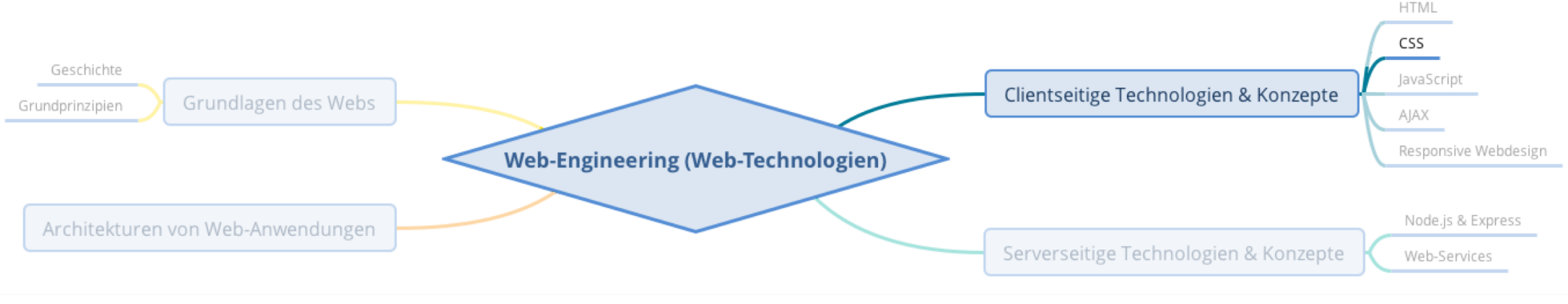
WEB-TECHNOLOGIEN

CLIENTSEITIGE TECHNOLOGIEN: CSS

THEMEN DER VERANSTALTUNG



LERNZIEL

Die Grundlagen von CSS verstehen und anwenden können

CASCADING STYLESHEETS (CSS)

- CSS beschreibt die Darstellung von HTML-Elementen
- Damit wird die Struktur bzw. Semantik des Inhalts (HTML) sauber von der Darstellung des Inhalts (CSS) getrennt (*Trennung von Zuständigkeiten*)

TRENNUNG VON STRUKTUR UND DARSTELLUNG

- Sorgt dafür, dass HTML-Dokumente übersichtlich und lesbar bleiben
- Spart Aufwand, da man die Darstellung für mehrere HTML-Dokumente gleichzeitig festlegen und anpassen kann
- Ermöglicht das Anbieten verschiedener Darstellungen für ein und denselben Inhalt (Barrierefreiheit, verschiedenen Geräte, etc.)

Beispiel: CSS Zen Garden

CSS: GESCHICHTE

1994:

Håkon Wium Lie (Mitarbeiter von Tim Berners-Lee am CERN) schlägt eine erste Version von CSS vor (damals noch *"Cascading HTML Style Sheets"*, CHSS)

1995:

Das W3C wird auf den Vorschlag aufmerksam

1996:

W3C veröffentlicht CSS Level 1 (CSS 1)

CSS: GESCHICHTE

1998

- W3C veröffentlicht CSS Level 2 (CSS 2)
- CSS 2 ist eine Obermenge von CSS 1 (und damit abwärtskompatibel)
- CSS 2 wurde von Browsern jedoch nur teilweise oder falsch implementiert

CSS: GESCHICHTE

2000

- W3C beginnt die Arbeit an CSS Level 3 (CSS3)
- Grundidee: Aufteilung von CSS in Module

CSS: GESCHICHTE

2002

- Aufgrund der Probleme mit CSS 2 arbeitet das W3C an CSS Level 2 Revision 1 (CSS 2.1)
- Es dauert jedoch bis...

CSS: GESCHICHTE

2011

- ...bis das W3C CSS 2.1 veröffentlicht
- Die Entwicklung an CSS3 läuft bis heute

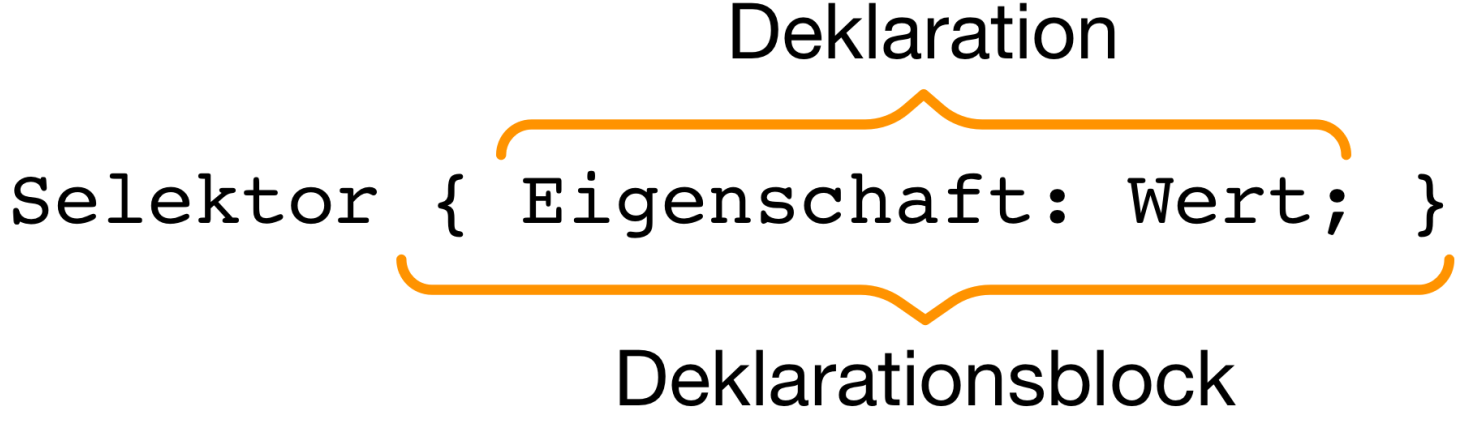
CSS3, ODER EINFACH NUR CSS?

- Ähnlich wie bei HTML (WHATWG) wird CSS selbst nicht mehr versioniert
- CSS besteht aus vielen Modulen (jeweils eigene Versionen und Arbeitsgruppen)
- Die W3C veröffentlicht lediglich ab und zu "Snapshots" mit dem aktuellen Stand

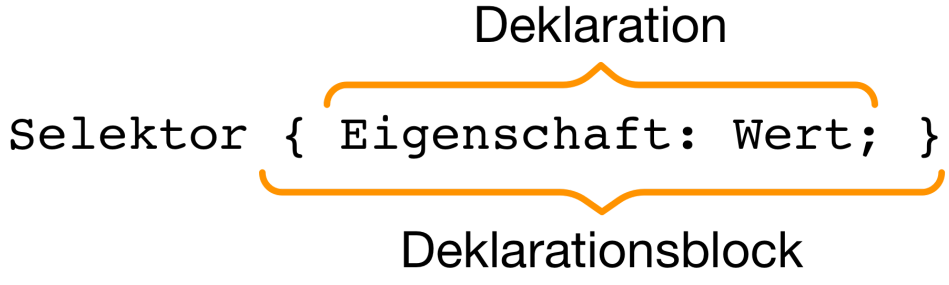
Infoseite zum Entwicklungsstand: → CSS Current Work

CSS-REGELN

- CSS dient der Formulierung von **Regeln**, welche die Darstellung von HTML-Elementen beschreiben
- Ein **Stylesheet** ist eine Sammlung solcher Regeln
- Aufbau einer CSS-Regel:



CSS-REGELN (2)



- Der **Selektor** gibt an, auf welches HTML-Element bzw. welche HTML-Elemente die Regel angewendet werden soll
- Der **Deklarationsblock** enthält eine oder mehrere Deklarationen (durch Semikolons getrennt)
- Eine **Deklaration** besteht aus zwei Teilen:
  1. Die **Eigenschaft**, welche modifiziert werden soll (z.B. Farbe, Schriftart)
  2. Der **Wert** (bzw. mehrere Werte) für die Eigenschaft (z.B. "blue", "Arial")

CSS-REGEL: BEISPIEL*

* (links = Code, rechts = wie im Browser dargestellt)

CSS

```css
/* Alle h1-Elemente selektieren */
h1 {
  /* Hintergrundfarbe auf "blau" setzen */
  background-color: blue;
  /* Textfarbe auf "weiß" setzen */
  color: white;
  /* Text in Großbuchstaben schreiben */
  text-transform: uppercase;
}
```

HTML

```html
<h1>Gestyle Überschrift</h1>
```

GESTYLE ÜBERSCHRIFT

KOMMENTARE

- Kommentare in CSS starten mit `/*` und enden mit `*/`

```
/* Ich bin ein Kommentar. */
```

- Analog HTML: Werden im Browser nicht dargestellt, sind aber dennoch im Quelltext einsehbar

EINBINDUNG VON CSS

1. **Inline-Style**: CSS-Deklarationen direkt am HTML-Element notieren
2. **Internes Stylesheet**: CSS-Regel im `head`-Bereich des HTML-Dokuments notieren
3. **Externes Stylesheet**: CSS-Regeln in externer Datei notieren und diese Datei im HTML-Dokument referenzieren

CSS EINBINDEN: INLINE-STYLE

- CSS-Deklarationen werden mit dem `style`-Attribut direkt am HTML-Element notiert
- Somit wird kein Selektor benötigt

Beispiel:

```
<h1 style="font-style: italic;
           color: orange;">
  Eine Überschrift
</h1>
<p style="font-family: fantasy;">
  Und ein Absatz.
</p>
```

CSS EINBINDEN: INTERNES STYLESHEET

- CSS-Regeln werden innerhalb des `style`-Elements im Kopfbereich (`head`) des HTML-Dokuments notiert
- Regeln gelten für das gesamte HTML-Dokument

INTERNES STYLESHEET: BEISPIEL

Beispiel:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <style>
      h1 {
        font-style: italic;
        color: orange;
      }

      p { font-family: fantasy; }
    </style>
  </head>

  <body>
    <h1>Eine Überschrift</h1>
    <p>Und ein Absatz.</p>
  </body>
</html>
```

CSS EINBINDEN: EXTERNES STYLESHEET

- CSS-Regeln werden in einer eigenen Datei notiert (Dateiendung: `.css`)
- Die Datei wird über das `link`-Element im Kopfbereich (`head`) des HTML-Dokuments eingebunden
- Es können auch mehrere CSS-Dateien auf diese Weise eingebunden werden

EXTERNES STYLESHEET: BEISPIEL

style.css

```
h1 {
  font-style: italic;
  color: orange;
}

p { font-family: fantasy; }
```

seite.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="style.css">
  </head>

  <body>
    <h1>Eine Überschrift</h1>
    <p>Und ein Absatz.</p>
  </body>
</html>
```

CSS EINBINDEN: BEWERTUNG

Inline-Style:

- ⛔ Inline-Styles verschlechtern die Lesbarkeit des HTML-Codes
- ⛔ Bei Verwendung von Inline-Styles wird Struktur (HTML) von Darstellung (CSS) nicht mehr sauber getrennt
- ⛔ Schlecht wartbar: Änderungen an den Styles können aufwändig werden
- ⛔ Keine Wiederverwendung der Styles
- ✅ Manchmal der einzige Weg, bestehende Regeln zu überschreiben (siehe "Kaskade")

Internes Stylesheet:

- ✅ Trennung von Struktur und Darstellung schon besser
- ⛔ Stylesheet kann jedoch nicht in anderen HTML-Dokumenten wiederverwendet werden

CSS EINBINDEN: BEWERTUNG (2)

Externes Stylesheet:

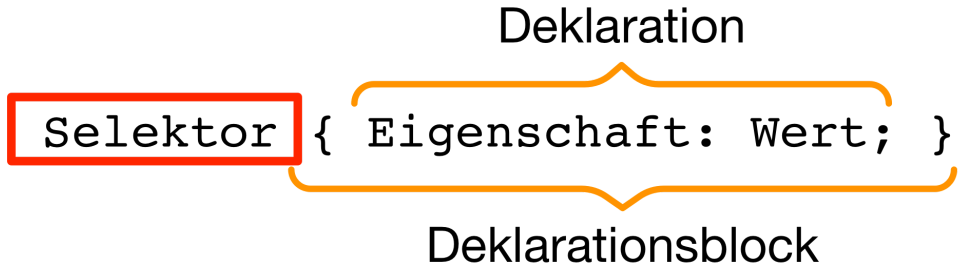
- ➕ Saubere Trennung von Struktur und Darstellung
- ➕ Gute Wartbarkeit und Wiederverwendbarkeit des Stylesheets (Einbinden in mehreren HTML-Dokumenten)
- ➕ Eignet sich z.B. gut zur Realisierung eines einheitlichen Layouts

❗ Empfehlung: Externe Stylesheets bevorzugen und auf Inline-Styles nach Möglichkeit verzichten!

CSS EINBINDEN: KOMBINATIONEN

- Die drei Möglichkeiten zum Einbinden von CSS können auch in Kombination genutzt werden
- Hierbei kann es zu Konflikten zwischen Regeln kommen (Beispiel: `h1` wird zweimal mit unterschiedlicher Farbe formatiert)
- Vorrangsregeln:
  1. Inline-Styles haben immer Vorrang
  2. Bei internen und externen Stylesheets ist die Reihenfolge im HTML-Dokument relevant: Die letzte (d.h. die im HTML-Dokument am weitesten unten stehende) Deklaration gewinnt

SELEKTOREN



- Erinnerung: **Selektoren** geben an, auf welches HTML-Element bzw. welche HTML-Elemente eine CSS-Regel angewendet werden soll (= welches HTML-Element zur Formatierung *selektiert* werden soll)
- CSS bietet verschiedene Arten und Kombinationen von Selektoren
- Die bisherigen Beispiele haben ausschließlich den **Typselektor** verwendet

SELEKTOREN: ÜBERSICHT

- Typselektor
- ID-Selektor
- Klassenselektor
- Universalselektor
- Attributselektor
- Pseudoklassen
- Pseudoelemente

TYPSELEKTOR

- Auch **Elementselektor** genannt
- Selektiert HTML-Elemente direkt anhand ihres Elementnamens

Beispiel:

```
/* Alle h1-Elemente selektieren */
h1 {
  /* Hintergrundfarbe auf "blau" setzen */
  background-color: blue;
}

/* Alle table-Elemente selektieren */
table {
   /* Tabellenrand als Linie mit 1 Pixel Breite darstellen */
   border: 1px solid;
}
```

ID-SELEKTOR

- Selektiert ein HTML-Element anhand seiner ID (Attribut `id`)
- Erinnerung: ID muss dokumentweit eindeutig sein!
- Syntax: `#id`

ID-SELEKTOR: BEISPIEL

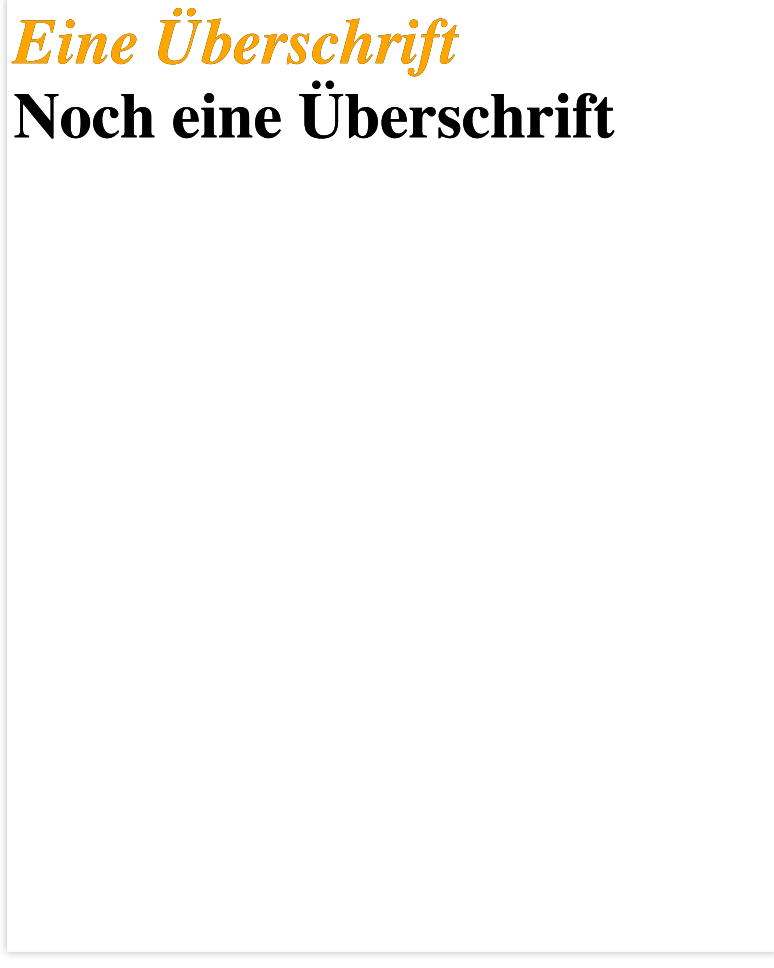
style.css

```css
/* Das Element mit der ID "appName" selektieren */
#appName {
  font-style: italic;
  color: orange;
}
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="style.css">
  </head>

  <body>
    <h1 id="appName">Eine Überschrift</h1>
    <h1>Noch eine Überschrift</h1>
  </body>
</html>
```



KLASSENSELEKTOR

- Selektiert alle HTML-Elemente mit einer bestimmten Klasse (Attribut `class`)
- Syntax: `.klassenname`

KLASSENSELEKTOR: BEISPIEL

style.css

```
/* Elemente mit der Klasse "wichtig" selektieren */
.wichtig {
  border: 1px dotted;
  background-color: yellow;
}
/* Elemente mit der Klasse "gross" selektieren */
.gross { font-size: larger; }
```

seite.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="style.css">
  </head>

  <body>
    <p class="wichtig">Wichtiger Absatz</p>
    <p>Nicht so wichtiger Absatz</p>
    <p class="wichtig gross">
      Wichtiger und großer Absatz
    </p>
    Wichtiges <span class="wichtig">Wort</span>
  </body>
</html>
```

Wichtiger Absatz

Nicht so wichtiger Absatz

Wichtiger und großer Absatz

Wichtiges Wort

ID ODER KLASSE?

Merkregeln:

IDs

- Der Selektor mit der Raute (#)
- Eindeutig: Eine ID darf nur einmal im HTML-Dokument vorkommen, jedes HTML-Element darf nur eine ID haben

Klassen

- Der Selektor mit dem Punkt (`.`)
- Nicht eindeutig: Eine Klasse darf beliebig oft im HTML-Dokument vorkommen, jedes HTML-Element darf beliebig viele Klassen haben (durch Leerzeichen getrennt)

UNIVERSALSELEKTOR

- Selektiert alle HTML-Elemente in einem HTML-Dokument
- Syntax: *

UNIVERSALSELEKTOR: BEISPIEL

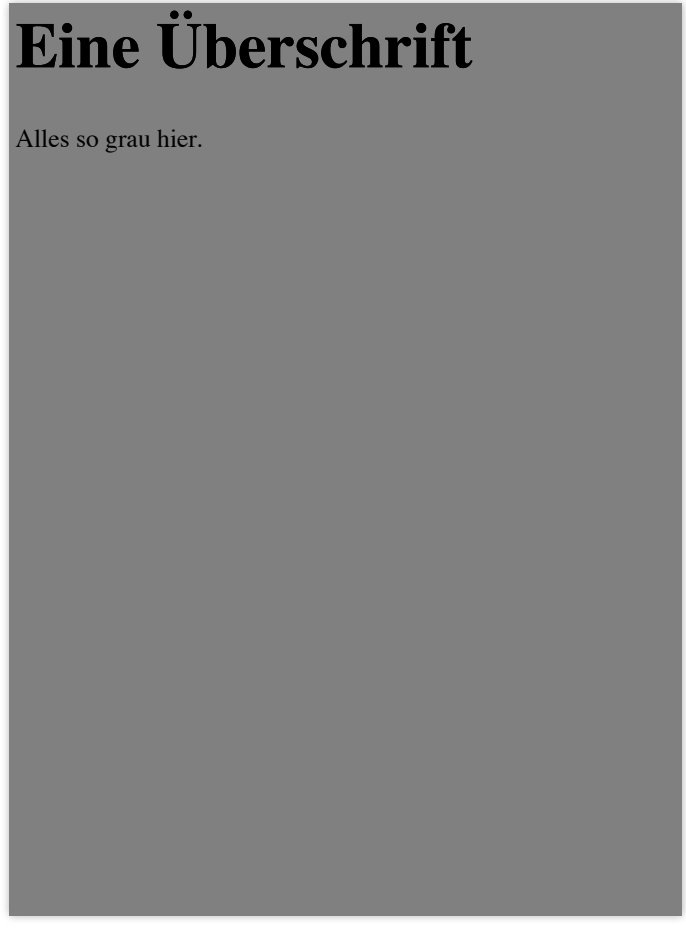
style.css

```
/* Alle Elemente im HTML-Dokument selektieren */
* {
  background-color: grey;
}
```

seite.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="style.css">
  </head>

  <body>
    <h1>Eine Überschrift</h1>
    <p>Alles so grau hier.</p>
  </body>
</html>
```



ATTRIBUTSELEKTOR

- Selektiert HTML-Elemente anhand ihrer Attribute und deren Werten
- Syntax und Ausprägungen:

Attributselektor	**Selektiert...**
`[attributname]`	alle Elemente mit Attribut `attributname`
`[attributname="wert"]`	alle Elemente, bei denen das Attribut `attributname` den Wert `wert` hat
`[attributname~="wert"]`	alle Elemente, bei denen das Attribut `attributname` als Wert eine Liste von durch Leerzeichen getrennten Worten hat, von denen eines `wert` entspricht

ATTRIBUTSELEKTOR (2)

- Syntax und Ausprägungen (Fortsetzung):

Attributselektor	Selektiert...
`[attributname|="wert"]`	alle Elemente, bei denen der Wert des Attributes `attributname` entweder ▪ `wert` entspricht, oder ▪ mit `wert` beginnt, gefolgt von einem Bindestrich
`[attributname^="wert"]`	alle Elemente, bei denen der Wert des Attributes `attributname` mit `wert` beginnt
`[attributname$="wert"]`	alle Elemente, bei denen der Wert des Attributes `attributname` mit `wert` endet
`[attributname*="wert"]`	alle Elemente, bei denen der Wert des Attributes `attributname` die Zeichenfolge `wert` enthält

ATTRIBUTSELEKTOR: BEISPIEL

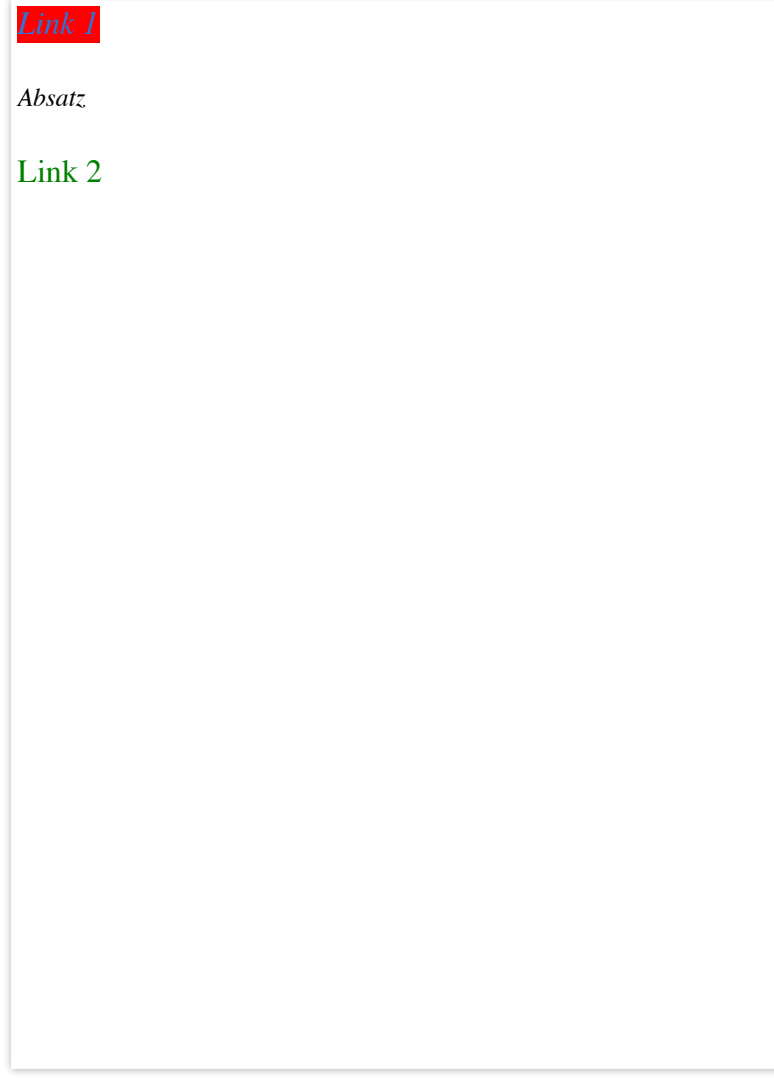
style.css

```css
/* Alle Elemente mit Attribut "title" */
[title] { font-style: italic; }
/*
 * Alle Elemente mit Attribut "href" und
 * Wert der mit ".de" endet
 */
[href$=".de"] { background-color: red; }
/*
 * Alle Elemente mit Attribut "href" und
 * Wert der ein "#" enthält
 */
[href*="#"] { color: green; }
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>

  <body>
    <a href="http://foo.de" title="Link">Link 1</a>
    <p title="Absatz">Absatz</p>
    <a href="#unten">Link 2</a>
  </body>
</html>
```



PSEUDOKLASSEN

- Selektieren HTML-Elemente, die sich in einem bestimmten Zustand oder an einer bestimmten Position befinden
- Syntax:
  - reguläre Pseudoklassen: `:pseudoklasse`
  - funktionale Pseudoklassen: `:pseudoklasse(argumente)`
- CSS bietet ca. 40 solcher Pseudoklassen, daher betrachten wir im Folgenden ein paar Beispiele

PSEUDOKLASSEN FÜR HYPERLINKS

Pseudoklasse	Selektiert...
`:link`	bisher unbesuchte Hyperlinks
`:visited`	schon besuchte Hyperlinks
`:target`	das Ziel eines Hyperlinks, sobald dieses angesprungen wurde

PSEUDOKLASSEN FÜR HYPERLINKS (2)

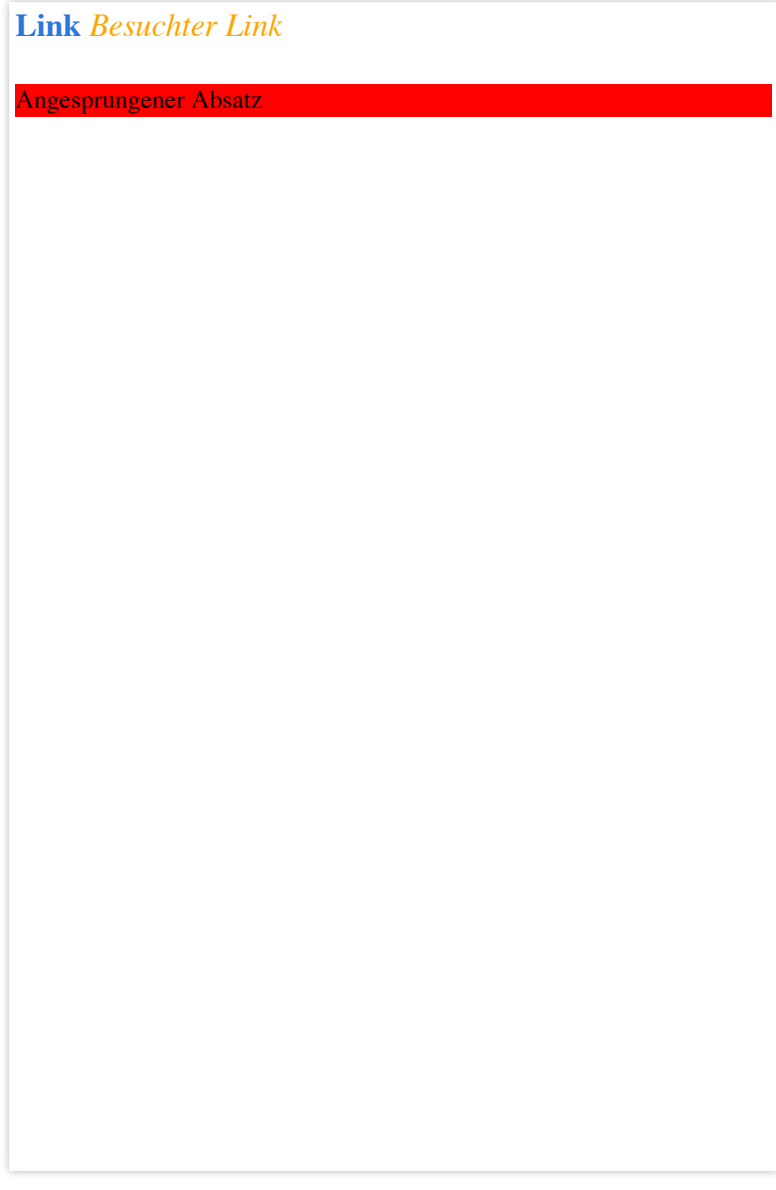
style.css

```css
/* Alle unbesuchten Hyperlinks selektieren */
a:link { font-weight: bold; }
/* Alle besuchten Hyperlinks selektieren */
a:visited {
  font-style: italic;
  color: orange;
}
/* Das durch einen Hyperlink angesprungene
   Ziel selektieren */
:target { background: red; }
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css"
          href="style.css">
  </head>

  <body>
    <a href="http://foo.de" title="Link">Link</a>
    <a href="#absatz">Besuchter Link</a>
    <p id="absatz">Angesprungener Absatz</p>
  </body>
</html>
```



PSEUDOKLASSEN FÜR BENUTZERAKTIONEN

Pseudoklasse	Selektiert...
`:hover`	Elemente, über denen sich der Mauszeiger befindet
`:active`	Elemente, die gerade aktiviert (z.B. angeklickt) werden
`:focus`	Element, das den Fokus hat

PSEUDOKLASSEN FÜR EINGABEELEMENTE

Pseudoklasse	Selektiert...
`:valid`/`:invalid`	Eingabeelemente, die valide/invalide Daten enthalten
`:required`/`:optional`	Eingabeelemente, bei denen Eingaben erforderlich/nicht erforderlich sind (siehe HTML-Attribut `required`)
`:checked`	Radiobuttons oder Checkboxen, die ausgewählt sind
`:enabled`/`:disabled`	Eingabeelemente, die freigegeben/deaktiviert sind (siehe HTML-Attribut `disabled`)
`:read-only`/`:read-write`	Eingabeelemente, die nicht editierbar/editierbar sind (siehe HTML-Attribut `readonly`)

PSEUDOKLASSEN FÜR EINGABEELEMENTE: BEISPIELE

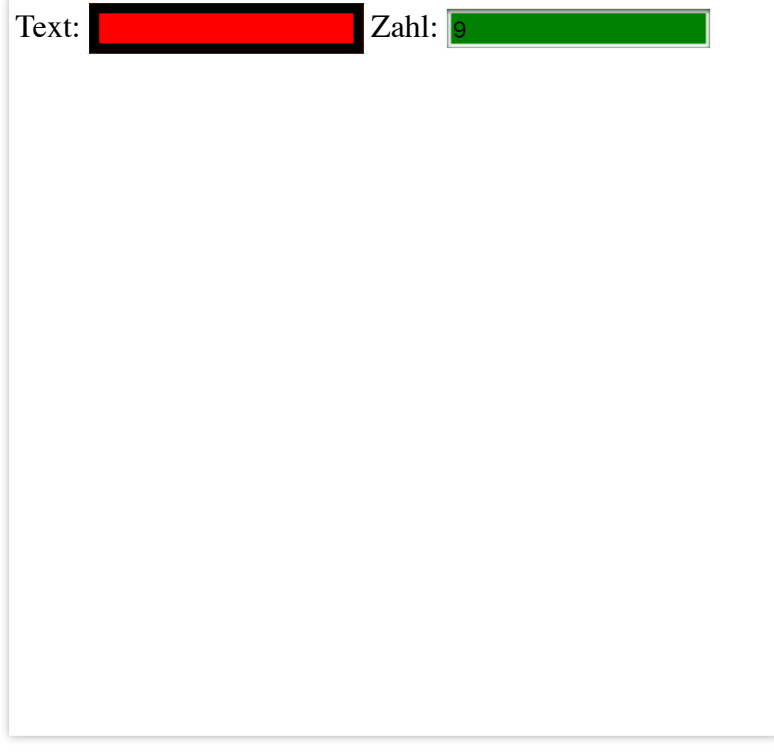
style.css

```css
/* Eingabeelemente mit gültigen Werten selektieren */
input:valid { background-color: green; }
/* Eingabeelemente mit ungültigen Werten selektieren */
input:invalid { background-color: red; }
/* Eingabeelemente mit erforderlichen Eingaben selektieren */
input:required { border: 5px solid; }
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>

  <body>
    Text: <input type="text" required>
    Zahl: <input type="number" max="10">
  </body>
</html>
```



STRUKTUR-PSEUDOKLASSEN

Pseudoklasse	Selektiert...
`:empty`	leere Elemente (d.h. ohne Inhalt)
`:first-child` / `:last-child`	Elemente, die das erste/letzte Kindelement ihres Elternelements sind
`:nth-child(n)`	Elemente, die das n-te Kindelement ihres Elternelements sind, wobei `n` folgendes sein kann: • ein ganzzahliger Wert, • ein arithmetischer Ausdruck, der einen ganzzahligen Wert ergibt, • `odd` für "ungerade Zahlen" • `even` für "gerade Zahlen"
`:nth-of-type(n)`	Elemente, die das n-te gleichartige Kindelement ihres Elternelements sind, mögliche Werte von `n` analog zu `:nth-child(n)`

STRUKTUR-PSEUDOKLASSEN (2)

style.css

```css
/* Alle ungeraden Listeneinträge selektieren */
li:nth-of-type(odd) { background-color: blue; }
/* Alle geraden Listeneinträge selektieren */
li:nth-of-type(even) { background-color: red; }
/* Den ersten Listeneintrag selektieren */
li:first-child { font-style: italic; }
/* Leere Elemente selektieren */
:empty {
  background-color: yellow;
  padding: 10px;
}
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>

  <body>
    <ul>
      <li>JavaScript</li>
      <li>HTML</li>
      <li>CSS</li>
    </ul>
    <p></p>
  </body>
</html>
```

- *JavaScript*
- HTML
- CSS

PSEUDOELEMENTE

- Selektieren HTML-Elemente, die nicht in der Struktur des HTML-Dokuments vorhanden sind
- Syntax: `::pseudoelement`

❗ Pseudoelemente werden erst seit CSS3 mit `::` notiert, um sie von Pseudoklassen zu unterscheiden - manche Browser erlauben auch noch die alte Schreibweise `:pseudoelement`

Empfehlung: Nur die neue Schreibweise verwenden!

PSEUDOELEMENTE (2)

Pseudoelement	Bedeutung
`::first-letter`	Selektiert das erste Zeichen in einer Zeile
`::first-line`	Selektiert die erste Zeile in einem Absatz
`::before`	Fügt formatierbaren Inhalt vor einem Element ein*
`::after`	Fügt formatierbaren Inhalt nach einem Element ein*

* Verwendung in Kombination mit der CSS-Eigenschaft `content`.

PSEUDOELEMENTE: BEISPIEL

style.css

```css
/* Das erste Zeichen in einem Abschnitt selektieren */
p::first-letter { font-size: xx-large; }
/* Die erste Zeile in einem Abschnitt selektieren */
p::first-line { font-family: fantasy; }
/* Inhalt vor Überschrift einfügen */
h1::before { content: url(assets/hand.png); }
/* Inhalt nach Überschrift einfügen */
h1::after { content: "!"; }
```

seite.html

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Titel meiner Web-Seite</title>
    <meta charset="utf-8">
    <link rel="stylesheet" type="text/css" href="style.css">
  </head>

  <body>
    <h1>Einleitung</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    tempor incididunt ut labore et dolore magna aliqua.</p>
  </body>
</html>
```