Webentwicklung

Frontend: CSS

Inhalt dieser Einheit

1. Motivation & Geschichte
2. Syntax und Grundgedanken
3. Die Kaskade
4. Typographie
5. Layout
6. Die vielen Möglichkeiten von CSS
7. Debugging

Motivation

Und ein bisschen Geschichte

Schon kennengelernt

- Bootstrap-Klassen zur Hervorhebung von Elementen

```
<p class="lead">...</p>
<div class="bg-info">...</div>
```

- Bootstrap-Klassen für Layout und Responsiveness

```
<div class="col-xs-6 col-md-4">...</div>
<span class="hidden-xs">...</span>
```

- Icons mit Font Awesome:

```
<i class="fa fa-bug"></i>
```

CSS Zen Garden: <200 Zeilen HTML

CSS Zen Garden

The Beauty of CSS Design

A demonstration of what can be accomplished through CSS-based design. Select any style sheet from the list to load it into this page.

Download the example html file and css file

The Road to Enlightenment

Littering a dark and dreary road lay the past relics of browser-specific tags, incompatible DOMs, broken CSS support, and abandoned browsers.

We must clear the mind of the past. Web enlightenment has been achieved thanks to the tireless efforts of folk like the W3C, WaSP, and the major browser creators.

The CSS Zen Garden invites you to relax and meditate on the important lessons of the masters. Begin to see with clarity. Learn to use the time-honored techniques in new and invigorating fashion. Become one with the web.

So What is This About?

There is a continuing need to show the power of CSS. The Zen Garden aims to excite, inspire, and encourage participation. To begin, view some of the existing designs in the list. Clicking on any one will load the style sheet into this very page. The HTML remains the same, the only thing that has changed is the external CSS file. Yes, really.

CSS allows complete and total control over the style of a hypertext document. The only way this can be illustrated in a way that gets people excited is by demonstrating what it can truly be, once the reins are placed in the hands of those able to create beauty from structure. Designers and coders alike have contributed to the beauty of the web; we can always push it further.

Participation

Strong visual design has always been our focus. You are modifying this page, so strong CSS skills are necessary too, but the example files are commented well enough that even CSS novices can use them as starting points. Please see the CSS Resource Guide for advanced tutorials and tips on working with CSS.

You may modify the style sheet in any way you wish, but not the HTML. This may seem daunting at first if you've never worked this way before, but follow the listed links to learn more, and use the sample files as a guide.

Download the sample HTML and CSS to work on a copy locally. Once you have completed your masterpiece (and please, don't submit half-finished work) upload your CSS file to a web server under your control. Send us a link to an archive of that file and all associated assets, and if we choose to use it we will download it and place it on our server.

Benefits

Why participate? For recognition, inspiration, and a resource we can all refer to showing people how amazing CSS really can be. This site serves as equal parts inspiration for those working on the web today, and gallery of future techniques we can all look forward to.

Requirements

Where possible, we would like to see mostly CSS 1 & 2 usage. CSS 3 & 4 should be limited to widely-supported elements only, or strong fallbacks should be provided. The CSS Zen Garden is about functional, practical CSS and not the latest bleeding-edge tricks viewable by 2% of the browsing public. The only real requirement we have is that your CSS validates.

Luckily, designing this way shows how well various browsers have implemented CSS by now. When sticking to the guidelines you should see fairly consistent results across most modern browsers. Due to the sheer number of user agents on the web these days — especially when you factor in mobile — pixel-perfect layouts may not be possible across every platform. That's okay, but do test in as many as you can. Your design should work in at least IE9+ and the latest Chrome, Firefox, iOS and Android browsers (run by over 90% of the population).

We ask that you submit original artwork. Please respect copyright laws. Please keep objectionable material to a minimum, and try to incorporate unique and interesting visual themes to your work. We're well past the point of needing another garden-related design.

This is a learning exercise as well as a demonstration. You retain full copyright on your graphics (with limited exceptions, see submission guidelines), but we ask you release your CSS under a Creative Commons license identical to the one on this site so that others may learn from your work.

By Dave Shea. Bandwidth graciously donated by mediatemple. Now available: Zen Garden, the book.

HTML CSS CC A11y GH

Select a Design:

- Mid Century Modern by Andrew Lohman
- Garments by Dan Mall
- Steel by Steffen Knoeller
- Apothecary by Trent Walton
- Screen Filler by Elliot Jay Stocks
- Fountain Kiss by Jeremy Carlson
- A Robot Named Jimmy by meltmedia
- Verde Moderna by Dave Shea

Archives:

- Next Designs ›
- View All Designs

Resources:

- View This Design's CSS
- CSS Resources
- FAQ
- Submit a Design
- Translations

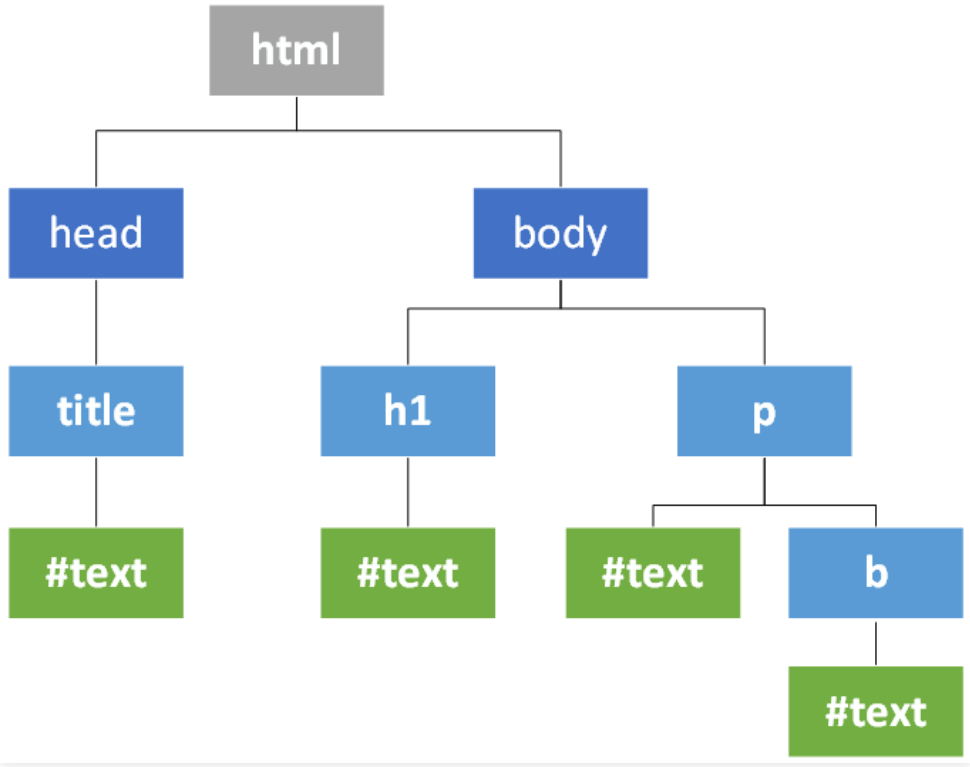
Quelle: http://www.csszengarden.com/examples/index

CSS Zen Garden: 200+ CSS-Designs

...

Quelle: http://www.mezzoblue.com/zengarden/alldesigns/

Geschichte: Fehlentwicklungen



- Idee von HTML: Markup für inhaltliche Struktur, nicht Optik
- Aber: Neuling vermissen Desktop-Möglichkeiten
  - neue Elemente: `<font>`, `<color>`, `<blink>`, `<center>`
  - neue Attribute: `bgcolor` oder `border`
- schlecht für Pflege und maschinelle Verarbeitung

Rettung: CSS

- 10/1994: Cascading HTML style sheets – a proposal
  - Ziel: Trennung von Darstellung und Inhalt
- Einige Ideen waren gut, andere wurden verworfen
  - Gut: deklarativ, regelbasiert, vererbend
- 1998: erster ACID-Test
  - um Standardkonformität zu testen

Quelle: http://commons.wikimedia.org/wiki/File:Acid-Css-Test.png

Entwicklung von CSS

- 12/1996: CSS Level 1
- 05/1998: CSS Level 2
  - Bis Anfang 2010: von keinem verbreiteten Webbrowser vollständig umgesetzt
- 2002-2011: CSS Level 2 Revision 1
- seit 2000: CSS Level 3
- Kein CSS 4
  - CSS 3 ist modular; nur dessen Module werden weiterentwickelt
  - D.h. CSS-Spezifikation ist auf 15+ Dokumente verstreut

Syntax und Grundgedanken

Konzepte

- deklarativ:
  - In CSS werden **Regeln** ausgedrückt
  - Regeln werden auf DOM-**Elemente** angewendet
- vererbend:
  - Regeln gelten allgemein auch für Kinder im DOM

Syntax: abstrakt

```
/* Kommentar */
Selektor [, Selektor-2, ...] {
    Regel;
    Regel-1;
    /* ... */
    Regel-n;
}
```

- CSS-Anweisung:
  - Selektor: Auswahl von DOM-Elementen
  - Regel: wird auf ausgewählte Elemente angewendet
- Formatierung
  - letztes Semikolon muss nicht, sollte aber
  - Whitespaces haben keine Bedeutung
  - Kommentare: Einzige Form, keine einzeiligen (`//` o.ä.)

Syntax: Beispiel

```
body { /* Selektor: das body-Element */
   font-size: 19px;   /* Regel 1 */
   line-height: 1.4;  /*        2 */
   max-width: 600px;  /*        3 */
   margin: 0 auto;    /*        4 */
}
```

- **Regel**: Schlüssel-Wert-Paar
  - getrennt durch Doppelpunkt
- **Schlüssel**: Eigenschaftsname (CSS-Property)
- **Werte**:
  - erlaubte Werte hängen vom Property ab
  - manchmal mehrere Werte möglich, durch Leerzeichen getrennt

Selektoren & Kombinatoren

Selektoren

```
      * { /* jedes Element                          */ }
     h1 { /* alle H1-Überschriften                  */ }
 .active { /* alle Elemente mit der Klasse "active" */ }
#wrapper { /* das Element mit der ID "wrapper"      */ }
img[alt] { /* alle Bilder mit einem ALT-Attribut    */ }

h1.active { /* H1-Elemente mit der Klasse "active" */ }
```

- Bedingungen (bzw. Kombination daraus; ohne Blank)
  - zum Ansprechen von DOM-Elementen
- Verschiedene Möglichkeiten:
  - Elementname, Attribute (`class`, `id`, etc.)
  - oder Position im DOM
- Umfassende Listen:
  - Wikipedia
  - SELFHTML

Selektoren – Kombinatoren

- Kombination: Verbindung zweier Selektoren
  - Zur “Navigation” im DOM

```
/* Nachfahren – alle Links in Navigation */
nav a {  }

/* Kinder – alle direkten Nachfahren von Nav */
nav > a {  }

/* Nachbarn – alle Absätze,
   die direkt auf eine H1-Überschrift folgen */
h1 + p { }

/* Geschwister – alle Paragraphen,
   die auf gleicher Ebene auf eine H1-Überschrift folgen */
h1 ~ p { }
```

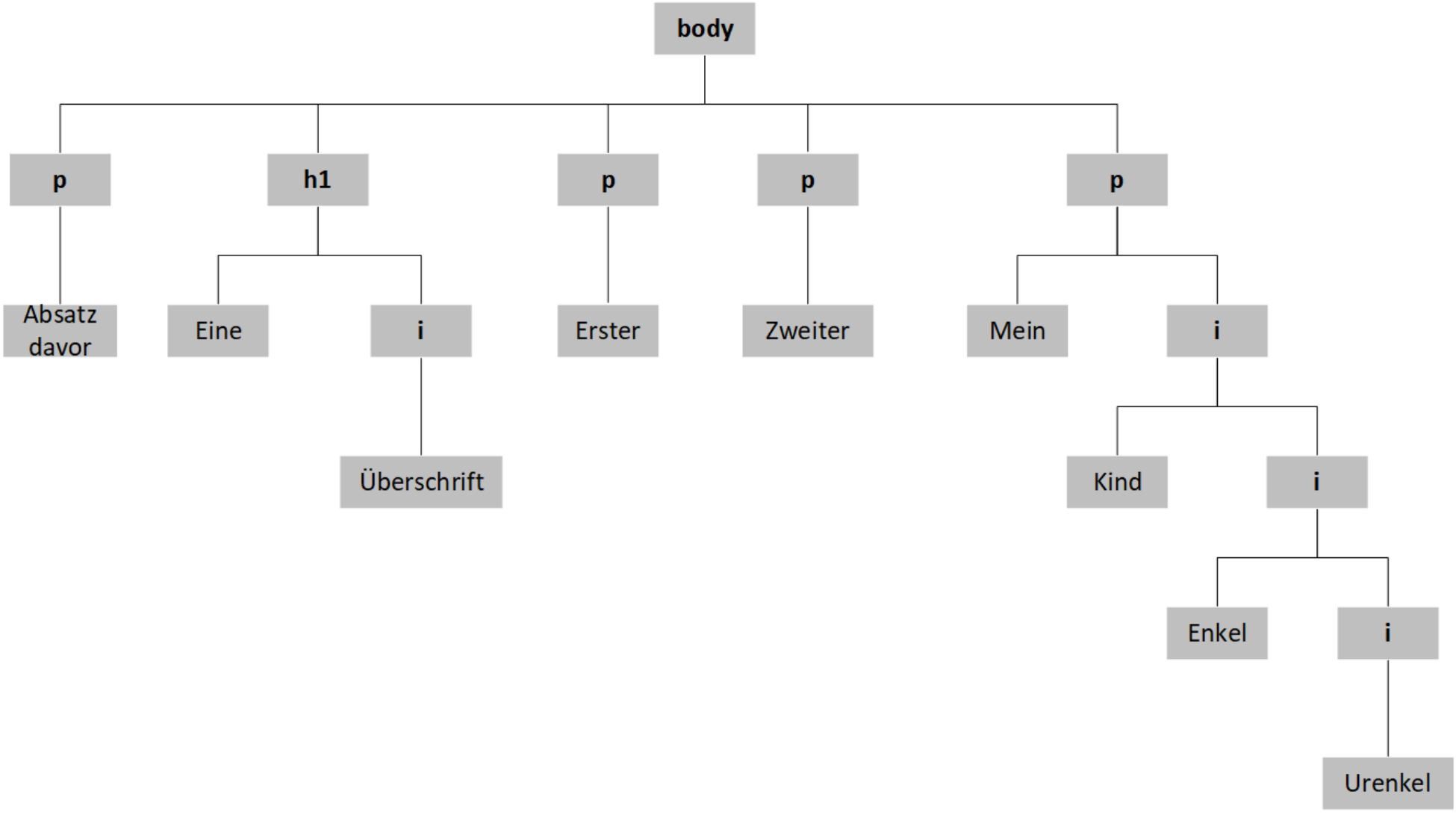
Kombinatoren: Beispiel

```
<p>Absatz davor</p>
<h1>Eine <i>Überschrift</i></h1>
<p>Erster Absatz</p>
<p>Zweiter Absatz</p>
<p>Mein
  <i>Kind,
    <i>Enkel, <i>Urenkel</i></i>
  </i>
</p>
```

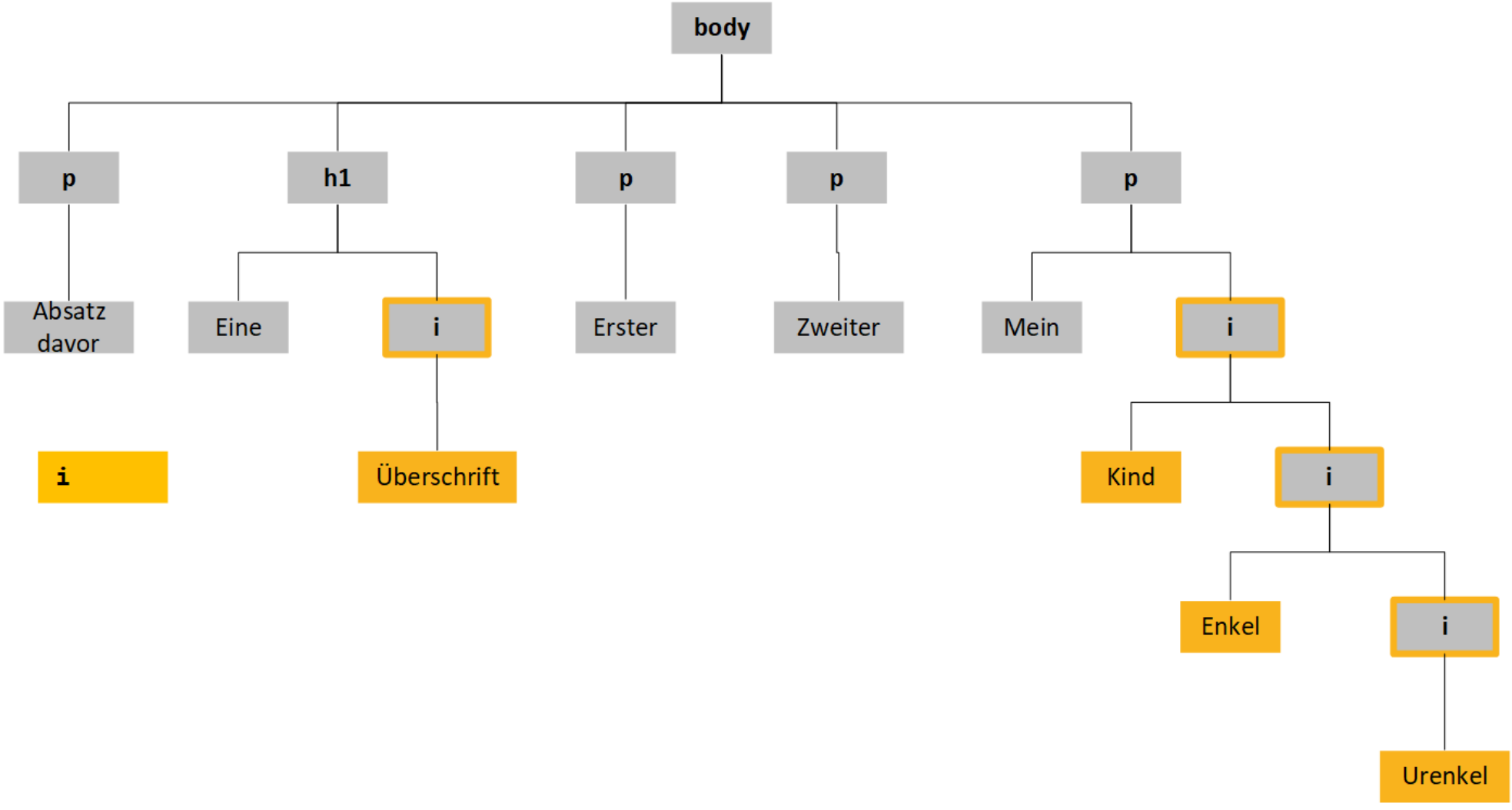
```
i       { color: gold; }
h1 ~ p  { color: red; }
h1 + p  { color: green; }
p i     { color: hotpink; }
p > i   { color: skyblue; }
```

- Preisfrage: Welche Text-Elemente haben welche Farbe?

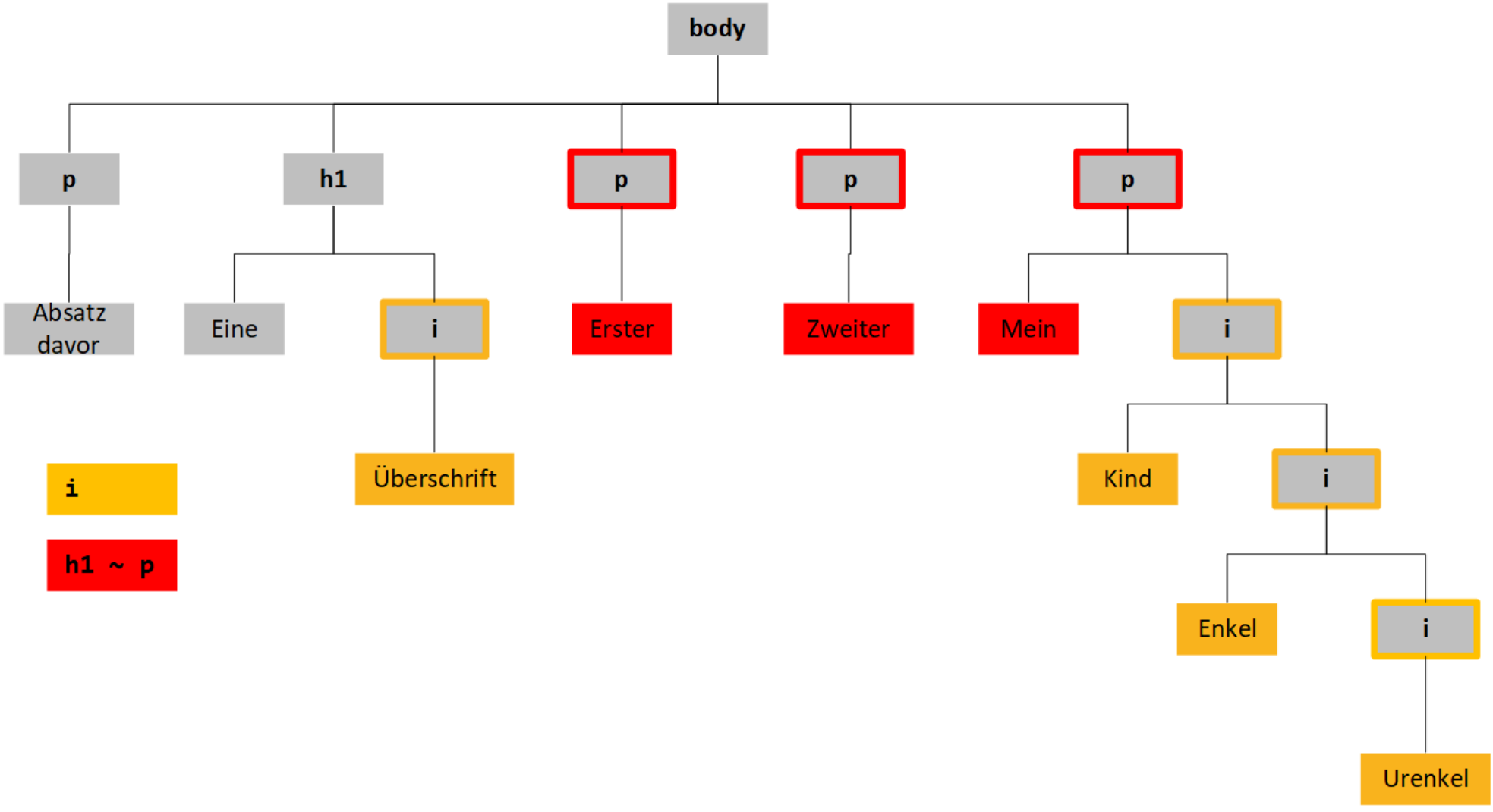
Kombinatoren-Beispiel: DOM



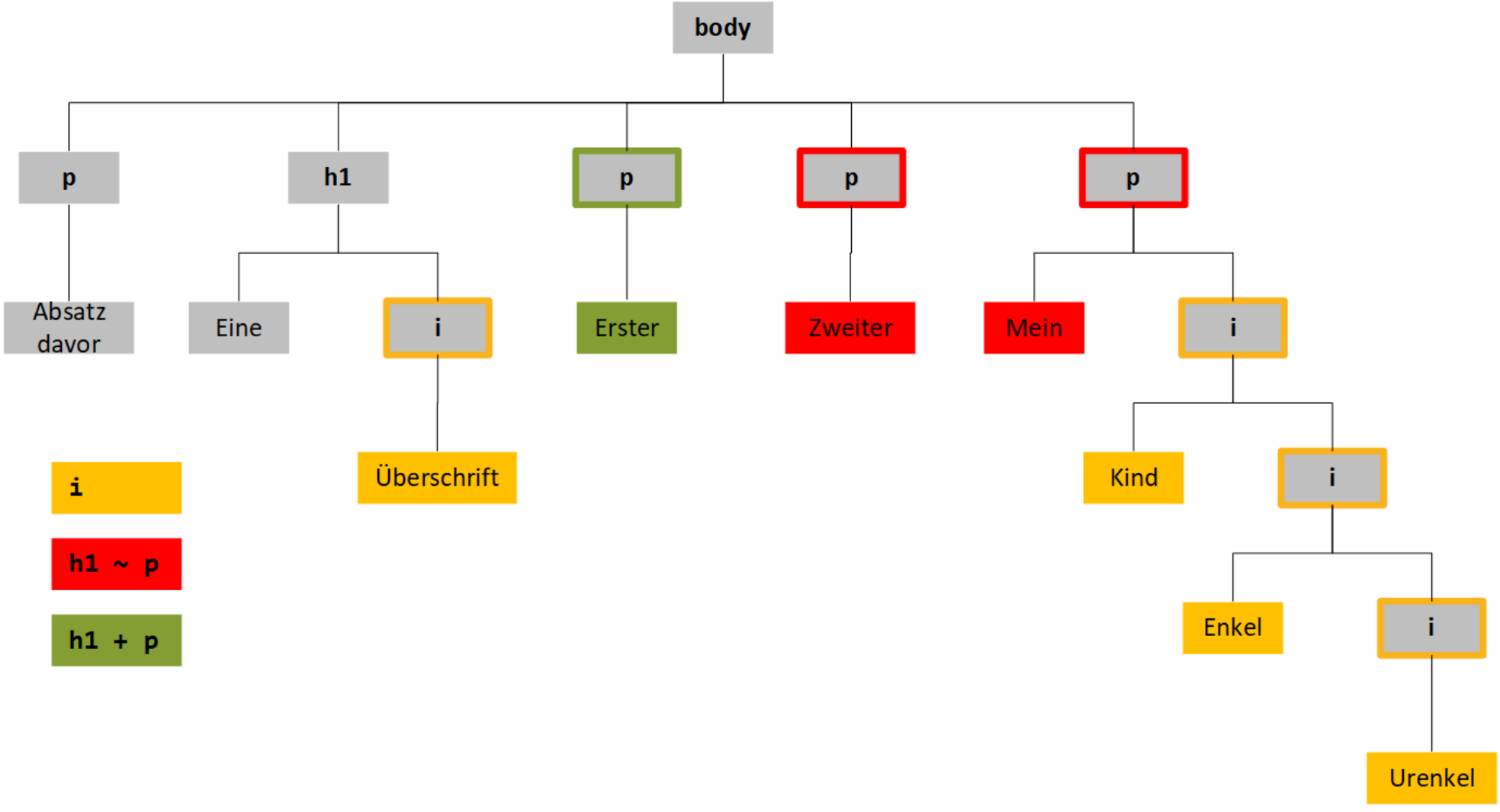
Kombinatoren-Beispiel: `i`



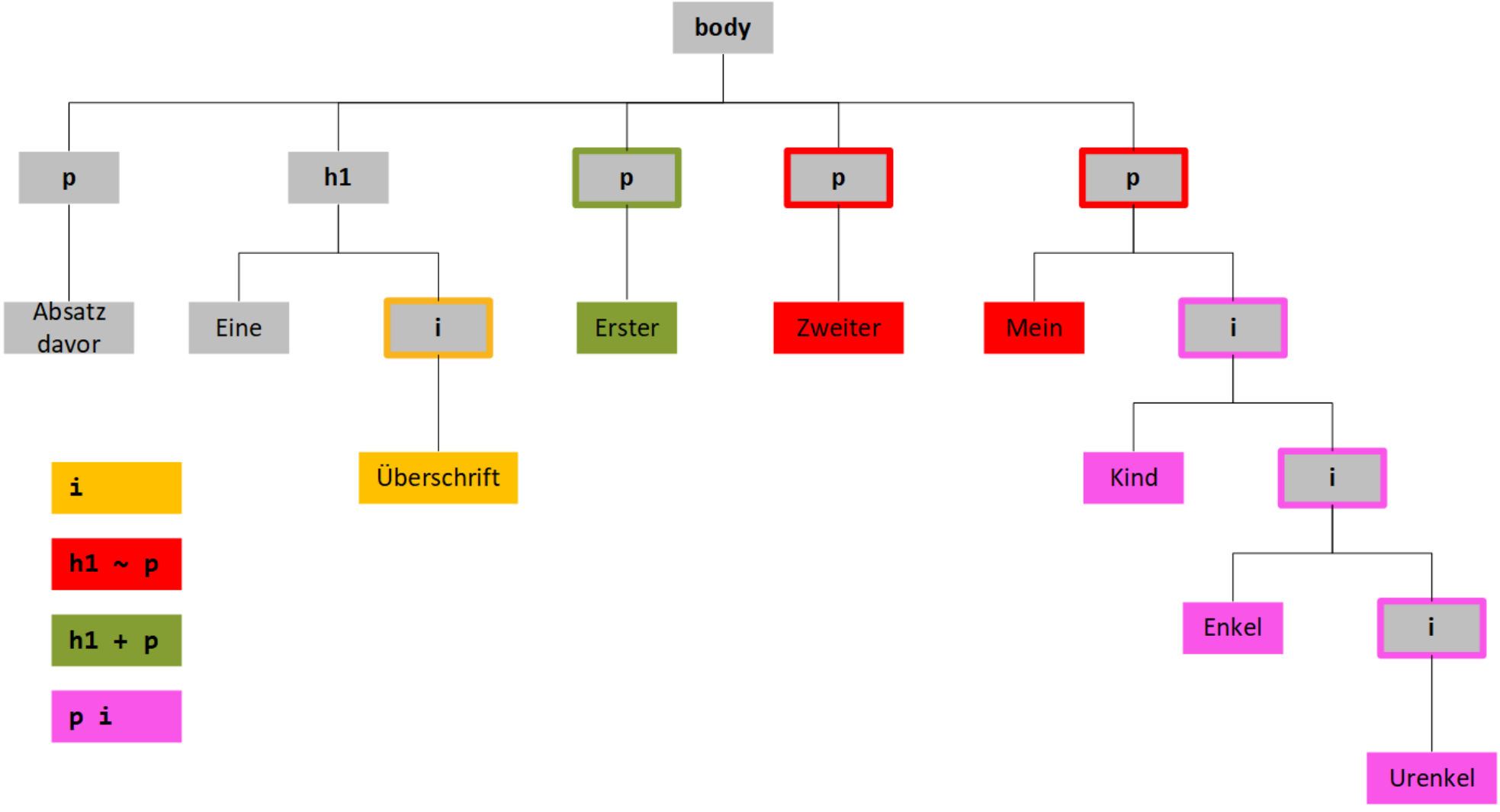
Kombinatoren-Beispiel: `h1 ~ p`



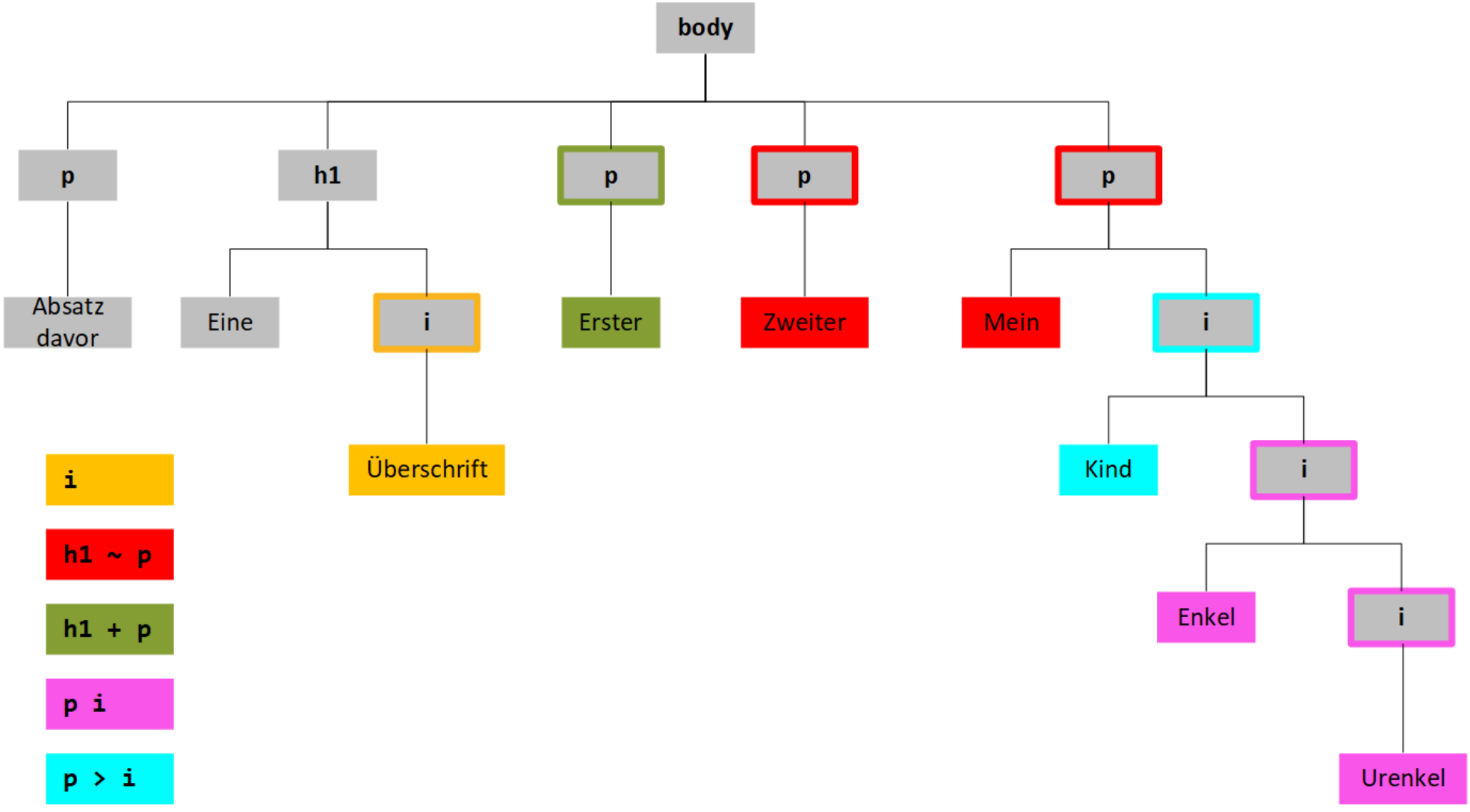
Kombinatoren-Beispiel: `h1 + p`



Kombinatoren-Beispiel: `p i`



Kombinatoren-Beispiel: `p > i`



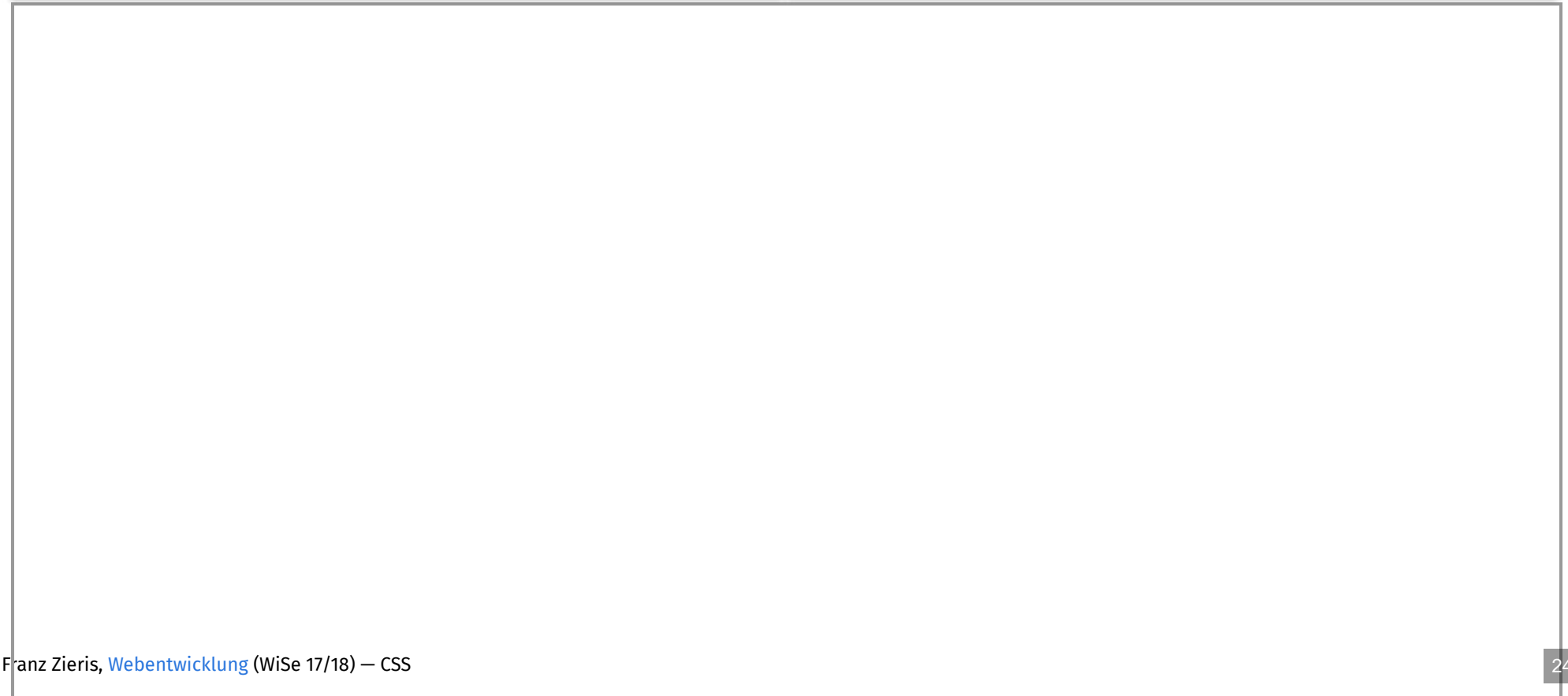
Kombinatoren-Beispiel: Ergebnis

HTML

```html
<p>Absatz davor</p>
<h1>Eine <i>Überschrift</i></h1>
<p>Erster Absatz</p>
<p>Zweiter Absatz</p>
<p>Mein <i>Kind, <i>Enkel,
  <i>Urenkel</i></i></i></p>
```

CSS

```css
body    { font-size: 150%; }
i       { color: gold; }
h1 ~ p  { color: red; }
h1 + p  { color: green; }
p i     { color: hotpink; }
p > i   { color: skyblue; }
```

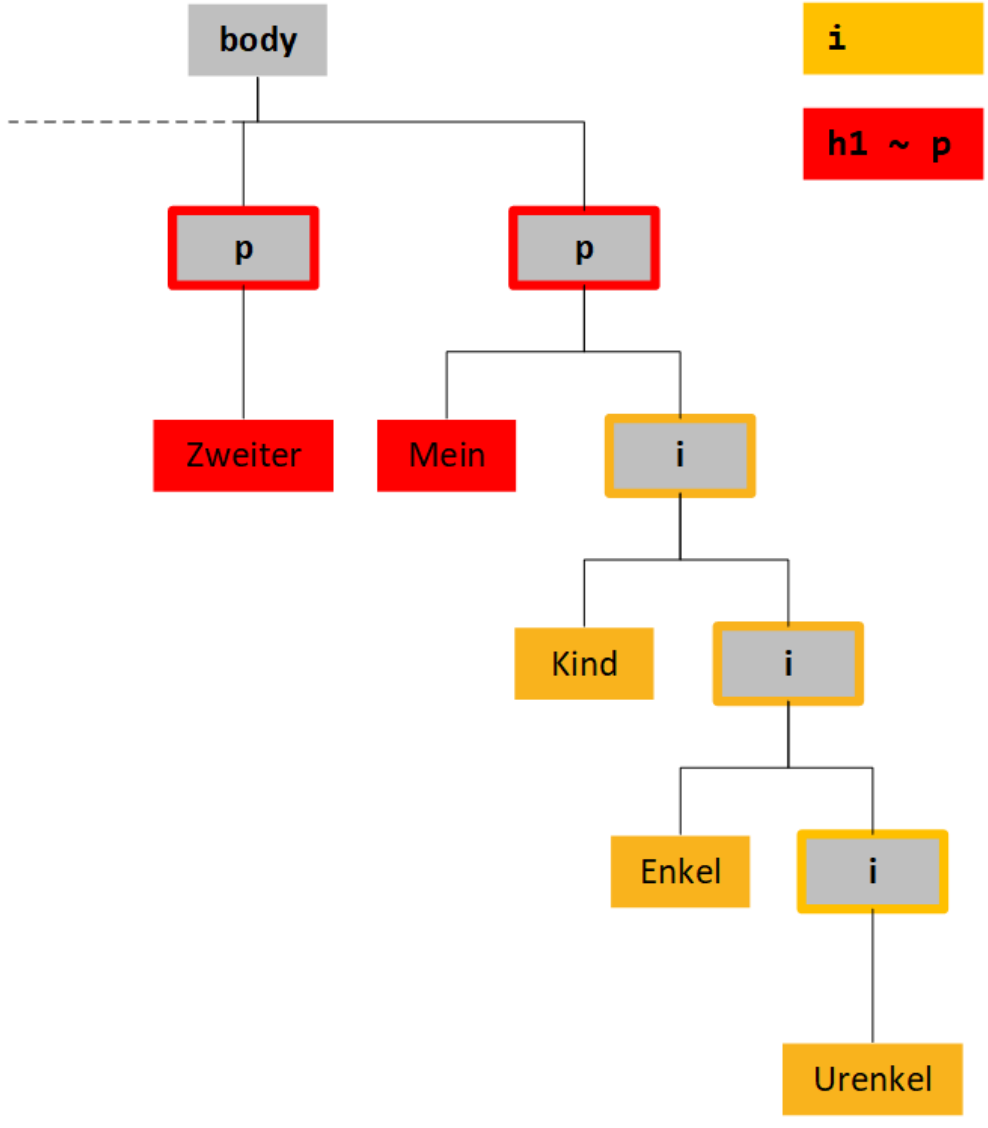


Moooooment …

Da war doch was!

Kombinatoren-Beispiel: `h1 ~ p`

- Wir sind die Regeln der Reihe nach durchgegangen
  - Aber: Warum wird der Absatz nicht komplett rot?
- CSS: kompliziertes System zur Ermittlung der konkreten Eigenschaften



Die Kaskade

Cascading Stylesheets

- Prüfung pro Element anhand der Selektoren:
  - keine Deklaration → Erben von Vorfahren im DOM

```
body { font-size: 18px; }
```

```
<p></p> <!-- Erbt Schriftgröße von 18px -->
```

  - genau eine Deklaration → Verwenden
  - mehrere Deklarationen → **Kaskade**
    1. nach Wichtigkeit (Ursprung, und notfalls `!important`)
    2. nach Spezifizität der Regel
    3. nach Reihenfolge (letzte Regel gilt)
- Eins der kompliziertesten Konzepte in CSS!

Quelle: https://little-boxes.de/artikelansicht/die-kaskade-im-schnelldurchgang.html

Spezifizität von Selektoren

- Gelten mehrere Selektoren für ein Element, gewinnt das spezifischste
- *IDs* zählen mehr als *Klassen* & *Attribute* zählen mehr als *Elementnamen*
  - pro Selektor Bestandteile zählen
  - Score: `#ids-#classes-#names`
  - Inline-Regeln sind am spezifischsten

Entscheidung bei Elementnamen

```HTML
<p><a>Link</a></p>
```

```CSS
body { font-size: 200%; }
p a { color: orange; } /* 0-0-2 */
a   { color: blue; }   /* 0-0-1 */
```

- Zwei Selektoren gelten für das `a`-Element
- Zwei Elementnamen `0-0-2` schlagen einen Elementnamen `0-0-1`

Entscheidung bei Klassen & Namen

```HTML
<p><a class="green">Link</a></p>
```

```CSS
body   { font-size: 200%; }
.green { color: green; }  /* 0-1-0 */
p a    { color: orange; } /* 0-0-2 */
a      { color: blue; }   /* 0-0-1 */
```

Link

- Drei Selektoren gelten für das `a`-Element
- Eine Klasse `0-1-0` schlägt Elementnamen (egal wie viele)

Entscheidung mit ID

```HTML
<p><a class="green" id="red">Link</a></p>
```

```CSS
body   { font-size: 200%; }
#red   { color: red; }    /* 1-0-0 */
.green { color: green; }  /* 0-1-0 */
p a    { color: orange; } /* 0-0-2 */
a      { color: blue; }   /* 0-0-1 */
```

Link

- Vier Selektoren gelten für das `a`-Element
- Eine ID `1-0-0` schlägt wiederum Klassennamen

Entscheidung mit ID und Klassen

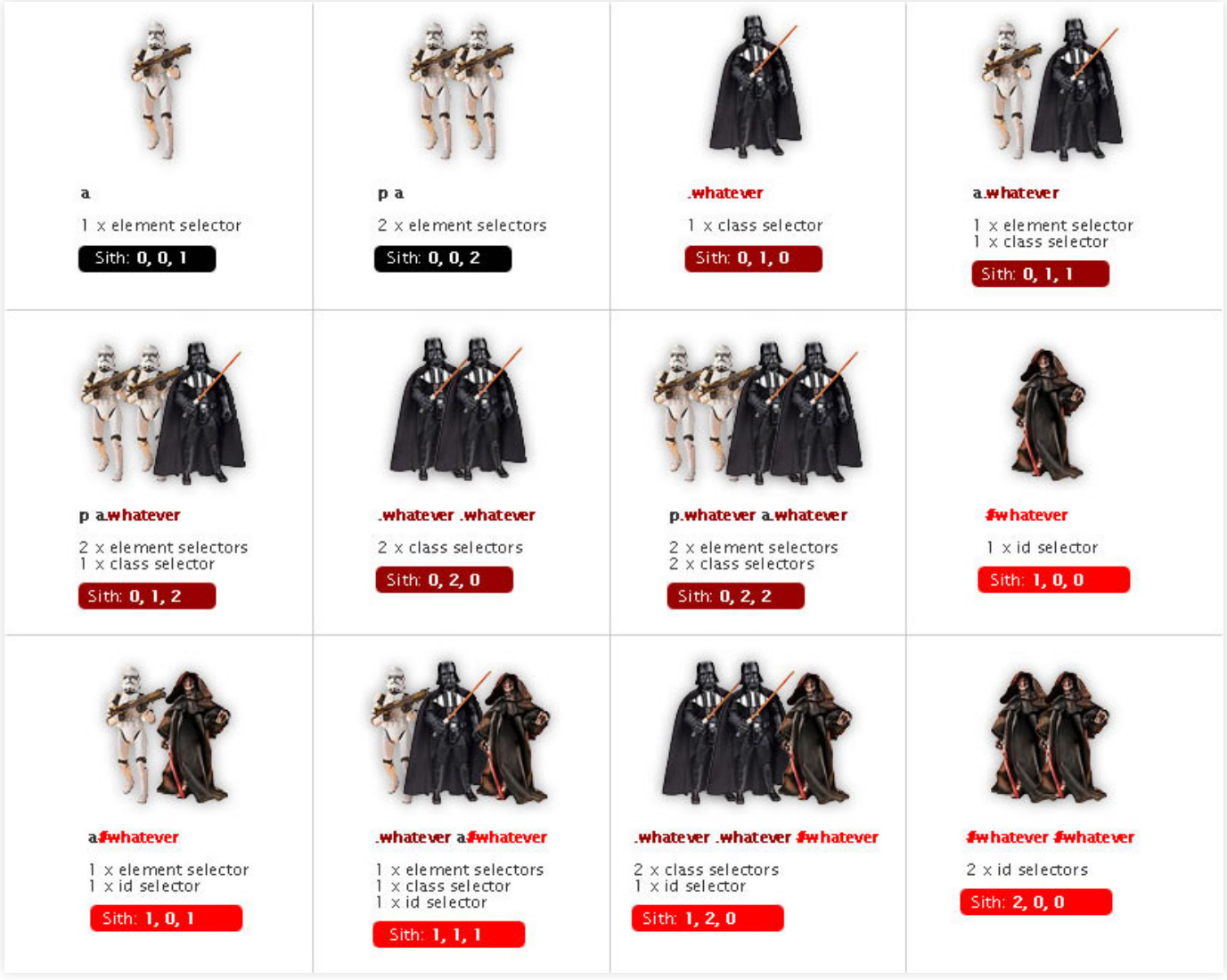
```HTML
<p class="green"><a class="green" id="red">Link</a></p>
```

```CSS
body   { font-size: 200%; }
#red         { color: red; }    /* 1-0-0 */
.green #red  { color: brown; }  /* 1-1-0 */
.green       { color: green; }  /* 0-1-0 */
p a          { color: orange; } /* 0-0-2 */
a            { color: blue; }   /* 0-0-1 */
```

Link

- Fünf Selektoren gelten für das `a`-Element
- Eine ID mit einem Klassennamen `1-1-0` schlägt eine ID ohne Klassennamen `1-0-0`

Specificity Wars



Quelle: http://www.stuffandnonsense.co.uk/archives/css_specificity_wars.html

Ursprung von Regeln

- **Ursprung** ist wichtiger als **Spezifizität**; fünf Stufen
  1. Regel aus Anwender-Stylesheet, mit `!important`
  2. Regel aus HTML-Dokument, mit `!important`
  3. Regel aus HTML-Dokument, ohne `!important`
  4. Regel aus Anwender-Stylesheet, ohne `!important`
  5. Regel aus Browser-Stylesheet
- Für Stufen 2 und 3 ist die Art der Einbindung egal

Quelle: https://little-boxes.de/lb1/11.3-sortiere-nach-wichtigkeit.html

Einbindung von CSS

- CSS kennt verschiedene Arten der Einbindung
  - **Extern** (`head`)

```
<link rel="stylesheet" href="style.css">
```

```
#red { color: red; } /* 1-0-0 */
```

    - Regel kommt zwar von “außerhalb”, ist aber spezifisch
  - **Intern** (`head` oder `body`)

```
<!-- Internes CSS (head oder body) -->
<style> .green { color: green; } /* 0-1-0 */ </style>
```

  - **Inline**:

```
<span id="red" class="green">Rot</span>
<span id="red" class="green" style="color: blue;">Blau</span>
```

    - Inline-Regeln haben keine Selektoren nötig: sind “am spezifischsten”
- Live-Demo

Einbindung: Abwägung

- Zu berücksichtigende Punkte
  - Ladezeiten/Netzwerklast
    - **extern**: Extra-Request, aber cache-bar
  - Änderbarkeit/Lesbarkeit
    - **inline**: keine Klassennamen, keine Trennung von Design und Inhalt
    - **intern**: schwache Trennung von Design und Inhalt
  - Lokalität/Lesbarkeit
    - **extern, intern & inline gemischt**: Informationen verstreut
  - Effektgröße
    - **extern**: evtl. mehrere Dokumente
    - **intern**: nur aktuelles Dokument
    - **inline**: nur aktuelles Element

Nochmal: Die Kaskade

- Prüfung pro Element anhand der Selektoren:
  - keine Deklaration → Erben von Vorfahren im DOM
  - genau eine Deklaration → Verwenden
  - mehrere Deklarationen → **Kaskade**
    1. nach Wichtigkeit (Ursprung)
       - *Dokument* > *Anwender* > *Browser*, notfalls überschreiben mit `!important`
    2. nach Spezifizität der Regel
       - inline ist am spezifischsten
       - *ID* > *Klassen* & *Attribute* > *Elementnamen*
    3. nach Reihenfolge (letzte Regel gilt)

Quelle: https://little-boxes.de/artikelansicht/die-kaskade-im-schnelldurchgang.html

Uff … können wir endlich

konkrete **Eigenschaften**

sehen?

Typographie

Typgraphie

```HTML
Lorem ipsum dolor sit amet ...
<p>Lorem ipsum dolor sit amet ...</p>
```

```CSS
p {
  font-size: 150%;
  color: red;
  font-family: 'Verdana',
    'Arial', sans-serif;
  font-weight: bold;
  font-style: italic;
  font-variant: small-caps;
  text-decoration: underline;
  text-align: right;
  letter-spacing: 5px;
  line-height: 2;
}
```

Lorem ipsum dolor sit amet ...

LOREM IPSUM DOLOR SIT AMET ...

Typographie: Schriftarten

- Google Fonts: https://fonts.google.com/
  - 800+ Schriftfamilien (= mehrere Varianten einer Schriftart)
  - Einbindung über CSS, z.B.

```
<link href="https://fonts.googleapis.com/css?family=Lato" rel="stylesheet">
```

```
body {
  font-family: 'Lato', sans-serif;
}
```

  - Alternativ: Einbindung über `@import`:

```
@import url('https://fonts.googleapis.com/css?family=Lato');
body {
  font-family: 'Lato', sans-serif;
}
```

`@import`-Beispiel

```html
<p>Normal paragraph</p>
<p class="lato">Let's <b>try</b> <strong>Lato</strong></p>
```

HTML

```css
@import url('https://fonts.googleapis.com/css?family=Lato:400,700,900
body          { font-size: 250%; }
.lato         { font-family: 'Lato', sans-serif; }
.lato strong { font-weight: 900; } /* bold = 700 */
```

CSS

Normal paragraph

Let's **try** **Lato**

Farbangaben

```
color: red;                     /* Farbname */
color: #ff0000;                 /* Hexadezimal RGB */
color: #f00;                    /* Hexadezimal RGB (kurz) */
color: rgb(255, 0, 0);          /* Dezimal RGB */
color: rgb(100%, 0%, 0%);       /* Prozentual RGB */
color: rgba(255, 0, 0, 1);      /* RGBa (a = alpha) */
color: hsl(0, 100%, 50%);       /* HSL (Hue, Saturation, Luminance) */
color: hsla(0, 100%, 50%, 1); /* HSL mit alpha */
```

- Analog für Properties `background-color`, `border-color`, …
- Siehe auch:
  - https://www.w3schools.com/colors/colors_hsl.asp

Quelle: https://developer.mozilla.org/de/docs/Web/CSS/color

Schriftgrößen

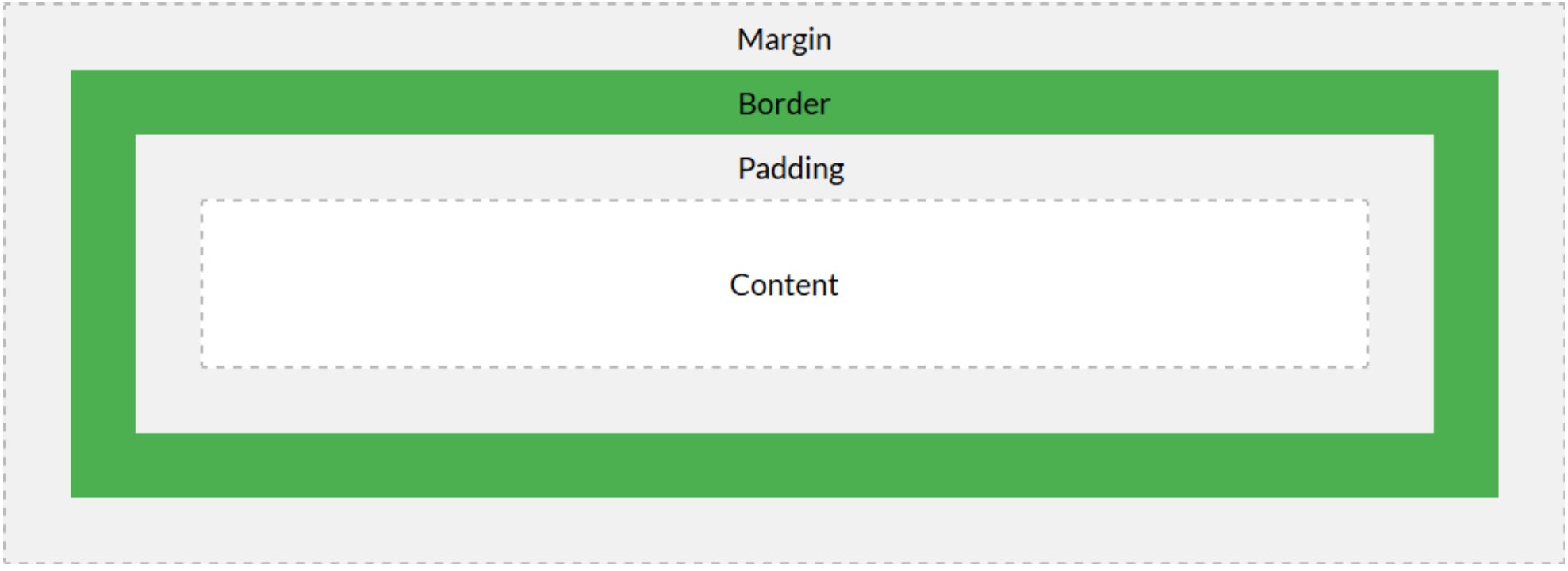
```HTML
<div style="font-size: 100%">
  <span>Standardgröße</span>
  <span style="font-size: 120%">rel. 20% größer</span>
  <span style="font-size: 1.5em">rel. 50% größer</span>
  <span style="font-size: 16px">abs. 16 Pixel</span>
  <span style="font-size: 16pt">abs. 16 Punkt</span>
</div>
```

Standardgröße rel. 20% größer rel. 50% größer abs. 16 Pixel abs. 16 Punkt

Layout & Element-Darstellung

Box Model

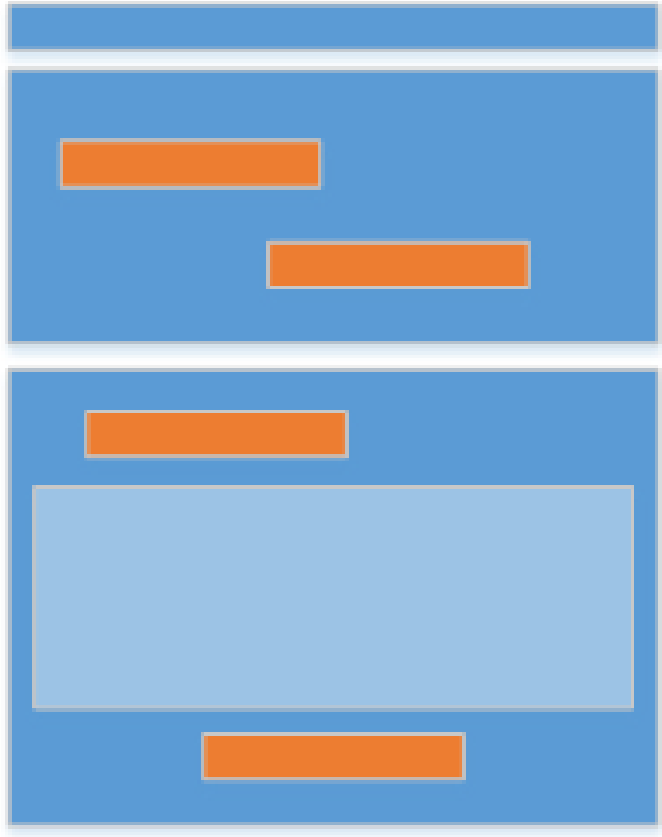
- Alle Elemente sind letztlich Boxen



Quelle: https://www.w3schools.com/css/css_boxmodel.asp

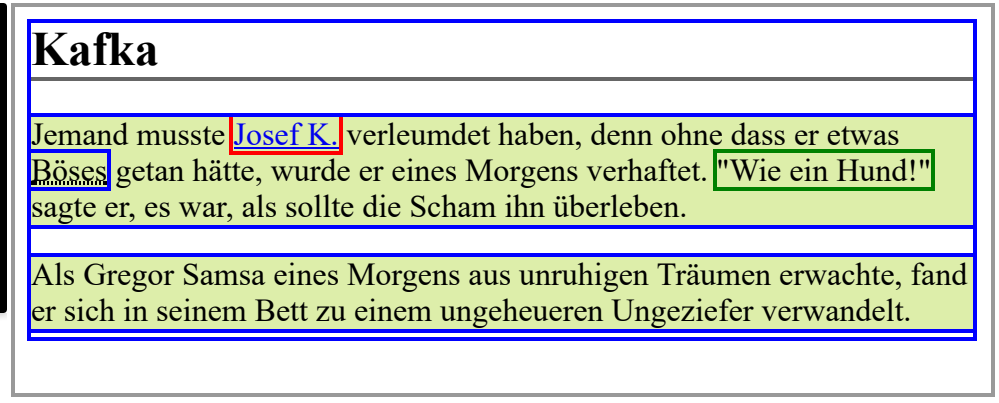
Element-Arten: Display

- **Block**
  - Eigener Absatz/Bereich im Dokument
  - Volle verfügbare Breite
  - Können selbst wieder Block- und Inline-Elemente enthalten
- **Inline**
  - Stehen im Fluss der anderen Elemente
  - Können Inline-Elemente enthalten



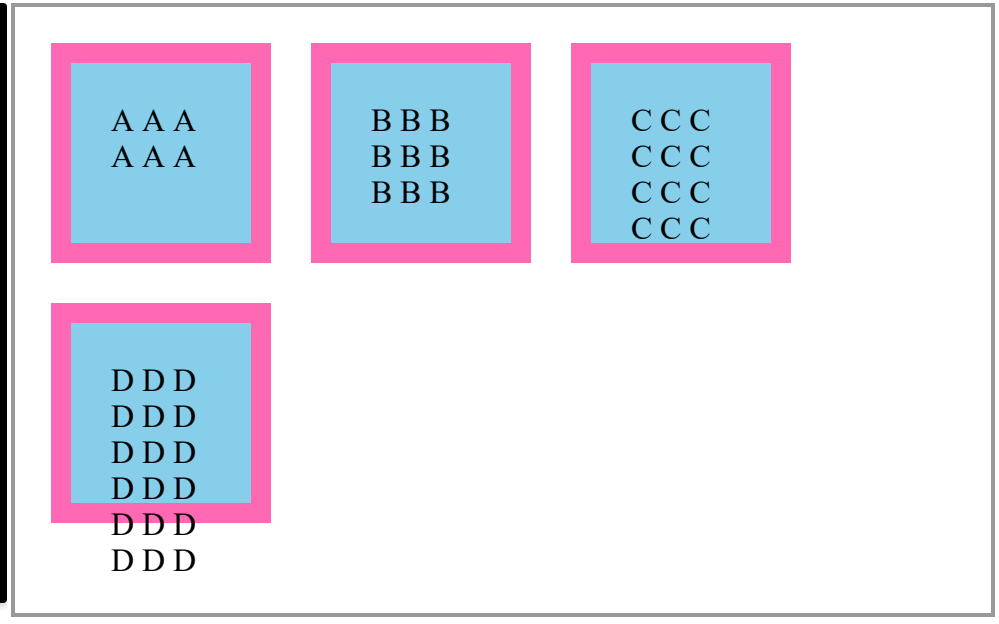
Blöcke sichtbar gemacht

```css
*  { outline: 2px solid blue; }
h2 { outline: 2px solid #666; }
p  { background-color: #DEA; }
a  { outline: 2px solid red; }
q  { outline: 2px solid green; }
```



Box Model

```css
.custom-box {
   width: 50px;
   height: 50px;
   margin: 10px;
   padding: 20px;
   border: 10px solid hotpink;
   background-color: skyblue;
   /* box-sizing: border-box; */
   float: left;
}
```



- Sollen Abmessungen “innen” oder “außen” gelten?
  - `box-sizing: border-box | content-box`

Floats

- `float`: hebt Element aus dem Fluss
  - an linke oder rechte Seite (`left`, `right`)
  - andere Inhalte umfließen
  - Floatende Elemente = Block-Elemente
- Clearing:
  - Nachfolgendes Element hat passende `clear`-Eigenschaft (`left`, `right`, `both`)

Floats

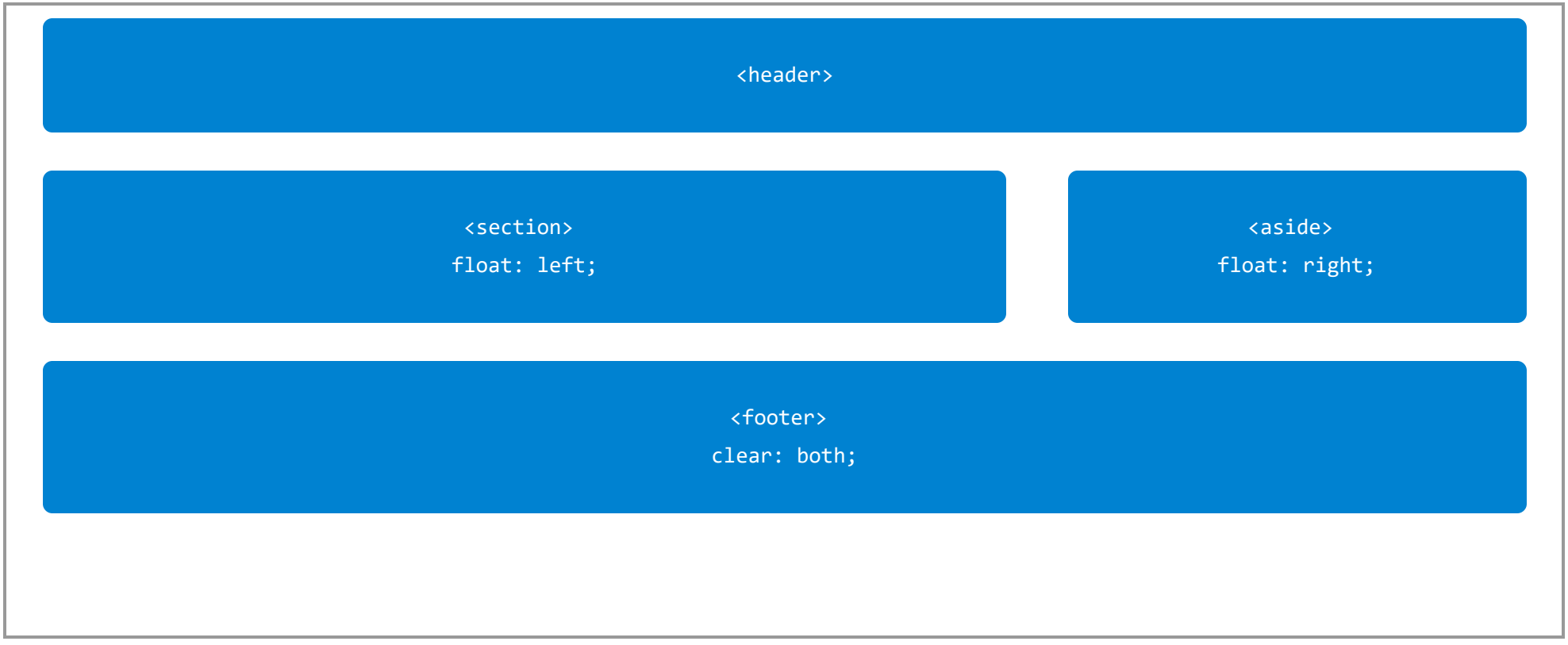
```css
h1 {
    font-size: 1.5em;
}

img {
   width: 50vw;
   float: right;
   margin-left: 1em;
   margin-bottom: .5em;
   border: 2px solid #ccc;
}
```

Die Leiden des jungen Werthers

Eine wunderbare Heiterkeit hat meine ganze Seele eingenommen, gleich den süßen Frühlingsmorgen, die ich mit ganzem Herzen genieße. Ich bin allein und freue mich meines Lebens in dieser Gegend, die für solche Seelen geschaffen ist wie die meine. Ich bin so glücklich, mein Bester, so ganz in dem Gefühle von ruhigem Dasein versunken, daß meine Kunst darunter leidet. Ich könnte jetzt nicht zeichnen, nicht einen Strich, und bin nie ein größerer Maler gewesen als in diesen Augenblicken.

Quelle: http://de.wikipedia.org/wiki/Datei:Donat_Werther_et_Lotte.jpg

Floats – Layouts



Quelle: Shay Howe, http://codepen.io/shayhowe/pen/utfmw

Positionierung

`position`	**Beschreibung**
`static`	Standardwert
`relative`	Bleibt in normalem Fluss und wird verschoben
`fixed`	Relativ zum Viewport des Browsers (beim Scrollen “fixiert”)
`absolute`	Relativ zu Vorfahren

Eigenschaften: `top`, `bottom`, `left`, `right`

Positionierung

HTML

```
<div class="abs o-r u-l">
  Überall
  <span class="abs o-r">
    oben rechts</span>
  <span class="abs u-l">
    unten links</span>
</div>
```

CSS

```
.abs {
  position: absolute;
  padding: 2em;
  background-color:
    rgba(0, 130, 209, .3);
}
.o-r { top: 0; right: 0 }
.u-l { bottom: 0; left: 0 }
```

Neue Layout-Mechanismen

- **Flexbox**:
  - eindimensional (entweder zeilen- oder spalten-basiert)
- **Grid** (experimentell)
  - zweidimensional
- Übersicht über alle CSS-Layout-Mechanismen:
  - https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Introduction

Flexbox

- Erster, ordentlicher Layout-Mechanismus in CSS
  - dynamische/s Layout, Anordnung und Verteilung
- Elemente werden in Höhe und Breite verändert, um vorhandenen Raum bestmöglich zu nutzen
- Sehr komplex, zu komplex für diese Einheit
  - ein paar Beispiele folgen
  - Mehr unter
    - Solved by Flexbox
    - A Complete Guide to Flexbox
    - Flexy Boxes: flexbox playground and code generator

Flexbox – Zentrierung

```html
<div class="zentrum"><p>Look ma, no hands!</p></div>
```
HTML

```css
.zentrum {
   display: flex;
   align-items: center;
   justify-content: center;
   height: 300px;
   background: green;
}

p {
   padding: 2em;
   background: gold;
}
```
CSS

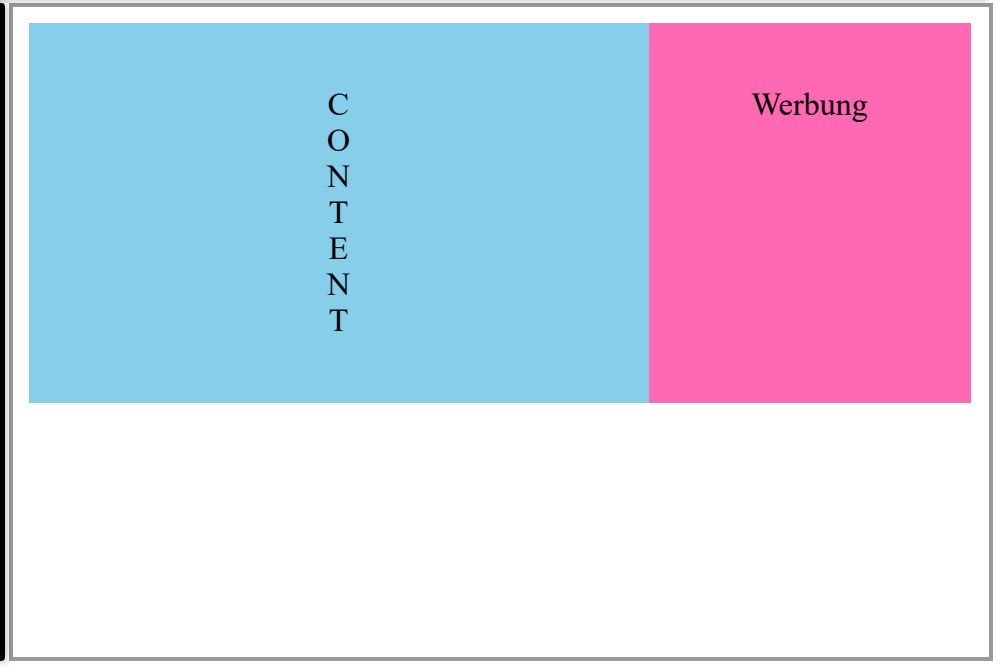
Flexbox – gleich große Boxen

HTML

```html
<main>
   <section>C<br>O<br>N<br>T<br>E<br>N<br>T</section>
   <aside>Werbung</aside>
</main>
```

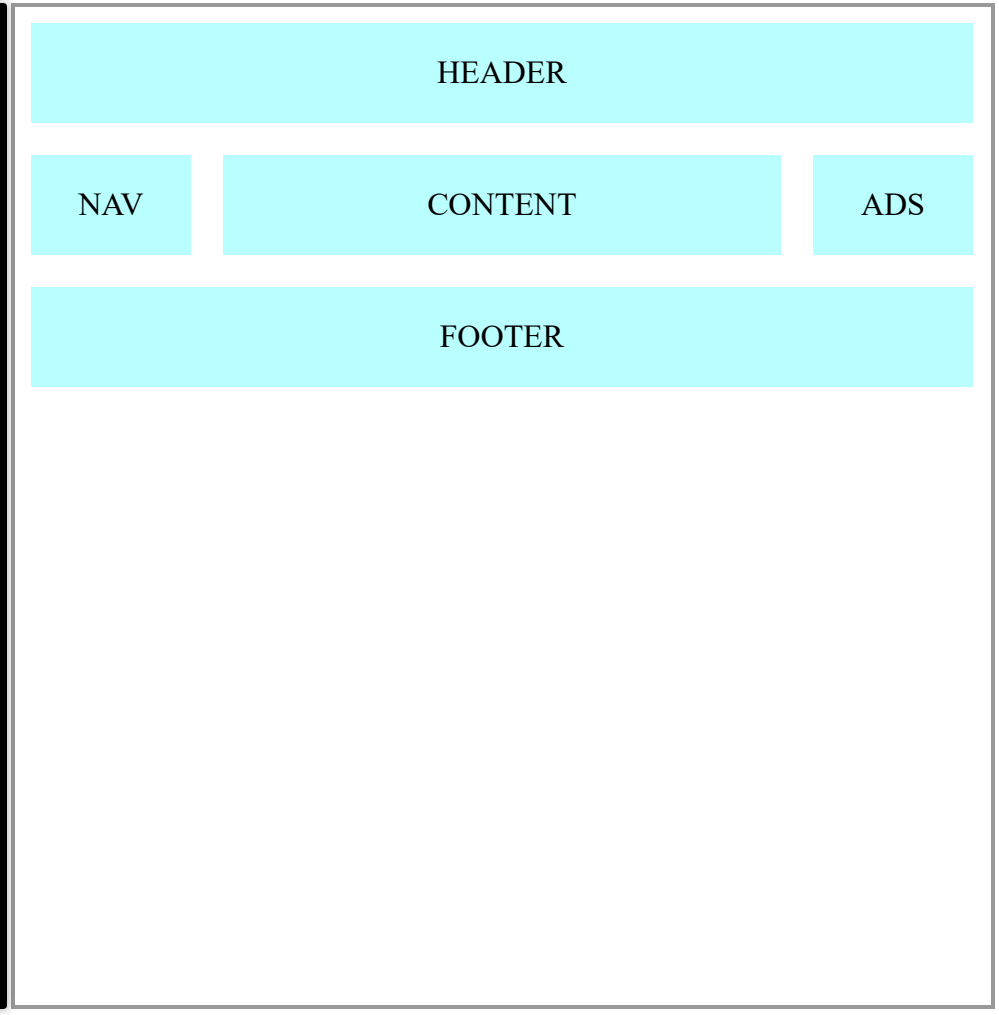
CSS

```css
main { display: flex; }

section { flex-grow: 1;
          background: skyblue; }

aside { width: 20vw;
        flex-shrink: 0;
        background: hotpink; }

main > * { padding: 2em;
           text-align: center; }
```



Flexbox – Holy Grail Layout

```css
main { display: flex;
       flex-direction: row;
       padding: 1em 0;
}

section { flex: 1;
          margin: 0 1em;
}

nav, aside { flex: 0 0 3em; }

header, footer, nav, section,
aside {
   background-color: #BAFFFF;
   padding: 1em;
   text-align: center;
}
```



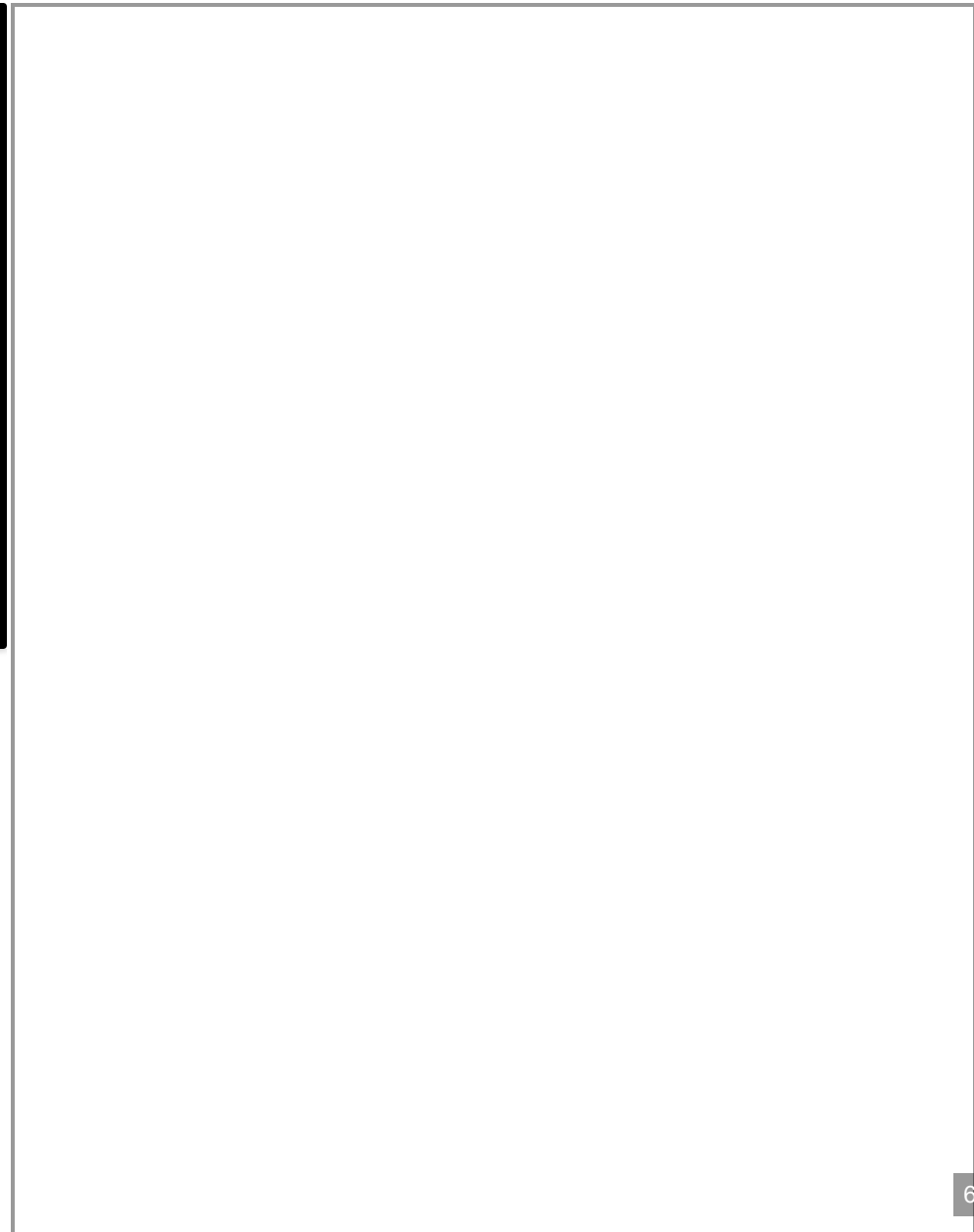
Spalten

```
<!-- Markup -->
<h2>...</h2>
<p>...</p> <!-- nur ein p-Element -->
```

```CSS
p {
  column-count: 2;
  column-gap: 2em;
  column-rule: solid 1px #aaa;
}
```

Einheiten für Breiten und Höhen

```css
body, div {
   box-sizing: border-box;
   width: 80%;
   margin: 8px;
   padding: 8px;
   border: 3px solid hotpink;
}
div { background: skyblue; }
.em { width:  8em; }
.px { width: 80px; }
.vh { width: 80vw; }
```



CSS kann noch mehr

Pseudo-Klassen, -Elemente und Funktionen

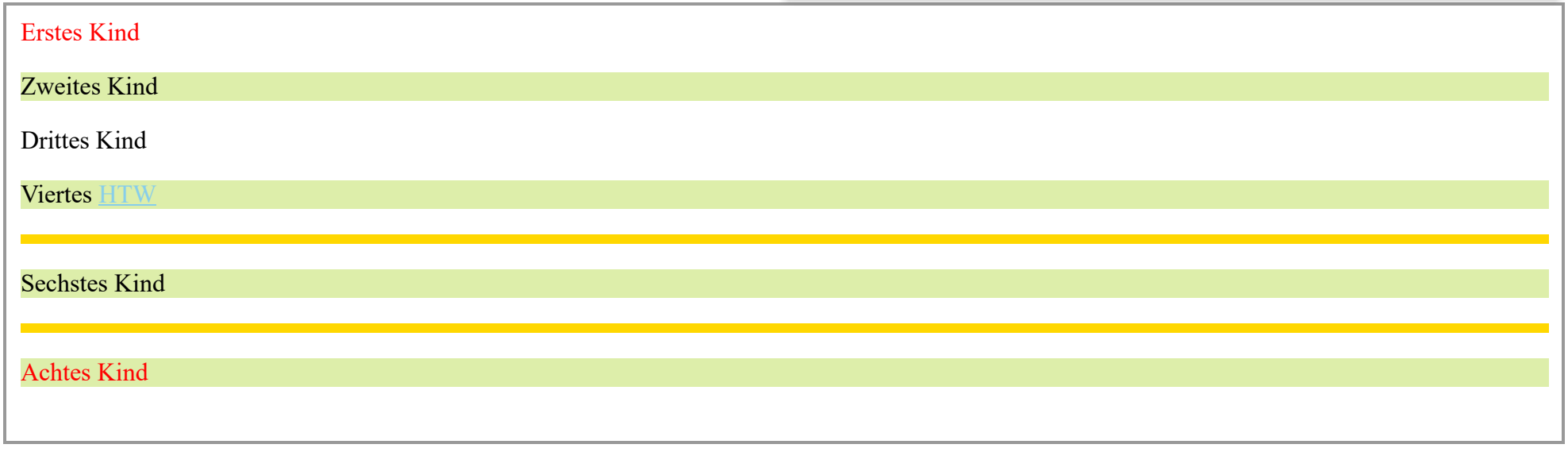
Selektoren: Pseudo-Klassen

HTML

```
<p>Erstes Kind</p>
<p>Zweites Kind</p>
<p>Drittes Kind</p>
<p>Viertes
  <a href="htw-berlin.de">HTW</a>
</p>
<p></p> <p>Sechstes Kind</p>
<p></p> <p>Achtes Kind</p>
```

CSS

```
a         { color: skyblue; }
a:visited { color: darkorchid; }
a:hover   { color: hotpink; }

p:first-child, p:last-child {
    color: red; }
p:empty { border: solid gold; }
p:nth-child(2n) {
    background-color: #DEA; }
```



Selektoren: Pseudoelemente

```html
<p>Lirum larum <a data-count="17" href="#">Posteingang</a></p>
```
HTML

```css
p::first-letter { color: red; }
a { text-decoration: none; }
[data-count]::after {
   content: attr(data-count);
   margin-left: 0.1em;
   padding: 0.1em 0.3em;
   background-color: #333;
   color: white;
   border-radius: 0.65em;
}
```
CSS

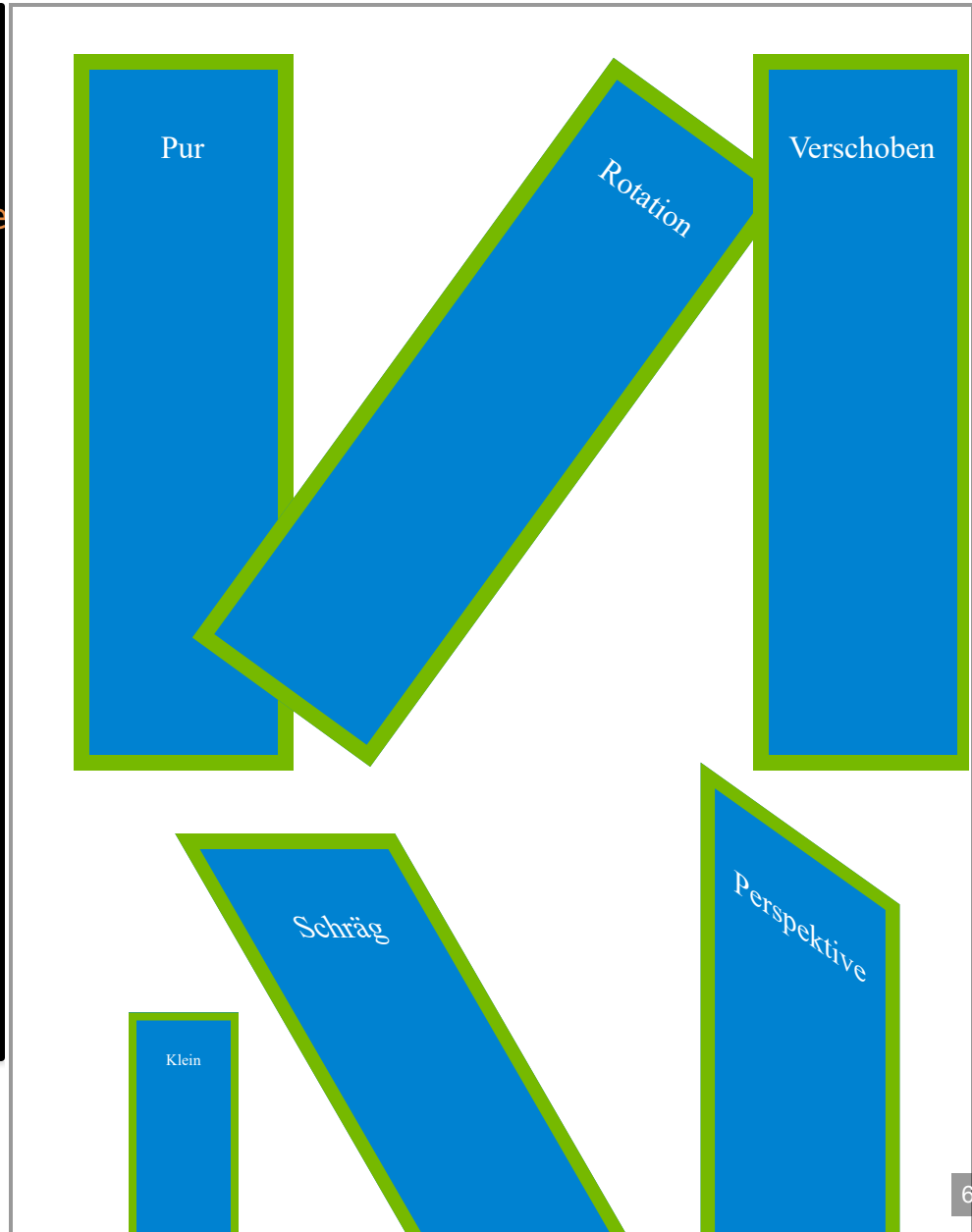
- Mehr dazu:
  - http://wiki.selfhtml.org/wiki/CSS/Selektoren/Pseudoelement

Funktionen

- CSS kennt ein paar Funktionen
  - Farben: `rgb()`, `rgba()`, `hsl()`, `hsla()`
  - Wert eines DOM-Element-Attributs: `content: attr(data-count)`
  - Berechnungen: `width: calc(50% - 2 * 5px)`
  - Transformationen: `rotate()`, `scale()`, `translateX()`, …

Transformationen

```CSS
.a { transform: rotate(0.1turn); }
.b { transform: translateX(2em); }
.c { transform: scale(0.5); }
.d { transform: skewX(30deg); }
.e { transform: perspective( 200px ) rotate

body {
   display: flex;
   justify-content: space-around;
   flex-wrap: wrap;
}

div {
   float: left;
   width: 5em;
   margin: 1em;
   padding: .5em;
   border: .5em solid #76b900;
   background-color: #0082d1;
   color: white;
   line-height: 4em;
   text-align: center;
}
```



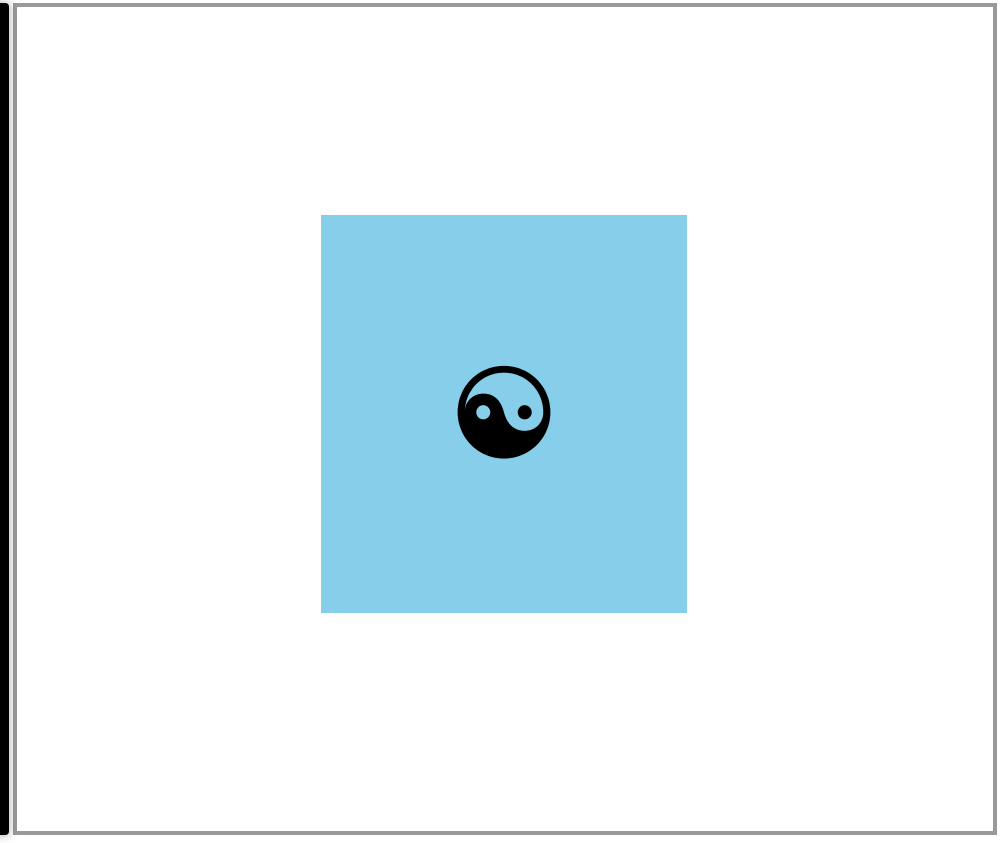
`@media` / media queries

- **Responsiveness**: Bedingte Wirkung von CSS-Regeln
  - für Anzeigemedien (`print`, …)
  - für Geräte-Abmessungen (`min-width`, …)
  - für Ausrichtungen (`orientation`, …)

```
body { font-size: 19px }

@media (min-width: 700px) and (orientation: landscape) { /* ... */ }

@media print {
    * { background: transparent !important; color: #000 !important; }
    body { font-size: 12pt }
    a[href]:after { content: " (" attr(href) ")"; }
    h2, h3 { page-break-after: avoid; }
}
```

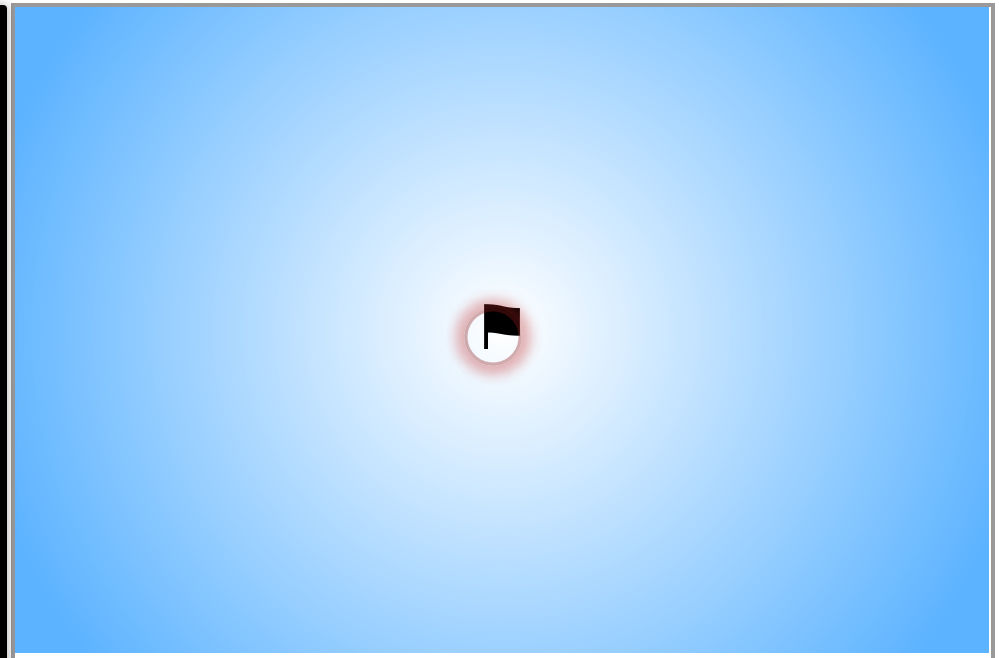
Transitions

```css
span {
   padding: 1em;
   transition: all 1s ease;
   background-color: skyblue;
   border-radius: 0; }

span:hover {
   border-radius: 50%;
   background-color: hotpink; }

body {
   font-size: 400%;
   text-align: center;
}
```



Keyframe Animations

```css
@keyframes pulsing {
  0% { transform: scale(0);
       opacity: 0.5; }

  100% { transform: scale(3);
         opacity: 0; }
}

.pulse { position: relative;
         display: inline-block; }

.pulse span {
  animation: pulsing 2s infinite
    ease-out;

  z-index: 1;
  position: absolute;
  top: 12px;
  left: -10px;
  display: block;
  width: 30px;
  height: 30px;
  border: 2px solid #8f1b1b;
  border-radius: 50%;
  box-shadow: 0 0 10px 5px #ba2323;

  -webkit-animation:
```

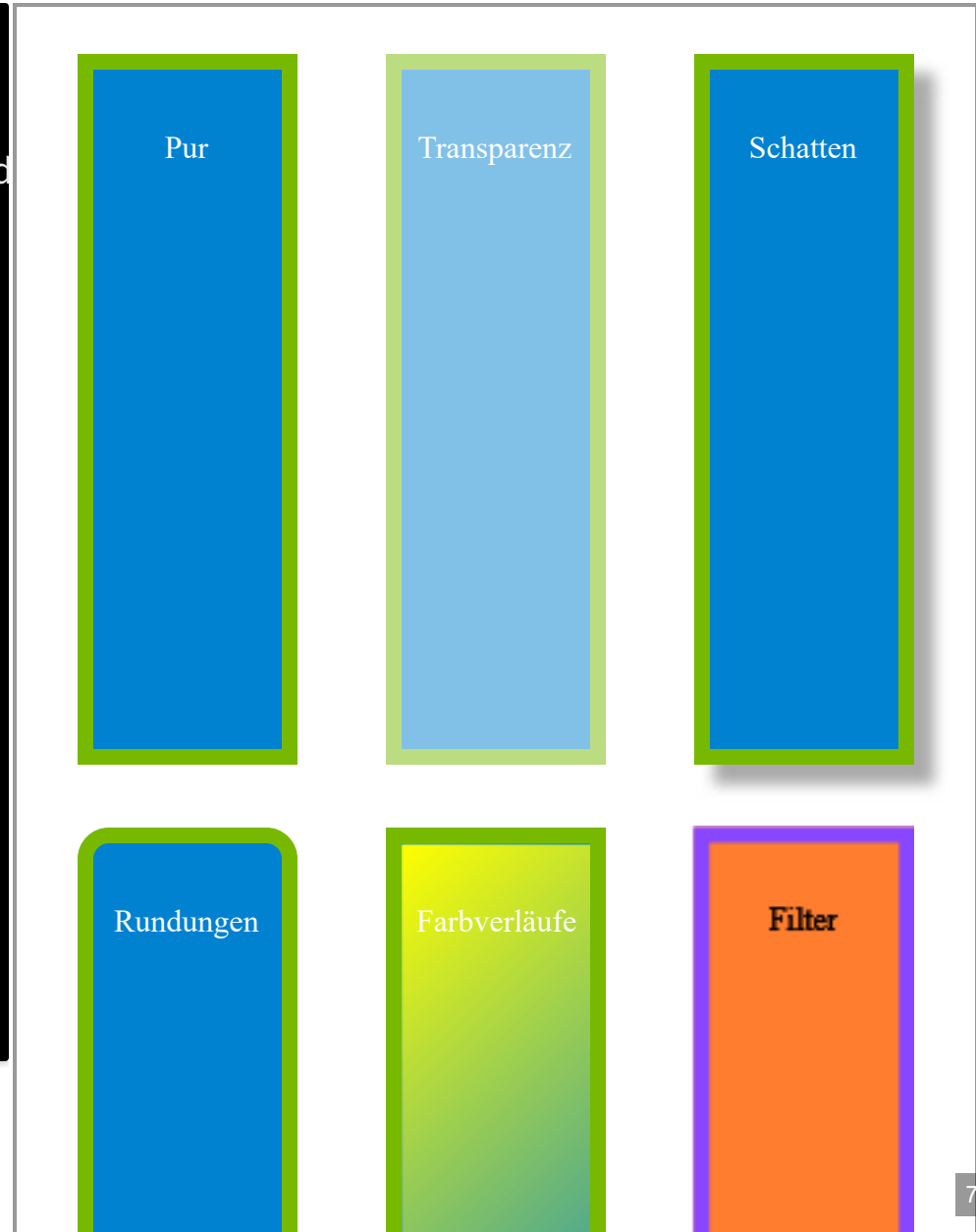


Keyframes

- Syntax:
  - `@keyframes myanimation { from { ... } to { ... } }`
    - oder `@keyframes myanimation { x% { ... } y% { ... } }`
  - Property: `animation`
    - mit Name (`myanimation`), `duration`, `delay`, `direction` (normal, reverse, alternate), `animation-iteration-count` (1, infinite), `animation-timing-function` (ease, linear etc)
- Mehr unter Stylie – A fun CSS animation tool

Grafik-Effekte

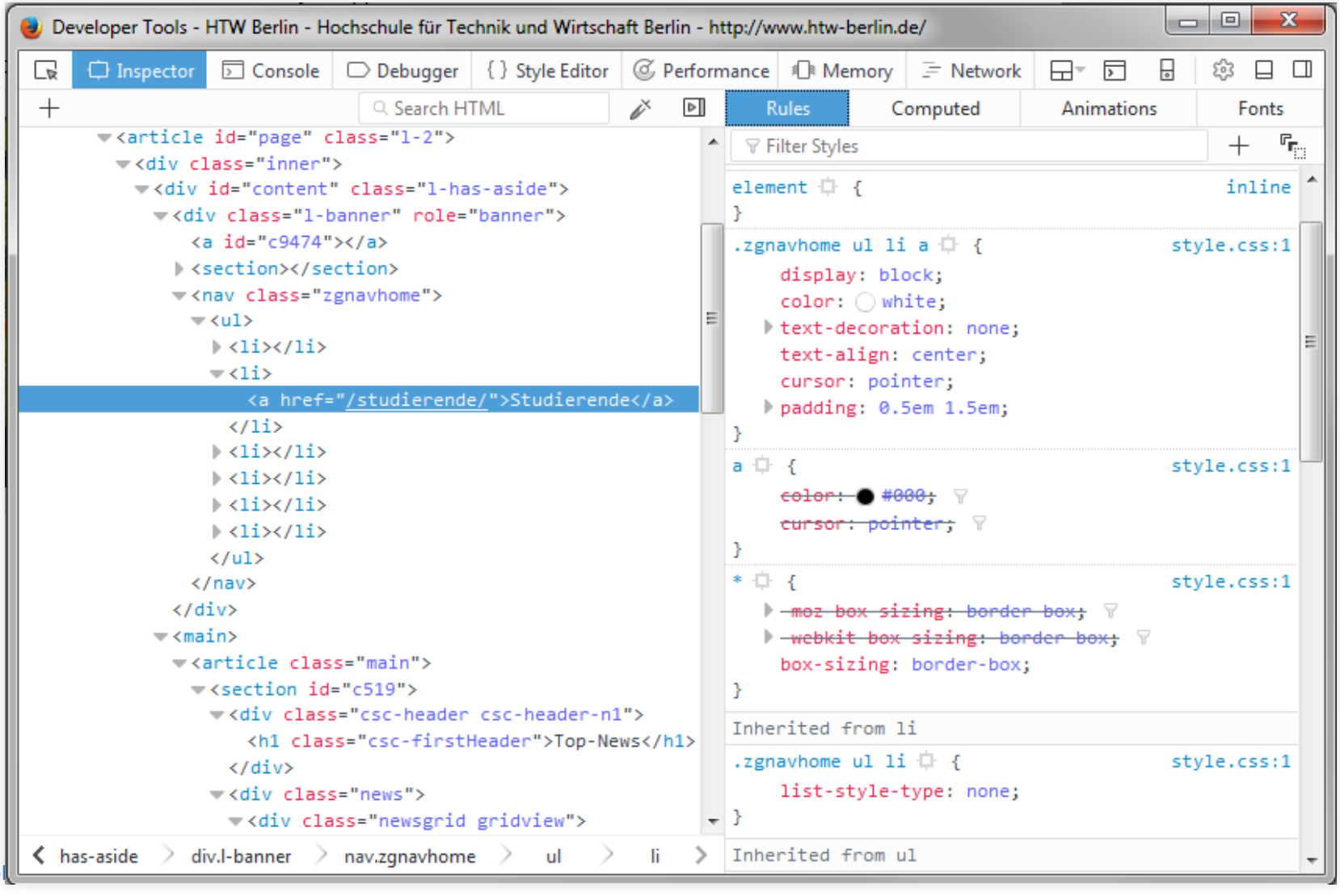
```CSS
.a { opacity: 0.5; }
.b { box-shadow: 10px 10px 10px #aaa; }
.c { border-radius: 1em; }
.d { background-image: linear-gradient(-45d
.e { -webkit-filter: invert(1); }

body {
   display: flex;
   justify-content: space-around;
   flex-wrap: wrap;
}

div {
   float: left;
   width: 5em;
   margin: 1em;
   padding: .5em;
   border: .5em solid #76b900;
   background-color: #0082d1;
   color: white;
   line-height: 4em;
   text-align: center;
}
```



Debugging

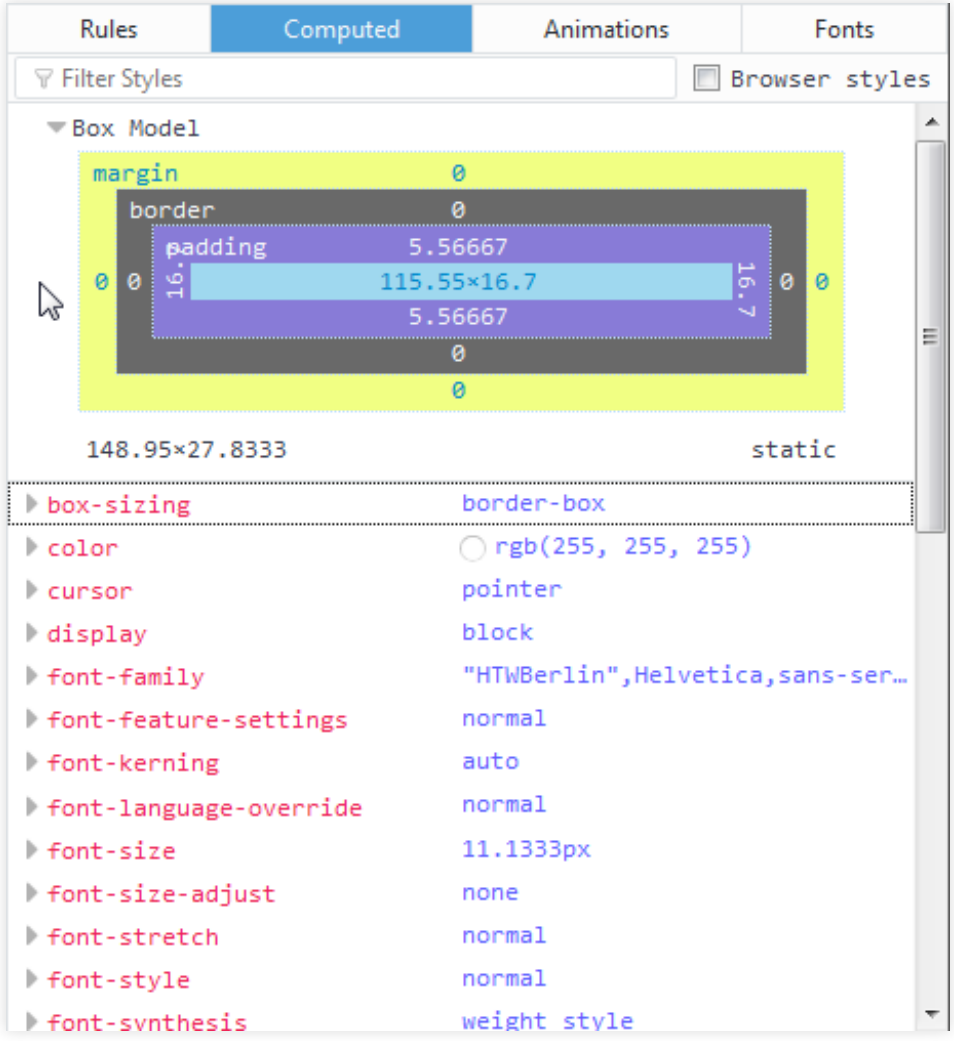
Debugging-Hinweise

- Moderne Browser: Entwicklertools via `F12`



Debugging-Hinweise

- Moderne Browser: Entwicklertools via `F12`



Link-Auswahl

- devdocs.io
  Schnelle Dokumentation für HTML, CSS, JavaScript und viele mehr
- caniuse.com
  Browser-Unterstützung diverser Features
- Code Guide
  Prägnante Best Practices für die Entwicklung von HTML und CSS
- CSS blend modes and filters playground
- CSS Best Practices
  - (eine Präsentation, Navigation mit Pfeiltasten)

Zusammenfassung: Heutige Inhalte

- Grundlegende CSS-Syntax und Spracheigenschaften
- Kaskadierung, Einbindungsmöglichkeiten
- Selektoren
- Box-Modell
- grundlegende CSS-Eigenschaften

Danke!