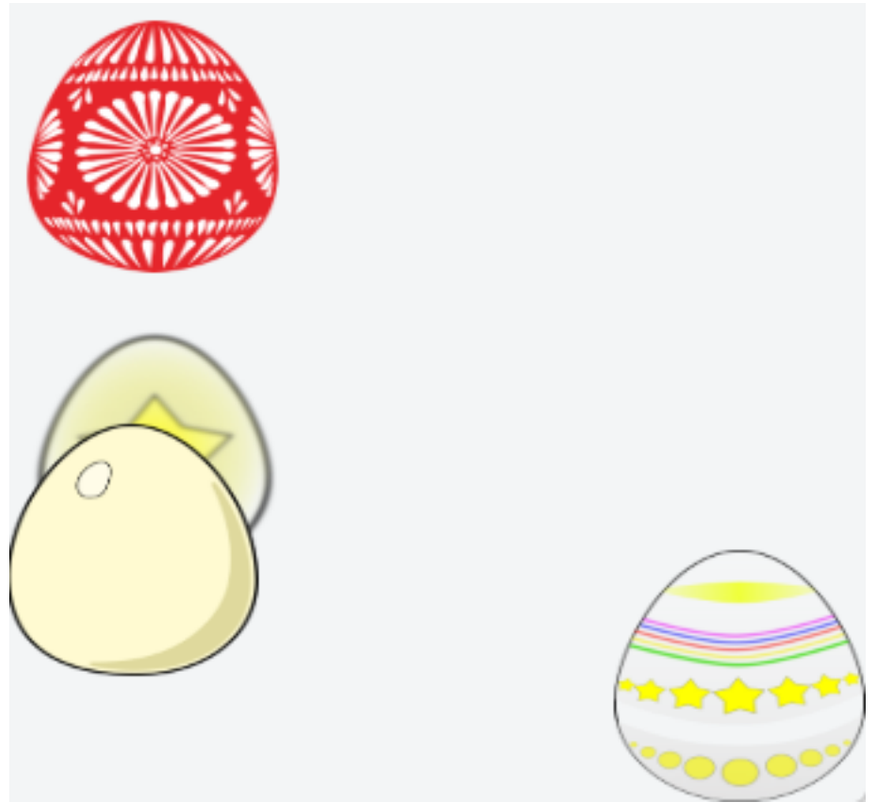
position:absolute

Element wird aus dem Fluß genommen (keine Platzreservierung)

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




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

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



`position:fixed`

Ähnlich wie "absolut", aber das umliegende “Element” ist der Viewport

Fläche des Dokuments, die im Browser sichtbar ist

Elemente mit **`position:fixed`** sind immer sichtbar

CSS Media Queries

Es gibt viele Arten von Geräten

- Verschiedene Geräte sollten verschiedene Styles bekommen
  - **Drucken** einer TODO Liste: schwarzweiß ohne farbige Blöcke
  - **Ansehen** einer TODO-Liste auf **kleinem Bildschirm**: nicht-essentielle Informationen entfernen (Footer etc.)
  - **Ansehen** einer TODO-Liste auf **großem Bildschirm**: alle Informationen anzeigen
  - **Text-to-Speech** Geräte: nicht-essentielle Informationen entfernen
- **CSS Media Queries** erlauben die Nutzung geräteabhängiger Stylesheets (genauer: abhängig vom Medientyp)

HTML: nur einmal schreiben

CSS: einmal pro Gerät schreiben

Media Queries können komplex sein

```
<link rel="stylesheet"
  media="screen and (min-width: 800px),
    (min-width: 3000px)"
  href="large-device.css">

<style>
  @media print {
    body {
      color: black !important;
      width: 100%;
      }
    }
  @media screen and (max-width: 300px) {
    #sidebar {
      display: none;
    }
  }
</style>
```

Media Queries können komplex sein

```
<link rel="stylesheet"
  media="screen and (min-width: 800px),
    (min-width: 3000px)"
  href="large-device.css">

<style>
  @media print {
    body {
      color: black !important;
      width: 100%;
      }
    }
  @media screen and (max-width: 300px) {
    #sidebar {
      display: none;
    }
  }
</style>
```

",": logisches "oder"

spezielle CSS Dateien

Regeln für verschiedene Geräte in einer Datei

Zum Drucken schwarzweiß nutzen

"and": logisches "und"

Verstecke Sidebar auf kleinen Geräten

https://developer.mozilla.org/en-US/docs/Web/CSS/Media_Queries/Using_media_queries

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***
    Learning Web App Development, Kapitel 4