



# Überblick Heute

---

- Wiederholung und weitere Selektoren
- Box Model und Positionierung
- Externe CSS-Definitionen
- Das Box-Model
- Floating & Overflow
- Entities, Emojis & Cursors

# Selektoren / Kombinatoren

---

## Typselektor

`h1`

## Klassenselektor

`.gelbeBox`

## Kindselektor

`a > img`

## ID-Selektor

`#formSearch`

## Attributsselektor

`input[type="submit"]`

## Nachfahrenselektor

`p a`

# Noch mehr Selektoren?

---

insgesamt fast 50 Möglichkeiten  
HTML-Elemente anzusprechen

- ✓ **Einfache Selektoren**  
HTML-Typ, Klassen, IDs, Attribute
- ✓ **Kombinatoren**  
Kinder, Nachfahren, Nachbarn, Geschwister
- **Pseudoelemente**
- **Pseudoklassen**

# Pseudoelemente

---

einfache Selektoren auf HTML-Elemente beschränkt.

→ Pseudoelemente beziehen sich auf Elemente, die nicht (explizit) im Code vorhanden sind.

Pseudoelement	Beispiel	Verwendung
::before	p::before	Bereich vor dem Element
::after	p::after	Bereich nach dem Element
::firstLine	p::firstLine	Erste Textzeile des Elements
::firstLetter	p::firstLetter	Erster Buchstabe des Elements

# Strukturelle Pseudoklassen

---

sprechen Elemente an, wenn diese eine bestimmte Eigenschaft (in ihrer inhaltlichen Struktur) besitzen

Pseudoelement	Beispiel	Verwendung
:empty	span:empty	Span-Elemente ohne Inhalt
:first-child	tr:first-child	Erste Zeile einer Tabelle
:last-child	body:last-child	Letzte Kindelement des Body
	tr:nth-child(3)	Dritte Zeile einer Tabelle
:nth-child(x)	:nth-child(odd)	Alle Kinder mit ungeradem Index
	:nth-child(even)	Alle Kinder mit geradem Index
:first-of-type	li:first-of-type	Erstes Element des Typs

es gibt noch weitere strukturelle Pseudoklassen!

# Dynam. Pseudoklassen 1/3

---

steuern Elemente an, deren Eigenschaften sich durch Interaktion mit dem Benutzer ändern können

Pseudoelement	Beispiel	Verwendung
gültig für: <a> , <area>		
:link	a:link	Noch nicht besuchte Hyperlinks
:visited	a:visited	Besuchte Hyperlinks
:any-link	a:any-link	Besuchte und unbesuchte Links

# Dynam. Pseudoklassen 2/3

---

steuern Elemente an, deren Eigenschaften sich durch Interaktion mit dem Benutzer ändern können

Pseudoelement	Beispiel	Verwendung
gültig für: <input>		
:disabled	input:disabled	deaktivierte Eingabefelder
:checked	input:checked	Checkbox mit Haken
:valid	input:valid	Gültige Eingaben
:invalid	input:invalid	Ungültige Eingaben

Beispiel für valid/invalid:

```
<input id="input2" required type="number">
```



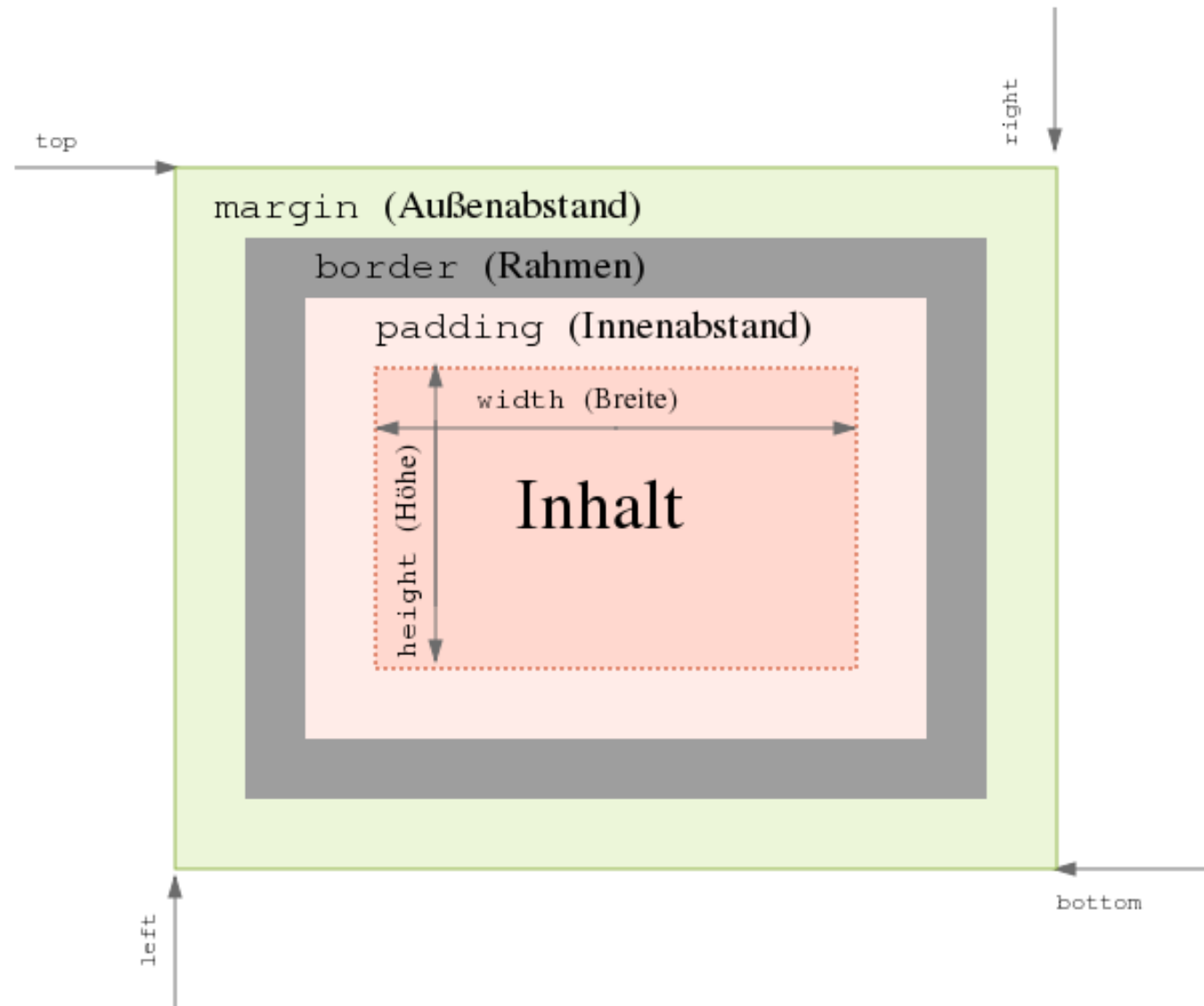
# Dynam. Pseudoklassen 3/3

---

steuern Elemente an, deren Eigenschaften sich durch Interaktion mit dem Benutzer ändern können

Pseudoelement	Beispiel	Verwendung
Maus- & Tastaturinteraktion		
:hover	div:hover	Maus überhalb des Elements
:active	a:active	Aktuell angeklicktes Element
:focus	input:focus	Element hat aktuell Fokus z.B. durch TAB-Taste

# Box-Model



# Box-Model

---

```
.gelbeBox {  
    margin: 1.5em;  
}
```

```
.gelbeBox {  
    margin-top: 1.5em;  
    margin-bottom: 0.2em;  
    margin-left: 0.8em;  
    margin-right: 0;  
}
```

```
.gelbeBox {  
    margin: 1.5em 0 0.2em 0.8em;  
}
```

# CSS-Anweisungen

---

## Inline Styles

- direkt am betroffenen Element
- nicht wiederverwendbar
- schlecht wartbar
- hoher Speicherbedarf

## Interne Styles

- in der betroffenen HTML-Datei
- bedingt wiederverwendbar
- schlecht wartbar
- mittlerer Speicherbedarf

# Externe CSS-Dateien

---

Best Practice:

**Möglichst alle Style-Angaben in externe Datei auslagern.**

Vorteile:

- Wiederverwendbarkeit
- zentrale Konfiguration → Code schlanker
- weitere CSS-Selektoren

Nachteil:

- Überblick behalten (sinnvolle Klassennamen!)
- Zusätzliche HTTP-Request notwendig

# Externe CSS-Dateien

---

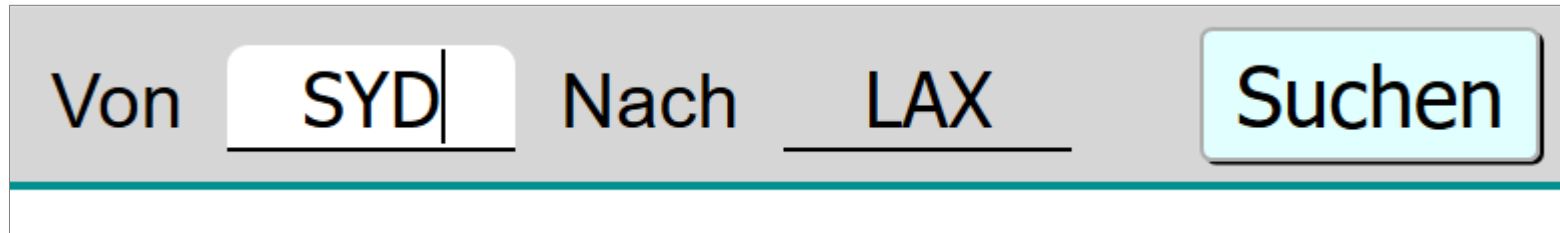
Einbindung erfolgt im Head-Bereich:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Seitentitel</title>
    <link rel="stylesheet" href="pfad/zum/stylesheet.css">
    <link rel="stylesheet" href="pfad/zum/zweitenStylesheet.css">
  </head>
```

<code>&lt;link&gt;</code>	Verbindung zu anderer Datei
<code>rel</code>	Beziehungstyp („ <i>relationship</i> “)
<code>href</code>	Pfad zur referenzierten Datei

`<link>` ist ein empty Tag!

# Übung: Erweitertes Styling s. 1/2



Von  Nach

## Formular:

Bündig mit Browserfenster, Innenabstand 5px

## Eingabefelder:

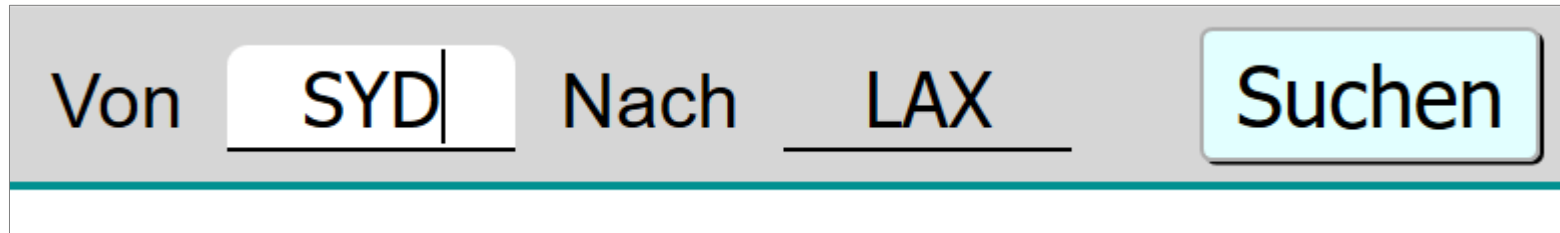
Rahmen nur unten; abgerundete Ecken oben; transparenter Hintergrund; bei Mausberührung und Eingabe: weißer Hintergrund

## Button:

Grauer, abgerundeter Rahmen; Höhe: 1.6\*Schriftgröße; Schriftgröße: 1.3\*Body-Schriftgröße; Schlagschatten (*box-shadow*); Hintergrund: #E0FFFF; bei Mausberührung: #E0EFEF; bei Click: #D0DFDF

# Übung: Erweitertes Styling s. 2/2

---



Von  Nach

- Abgesehen vom Formular ist die Seite vorerst ohne Inhalt
- Alle Styling-Angaben in externer CSS-Datei
- Verwenden Sie verschiedene Selektoren / Kombinatoren
- Häufige Eigenschaften in eigenen Klassen kapseln,  
z.B. **.text-center**



# div & span

---

Ich bin ein <div>

Ich bin ein <div> im <div>

Und ich ein <span> im <div>

## **div („division“)**

- Block-Element, d.h. nimmt vollständige Breite des Parentelements ein.
- ähnlich zu Paragraph <p>  
aber: Lässt sich verschachteln!
- Vermutlich mit am häufigsten verwendetes Tag

# div & span

---

Ich bin ein <div>

Ich bin ein <div> im <div>

Und ich ein <span> im <div>

## span

- Inline-Element, d.h. nimmt nur die beanspruchte Breite ein
- lässt sich verschachteln
- Streng genommen: Darf nur innerhalb von Block-Elementen (z.B. div, p, h1) verwendet werden

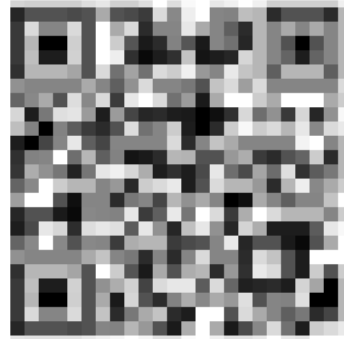
# Experiment: CSS-Quiz

---

3 Minuten, 5 Fragen

Bitte melden Sie sich an:

> Scannen Sie den QR-Code oder rufen Sie die angegebene Website auf:



<http://www.swt.hs-mannheim.de/LA05>

> Wählen Sie **Kurs beitreten** und nutzen Sie folgendes Codewort:

**LA05**

> Klicken sie auf **Teilnehmen** und warten Sie auf den Beginn der Umfrage

Einheit starten

# Positionierung

---

```
position: absolute;
```

- absolute Positionierung in Bezug zum Viewport oder relativ-positioniertem Parent
- unabhängig vom Textfluß
- Nicht *absolut* positionierte Elemente erscheinen dahinter

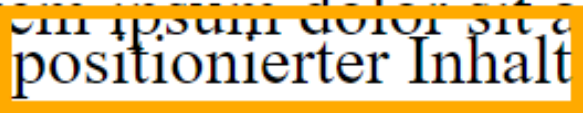
# Positionierung

---

```
position: absolute;
```

```
<style>
  div.positioniert {
    position: absolute;
    left: 30px; top: 110px;
    border: 3px solid orange;
  }
</style>
```

Lorem ipsum dolor sit amit

Lorem ipsum dolor sit amit  
positionierter Inhalt

- einfach und intuitiv
- nicht responsive  
(bzw. kompliziert)
- Textfluß wird ignoriert
- geeignet für Inhalte mit fester Größe geeignet  
(z.B. Header)

# Positionierung

---

```
position: relative;
```

- Positionierung relativ zum davor liegenden Inhalt
- „Rutscht“ mit dem Textfluss
- Kann als Container für absolut positionierte Elemente genutzt werden

# Positionierung

---

`position: relative;`

```
<style>
  div.positioniert {
    position: relative;
    left: 30px; top: 110px;
    border: 3px solid orange;
  }
</style>
```

Lorem ipsum dolor sit amit

Lorem ipsum dolor sit amit

positionierter Inhalt

# Positionierung

---

```
position: fixed;
```

- Bezug immer zum Viewport
- Verbleibt in der Position, auch bei scrollen
- z.B. gut geeignet „mit-fliegende“ Leisten oder Footer-Leisten



# Positionierung

---

```
position: static;           /* DEFAULT */
position: absolute;
position: relative;
position: fixed;
position: sticky;
```

- Positionierung je nach Anwendung auswählen
- Immer an unterschiedl. Endgeräte denken
- häufig einzelne Elemente besonders positioniert, die meisten aber Teil des Textflusses

# Übung: Positionierung

---

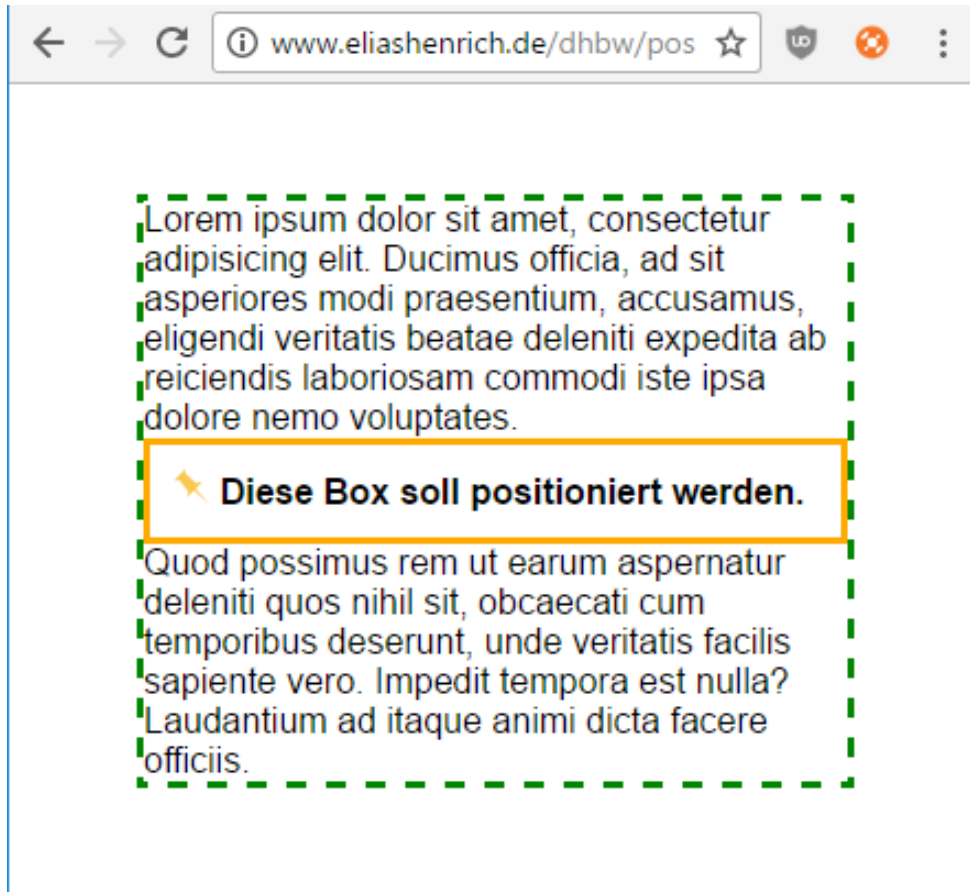
Öffnen Sie in Ihrem Webbrowser

**<http://eliashenrich.de/dhbw/positionierung.html>**

und kopieren Sie den Quellcode.

- 1) Wie verhält sich die .orangeBox in Bezug auf Ihre Position im Text?
- 2) Versuchen Sie folgende Beispiele anhand des Codes nachzuvollziehen.

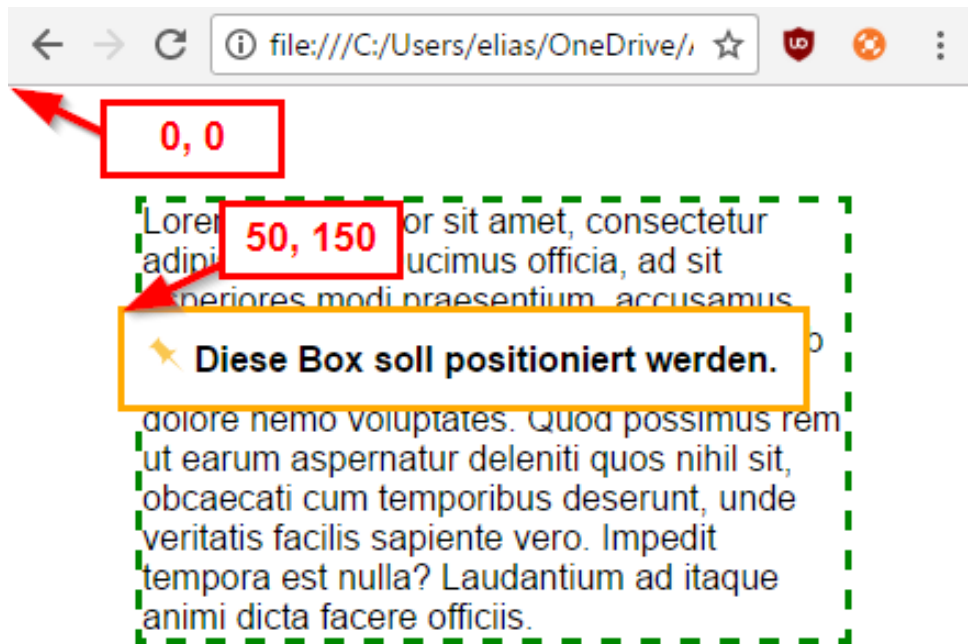
# Übung: Positionierung



## position: static

- Box ist im Textfluss  
d.h. sie taucht an der Stelle auf, an der sie im Text steht
- ggf. Positionierung durch Box-Model
- Default-Einstellung  
→ muss nicht explizit angegeben werden

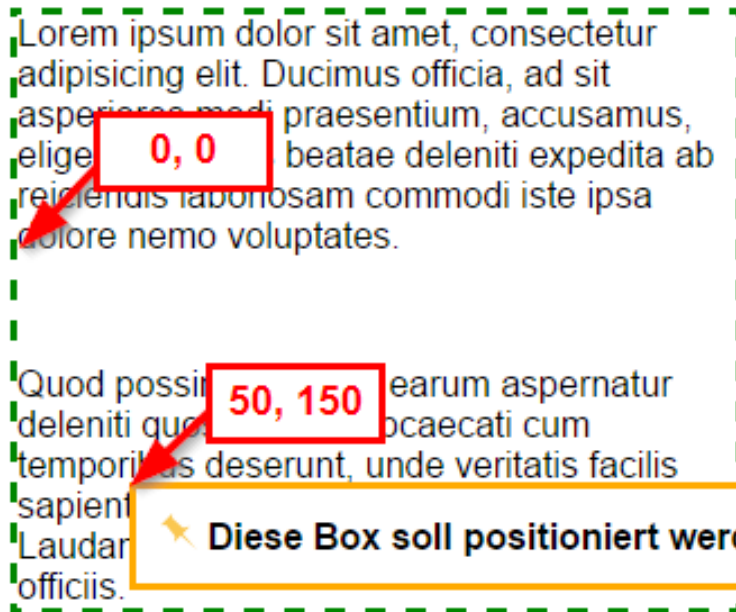
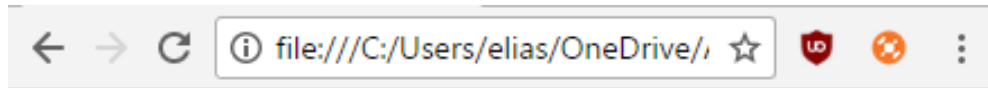
# Übung: Positionierung



**position: absolute**

- Box vom Text losgelöst
- Bezug auf Viewport oder relativ-positionier. Parent-Element
- Positionierung durch
  - Top / Bottom
  - Left / Right

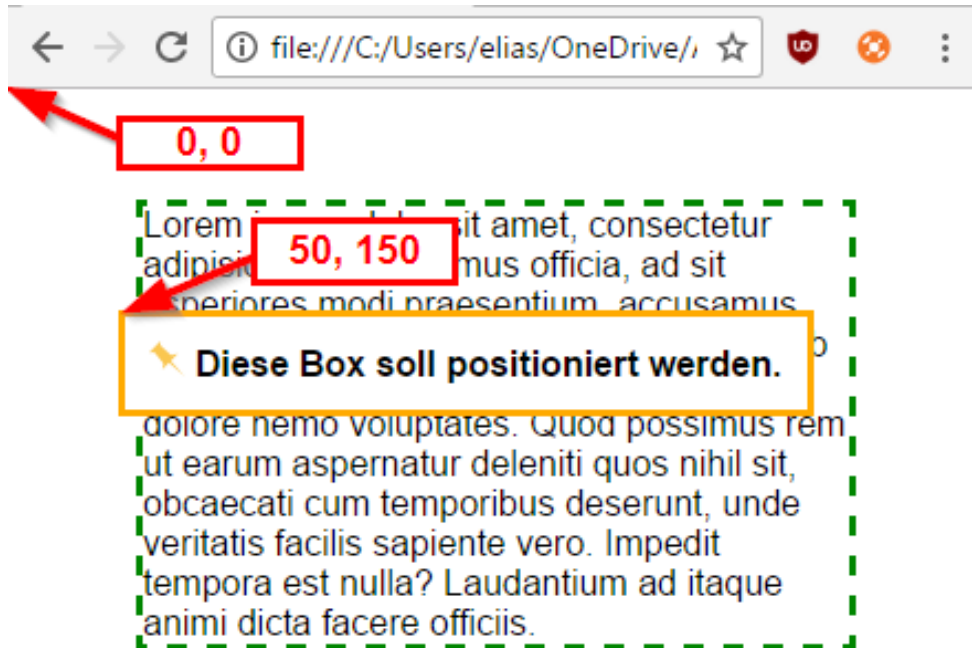
# Übung: Positionierung



**position: relative**

- ähnlich zu absolute
- Bezug auf Textfluss
- Positionierung durch
  - Top / Bottom
  - Left / Right

# Übung: Positionierung

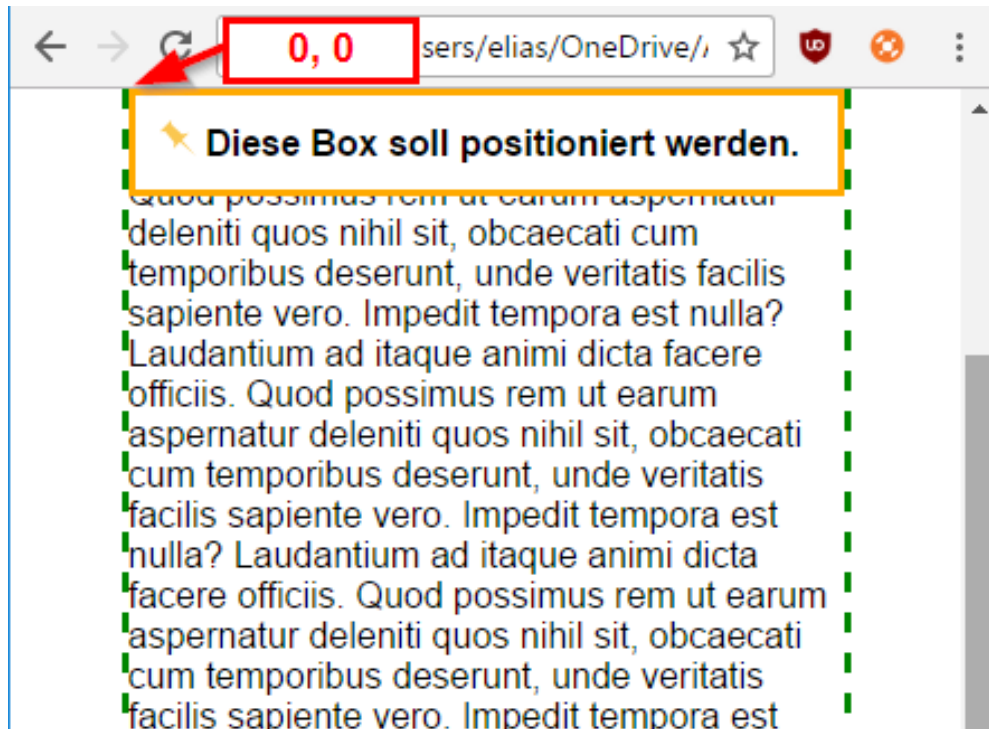


**position: fixed**

- ähnlich zu absolute
- Bezug immer auf Viewport
- unabhängig von Scrollen
- Positionierung durch
  - Top / Bottom
  - Left / Right

Verlängern Sie den Text, um zu sehen, wie sich die Box beim Scrollen verhält.

# Übung: Positionierung



Verlängern Sie den Text, um zu sehen, wie sich die Box beim Scrollen verhält.

## position: sticky

- ähnlich zu fixed
- Bezug erst auf Textfluß, dann auf Viewport
- Positionierung durch
  - Top / Bottom
  - Left / Right

# float

verschiebt ein Element an die Innenkante  
seines Elternelements

```
<p>  
    
  Lorem ipsum dolor, sit amet consectetur adipisicing elit Laborum.  
  <br>  
  Lorem ipsum dolor, sit amet consectetur adipisicing elit [...]  
</p>
```



Lorem ipsum dolor, sit amet  
consectetur adipisicing elit  
Laborum.

Lorem ipsum dolor, sit amet  
consectetur adipisicing elit. Laborum, fugit  
eius harum debitis doloremque necessitatibus  
animi officiis, molestias amet architecto  
perferendis placeat ex aperiam consequatur  
accusamus beatae explicabo sed facilis.



# float - clear

---

stellt nach einem float-Element den normalen Textfluss wieder her

```
<p>  
    
  Lorem ipsum dolor, sit amet consectetur adipisicing elit Laborum.  
  <br style="clear: both;">  
  Lorem ipsum dolor, sit amet consectetur adipisicing elit [...]  
</p>
```



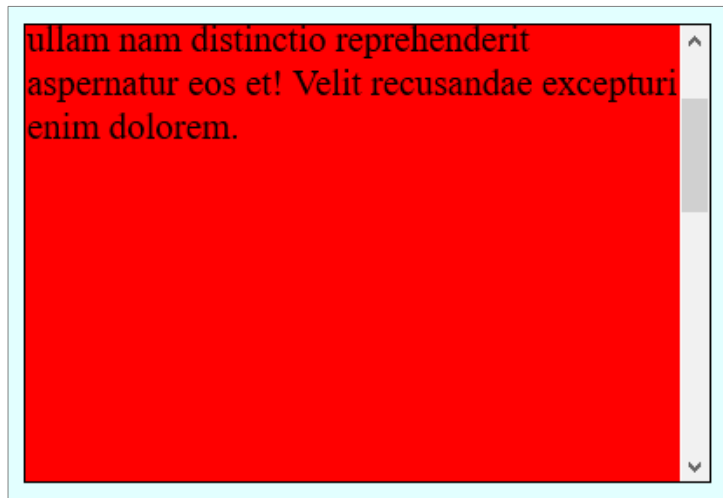
Lorem ipsum dolor, sit amet  
consectetur adipisicing elit  
Laborum.

Lorem ipsum dolor, sit amet consectetur  
adipisicing elit. Laborum, fugit eius harum  
debitis doloremque necessitatibus animi  
officiis, molestias amet architecto perferendis  
placeat ex aperiam consequatur accusamus  
beatae explicabo sed facilis.

# Overflow

ermöglicht Inhalte, die größer als ihr Container sind

```
<div style="height: 200px; overflow-y: auto;">
  <div style="height: 600px; background-color: red;">
    [...]
  </div>
</div>
```



overflow	→ horizontal & vertikal
overflow-x	→ horizontal
overflow-y	→ vertikal

auto	→ standard, wenn notwendig
Scroll	→ immer zeigen
Visible	→ Inhalt überlappt Parent
Hidden	→ Scrollen immer ausblenden

# Übung: Seitenstruktur

---

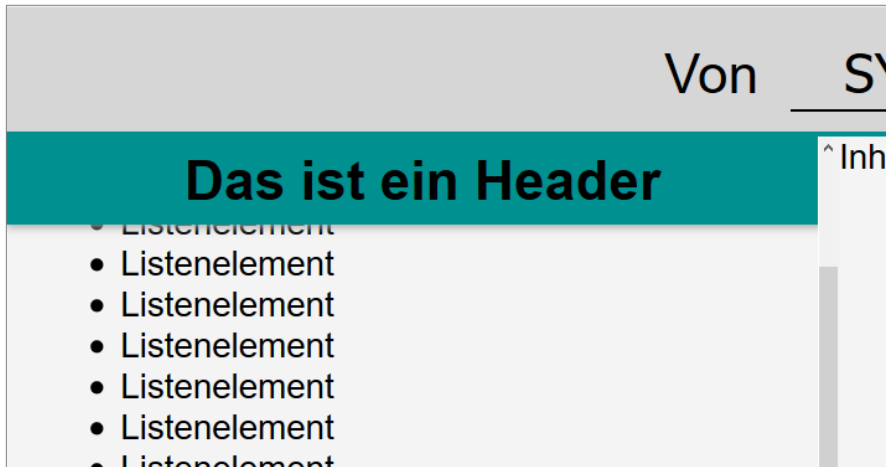
The diagram illustrates a web page layout. At the top, there is a search bar with the text "Von SYD Nach LAX" and a "Suchen" button. Below the search bar, the page is divided into two main content areas: "Inhalt links" (left content) and "Inhalt rechts" (right content). The "Inhalt links" area is colored red and occupies 35% of the width. The "Inhalt rechts" area is colored blue and occupies 65% of the width. The entire content area is filled, demonstrating a single-page application design without visible scrollbars.

- Inhaltsbereich füllt Fenster unterhalb des Formulars aus
- Links: Breite 35%
- Rechts: Breite 65%
- „Single-Page-Application“ - Es sollen keine Scrollleisten sichtbar sein
- Farben dienen nur der Verdeutlichung

Es existieren unzählige Möglichkeiten zur Realisierung

# Übung: Seitenstruktur

---



- Unsortierte Liste mit vielen Test-Einträgen
- Liste lässt sich nach unten scrollen

Seitenleiste um Heading erweitern:

- HTML-Typ h1
- Hintergrund: darkcyan
- Schatten: 0 1px 3px #9F9F9F
- behält Position beim Scrollen  
→ Liste hinter Heading
- Schriftgröße 1.2\*Body-Schriftgr.

# HTML Entities & Sonderzeichen

In Auszeichnungssprachen wie HTML sind Entities eine Möglichkeit, bestimmte Zeichen mit sprechenden Abkürzungen zu verbinden.

Quelle: <https://wiki.selfhtml.org/wiki/Entity>

---

<	>	&lt;	&gt;
---	---	------	------

---

©	&copy;
---	--------

---

$\frac{3}{4}$	&frac34;
---------------	----------

---

∞	&infin;
---	---------

---

---

Dingbats

	&#9988;
---	---------

---

Emoticons

	&#x1F354
---	----------

---

Durch UTF-8 können Sonderzeichen - wie auch Umlaute - direkt im Quellcode eingefügt werden.

# CSS: Cursor

In Auszeichnungssprachen wie HTML sind Entities eine Möglichkeit, bestimmte Zeichen mit sprechenden Abkürzungen zu verbinden.

Quelle: <https://wiki.selfhtml.org/wiki/Entity>

---

<	>	&lt;	&gt;
---	---	------	------

---

©	&copy;
---	--------

---

$\frac{3}{4}$	&frac34;
---------------	----------

---

∞	&infin;
---	---------

---

---

Dingbats


---

	&#9988;
---	---------

---

Emoticons

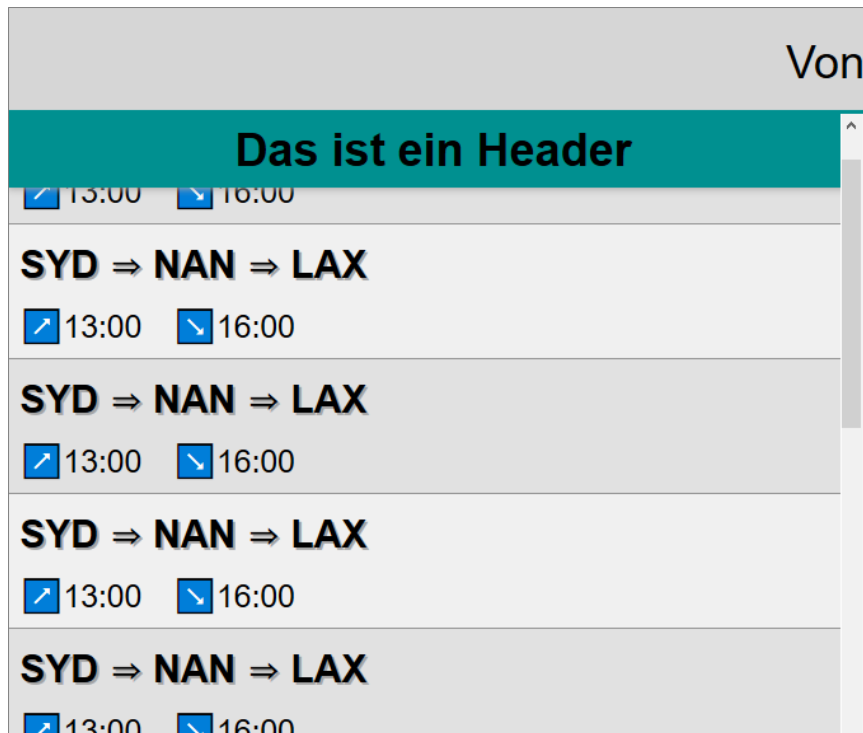
---

	&#x1F354
---	----------

---

Durch UTF-8 können Sonderzeichen - wie auch Umlaute - direkt im Quellcode eingefügt werden.

# Übung: Listenelemente



- `<ul>` und `<li>`-Konstrukt
- Jedes Listenelement ist 4em hoch
- „Finger“-Cursor
- Hintergrund abwechselnd `#DFDFDF` und `#EFEFEF`
- Bei Mausberührung alle: `#FFEFEF`
- Rahmen unten `#909090`
- Flug-Route:
  - 1rem & Fettschrift
  - `text-shadow: 1px 1px 0 #A0A0A0;`
- Flugzeiten:
  - Größe: 1.1em
  - Emojis vor der Zeit nur an einer Stelle definieren (CSS)