HTML UND CSS

PROF. ANDREAS HARTMANN - WEBTECHNOLOGIEN

INHALT

HTML5

- Konzept
- Beispiele

CSS3

- Konzept
- Beispiele

Übungsaufgaben

- Vorbereitung
- Dokumentation

QUELLEN, LITERATUR

- Quellenangaben

  Der Inhalt richtet sich – soweit nicht anders gekennzeichnet – aus an:

  - HTML5 und CSS3 : das umfassende Handbuch, Jürgen Wolf, Rheinwerkverlag, Bonn, 2015 (http://d-nb.info/1062929829)
  - Webarchitektur: Erik Wilde, UC Berkeley, (http://dret.net/lectures/web-fall10/)
- Es handelt sich um eigene Darstellungen wenn keine Quelle angegeben ist.
- Zum Nachschlagen mit vielen Beispiele ebenfalls gut:
  - http://www.html-seminar.de

KONVENTIONEN

- Alle Beispiele wurden mit Brackets geschrieben und mit Chrome angezeigt.
- Zitate werden mit „Zitat“ gekennzeichnet.
- Code wird mit `code` gekennzeichnet.
- Namen und Hervorhebungen werden mit *Name* gekennzeichnet.

HTML-EINFÜHRUNG

KONZEPT, WEBSEITEN MIT HTML5

HINWEIS

In diesem Lernabschnitt werden Sie einige Beispiele kennenlernen. Jedes Thema wird dabei jeweils einmal mit und ohne konkreten Inhalt gezeigt. Die Inhaltsbeispiele beziehen sich auf eine fiktive Webseite zum Comic Asterix.

WARUM HTML

Warum HTML lernen, wenn es doch tolle Werkzeuge gibt?

Denken Sie an ein Projekt beim Arbeitgeber/Auftraggeber:

Im Projekt soll u.a. eine umfangreiche Webseite erstellt werden (z.B. Wiki).

Projektleiter/in kommt aus anderem Bereich und kennt sich mit IT nicht aus.

Wer wählt die richtigen Werkzeuge und Technologien aus?

Wer entscheidet, ob die Werkzeuge "sinnvollen" HTML-Code erzeugen?

Wer identifiziert Probleme bei der automatischen Generierung von HTML?

Es geht also in erster Linie darum, HTML-Grundkenntnisse zu besitzen und anwenden zu können.

HINTERGRÜNDE ZUR ENTSTEHUNG (CA. 1990)

Das Internet hatte die Kommunikation nachhaltig verändert.

Informationen waren digital und weltweit zugänglich.

Protokolle (`http`) und Netzwerk (`tcp`) vorhanden, es fehlte aber noch ein Standard für die grafische Darstellung.

Das war die Geburtsstunde von HTML (1992) und *Mosaic* (1993), einem der ersten *Web-Browser*.

GRUNDLAGEN

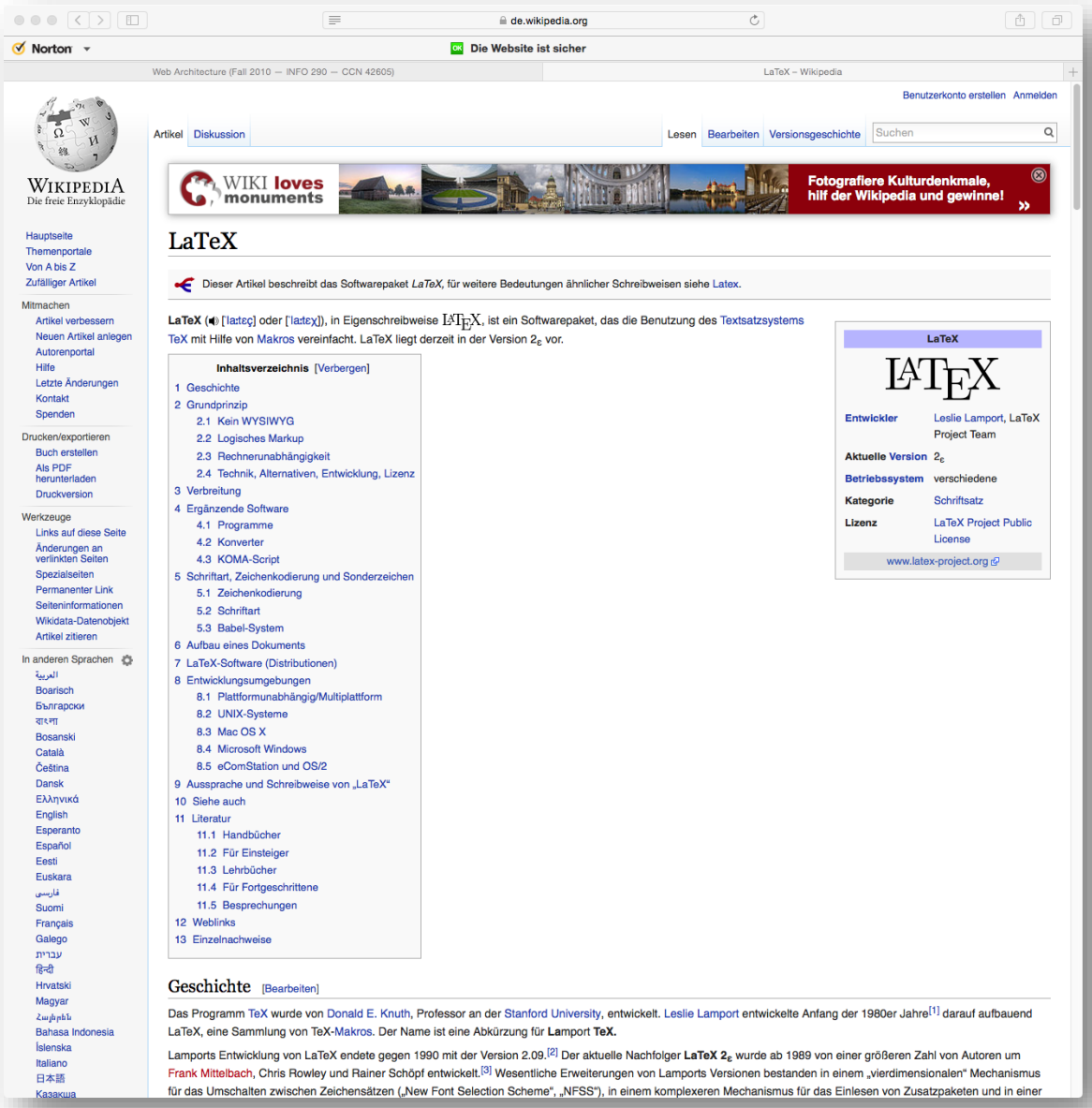
- HTML ist eine Textauszeichnungssprache.
- HTML basiert auf SGML.
- Die Auszeichnung erfolgt über SGML-Elemente (“Tag’s”).

*kurz für „Standard Generalized Markup Language“; eine Sprache zur Beschreibung der Struktur von Textdokumenten

HTML & WEBSEITE

Sie kennen HTML in Form einer angezeigten Webseite.

BEISPIEL: WEBSEITE



UNTERSCHIEDLICHE ARTEN

Wir unterscheiden statische und dynamische Webseiten.

statisch

manuelle Erstellung

HTML-Datei

auf Webserver geladen

unabhängig

“Wenn es schnell gehen muss”

dynamisch

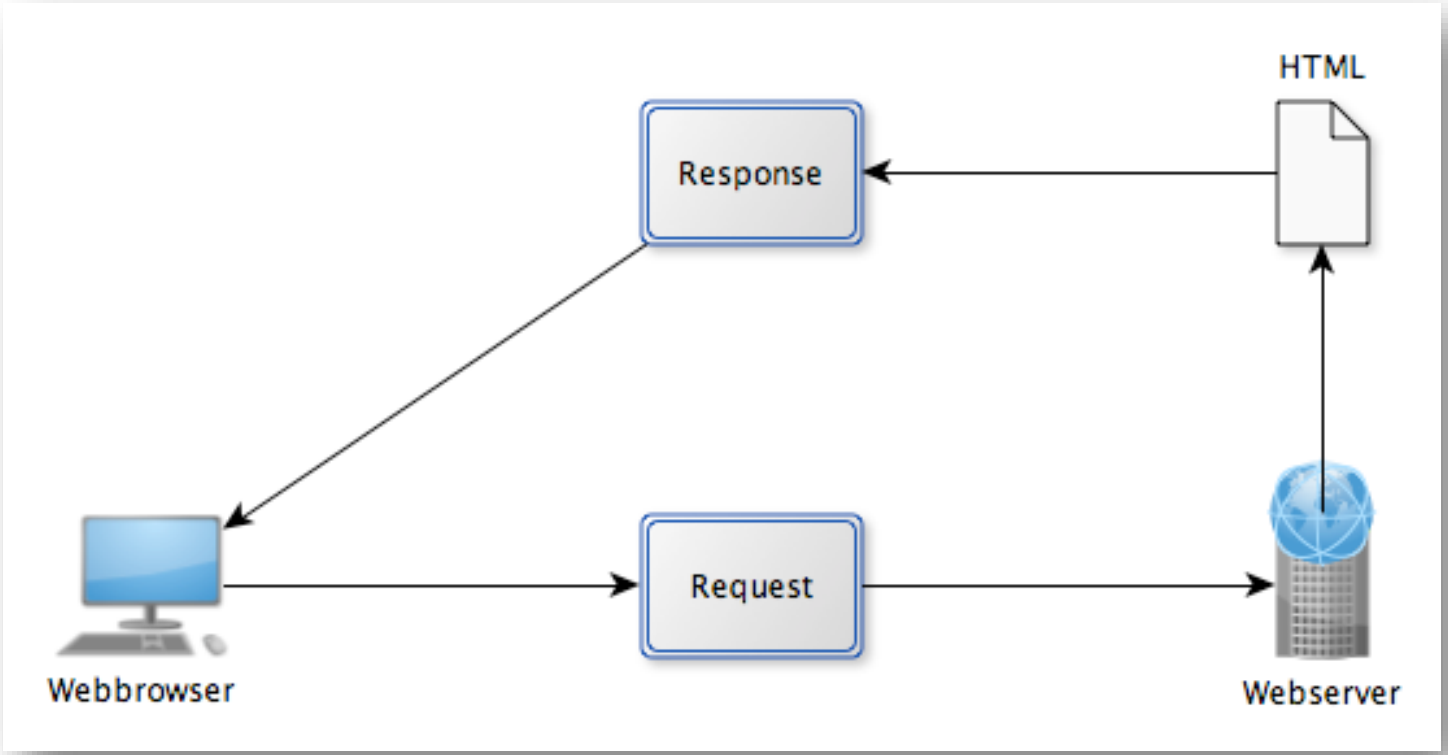
automatische Erstellung

Trennung von Inhalt & Programmierung

Werkzeug erforderlich

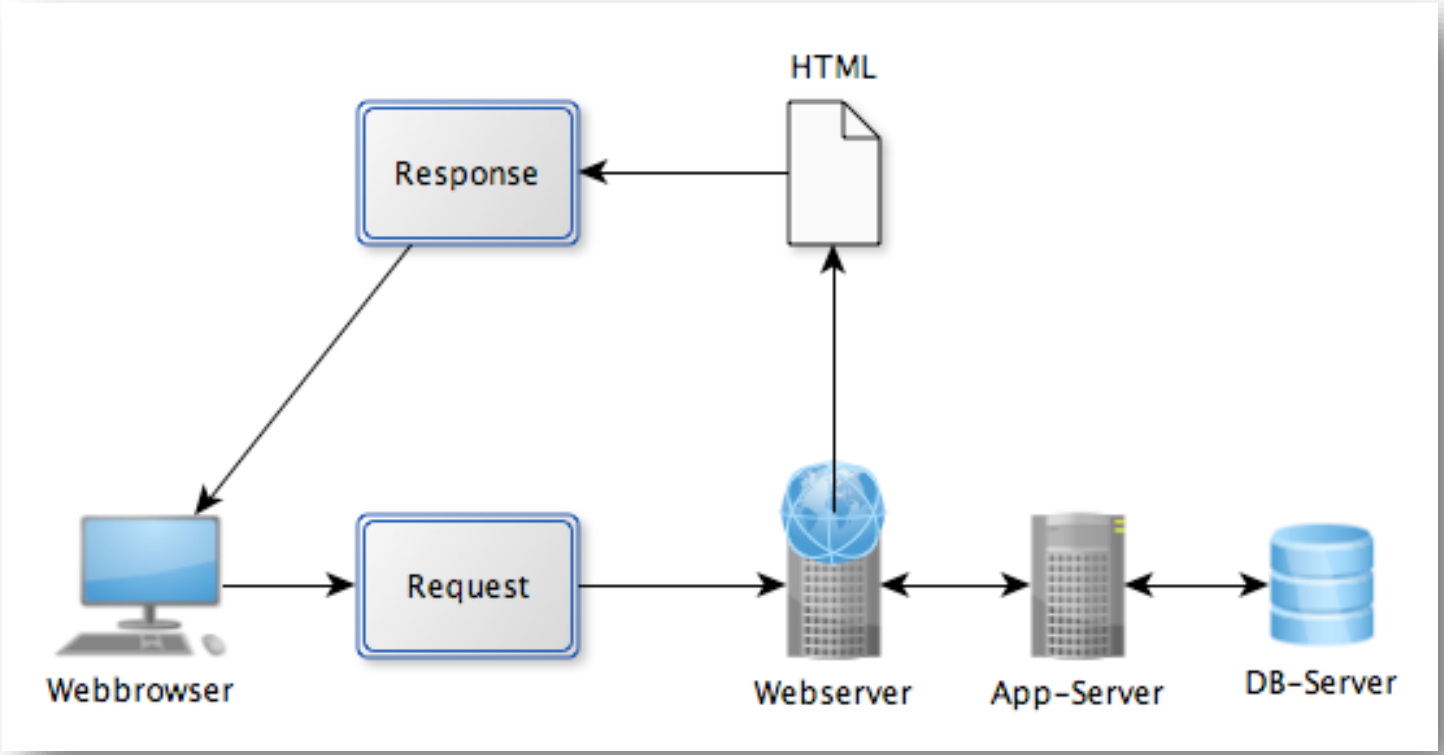
abhängig

“Wenn es umfangreich ist”



STATISCHE WEBSEITE

schematische Darstellung



DYNAMISCHE WEBSEITE

schematische Darstellung

„HTML-PROGRAMMIERUNG“

HTML ist keine Programmiersprache! Beachten Sie auch die Unterschiede bzgl. Webapplikationen.

ABGRENZUNG ZUR PROGRAMMIERSPRACHE

HTML

Markup-Language (Auszeichnungssprache)

Output:

HTML-Dokument

Ziel:

Informationen übertragen

wird **dargestellt**

C, C++, Java, ...

Programmiersprache (Hochsprachen)

Output:

ausführbares Computerprogramm

Ziel:

i.d.R. eine Aufgabe lösen

wird **ausgeführt**

MARKUP-LANGUAGE

Die Definition und die Grundlagen sehen wir uns an, wenn wir XML betrachten.

WERKZEUGE FÜR DIE WEBSEITENGESTALTUNG

Da HTML vom Browser interpretiert wird, ist kein Compiler notwendig.

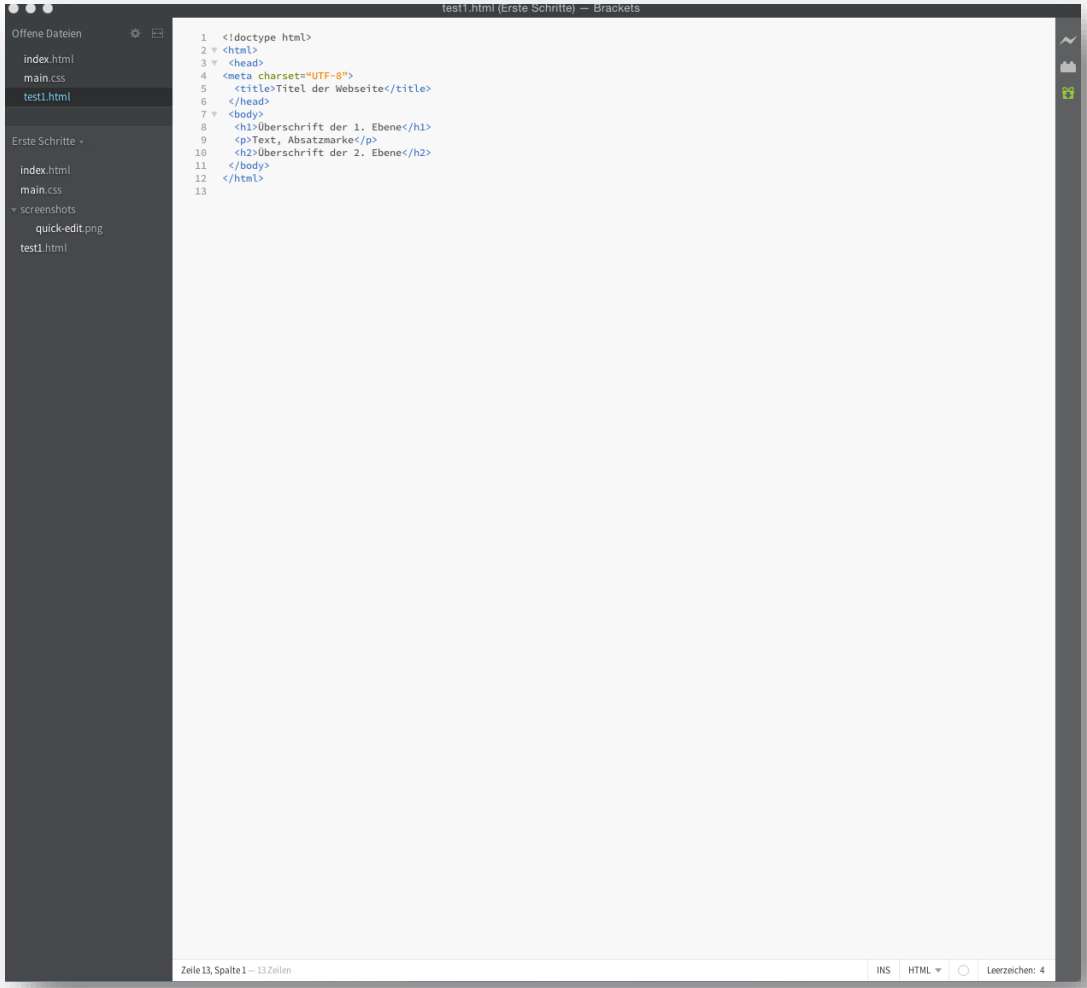
Ein einfacher Texteditor ist ausreichend.

HTML-Editoren sind hilfreich (z.B. Highlighting).

WYSIWYG-Editoren – hängt vom Einsatzgebiet ab.

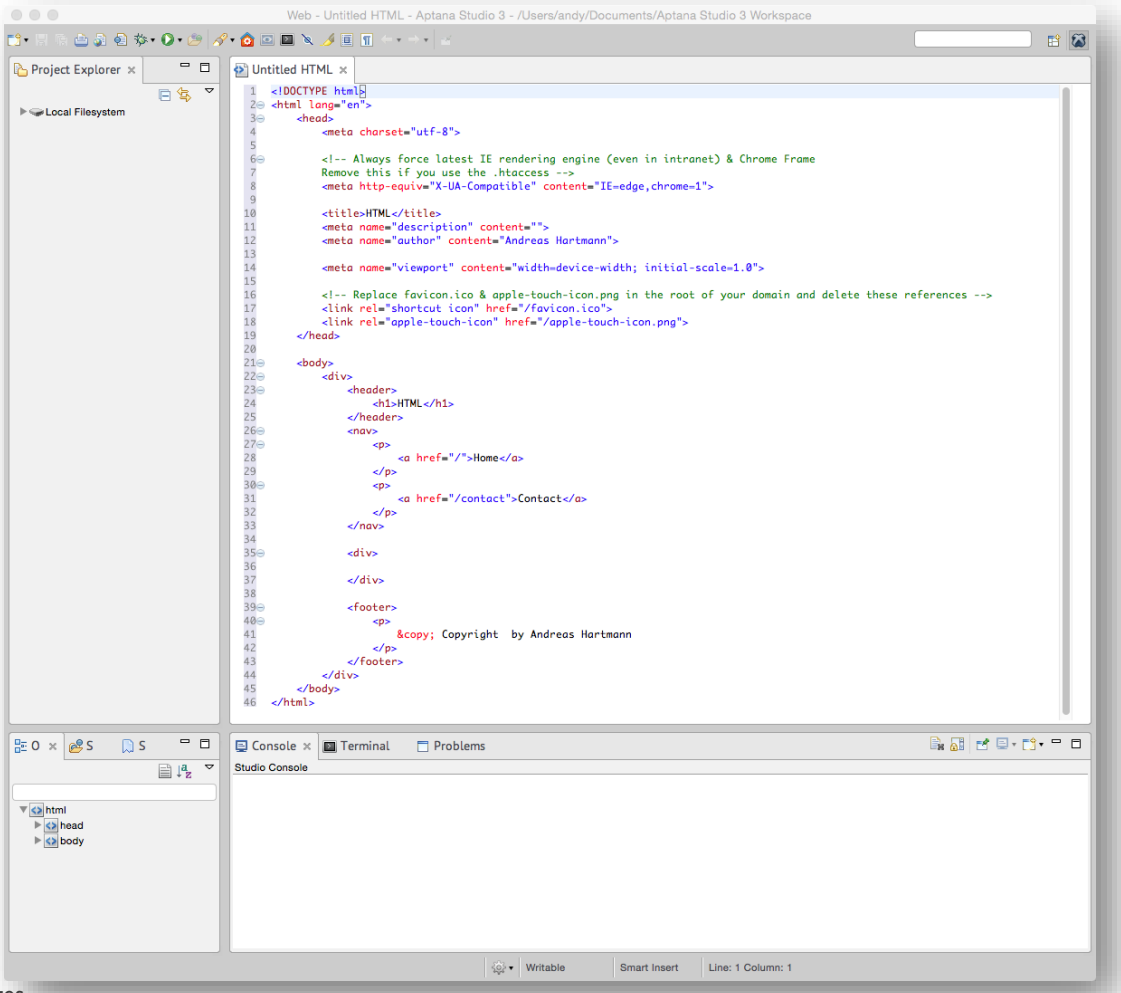
BRACKETS

HTML-Editor mit Anbindung an Chrome und vielen Extensions



APTANA STUDIO 3

HTML-Editor mit umfangreicher Projektverwaltung und Templates



CMS-WERKZEUGE

Bei großen Projekten können die Webseiten nicht alle einzeln bearbeitet werden. Eine Herausforderung ist auch die strukturierte Bereitstellung der Medien (Bilder, Video).

CONTENT-MANAGEMENT-SYSTEME

Typo3

- Open Source
- über 500.000 mal im Einsatz (Quelle: www.typo3.org)
- z.B. Standardsystem in der Bundesverwaltung

EgoCMS

- z.B. an der Friedrich-Schiller-Universität Jena (Stand 2015)
- über 60.000 einzelne Webseiten
- mehr als 400 Redakteure
- mehr als 15 einzelne Mandanten

** Stark veralteter Screenshot*

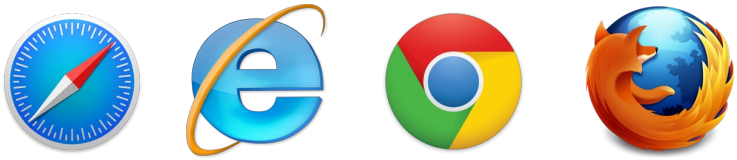
TYPO3 - SCREENSHOT

TECHNOLOGIEN

HTML dient dem Austausch von Informationen zwischen einem Server und einem Client. Welche Technologien werden dafür benötigt?

CLIENT-TECHNOLOGIEN

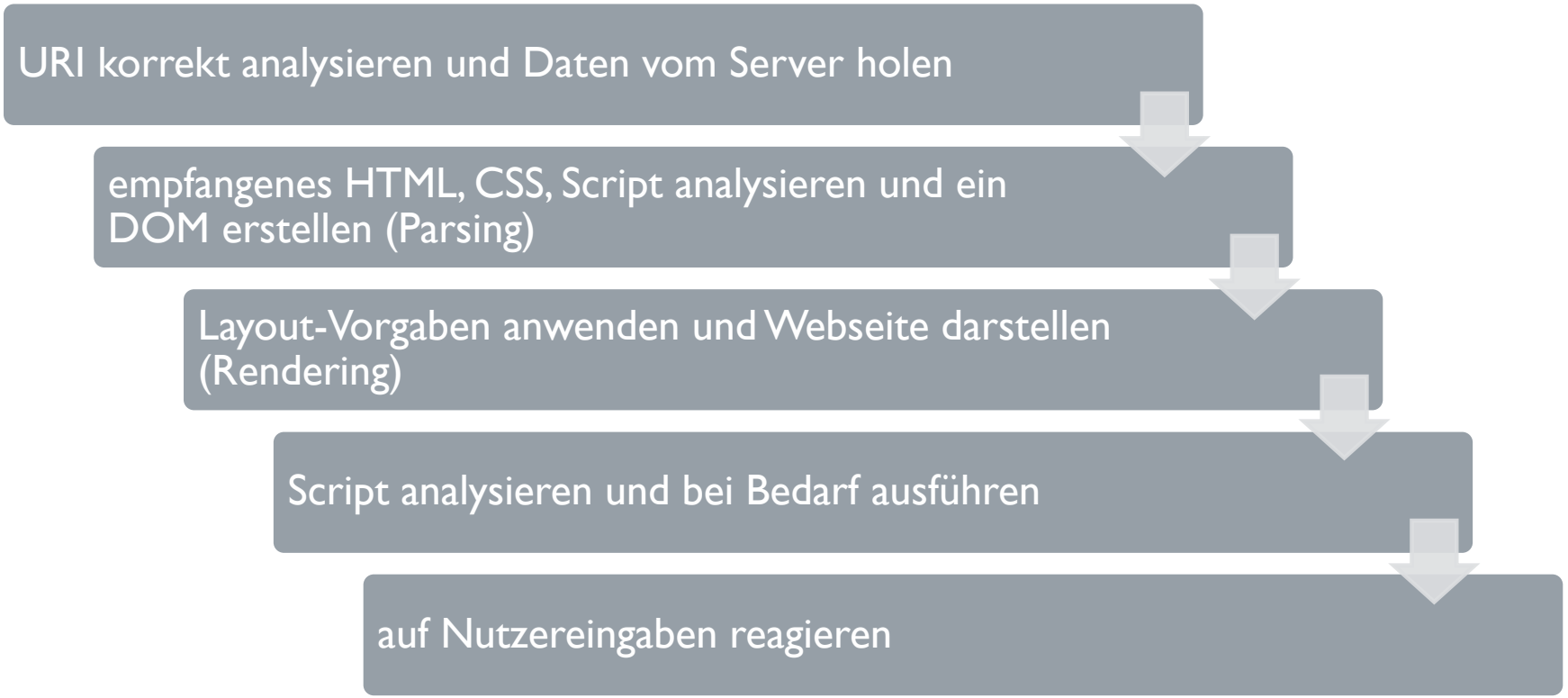
- Die Protokollunterstützung (`http`) ist heute i.d.R. im Betriebssystem implementiert.
  - vgl. *frühe* Implementierungen für mobile Geräte, z.B. Symbian OS!
  - vgl. eingebettete Systeme!
- Ein Browser, der sich an Standards hält ☺
  - IE, Edge
  - Safari
  - Chrome
  - Firefox



AUFGABEN DES BROWSERS

- Kommunikation, z.B. zwischen Browser und (Web-)Server bzw. zwischen Browser und Betriebssystem
- Anzeige eines *korrekt* dargestellten HTML-Dokuments auf der Anzeige des Anwenders (Rendering)
- Ausführung von Scripten (JIT oder Interpreter)
- Zusatzfunktionen

EIN TYPISCHER ABLAUF



SERVERTECHNOLOGIEN

- Webserver
  - im Intranet bzw. Internet eindeutig (URI) erreichbar (`http`)
  - beantwortet Anfragen des Clients
  - z.B. nginx, Apache
- weitere Technologien
  - Script-Technologien (PHP, Python, Ruby, Perl)
  - Datenbanken
  - Frameworks

HTML KONZEPT

- HTML = Markup-Language
  - vgl. auch LaTeX
  - vgl. auch XML
- Zweck ist die Darstellung von Inhalten auf dem Client.
- Fokus ist demnach die Strukturierung und das Layout des darzustellenden Inhalts (Dokuments).

- Was wäre die Alternative zu HTML? Gibt es dafür Beispiele?

HTML5 - STANDARD

Mit HTML5 wurde ein Standard verabschiedet, der noch stärker auf die Trennung von Inhalt, Semantik und Layout achtet. Im Vordergrund von HTML5 steht die Semantik.

Weiterhin wurde HTML erweitert, um auf Anforderungen z.B. von mobil genutzten Apps zu reagieren.

CONTENT, LOGIC & STYLE

- HTML verwendet *Tags* zur Beschreibung der Logik (Struktur) und des Styles (Darstellung).
- Der Content (Inhalt) ist nicht Bestandteil von HTML.
- Mit dem Konzept von Stylesheets wird der Style in CSS verlagert (dazu später: CSS).

- Jedes HTML-Dokument entspricht einer Baumstruktur – dem *Document Objekt Modell* (DOM).
- Hinweis: ein Browser ist daher immer auch ein Parser & Interpreter für das DOM.

HTML TAGS

- Sprachelemente von HTML sind Tags, z.B.

```
<body> </body>
```

- Tags haben ggf. Attribute zur näheren Beschreibung, z.B.

```
<a href="http://www.ix.de">ein Link</a>
```

- Kommentare

```
<!-- das ist ein Kommentar -->
```

- Tags können geschachtelt werden (Regeln beachten!)

BEISPIEL

Sehen wir uns ein einfaches Beispiel an. Links steht der Source-Code und rechts zeigt ein Screenshot die Anzeige unter OS X / Chrome.

AUFBAU EINER WEBSEITE MIT HTML

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
  <h1>Überschrift der 1. Ebene</h1>
  <p>Text, Absatzmarke</p>
  <h2>Überschrift der 2. Ebene</h2>
 </body>
</html>
```

GallierRugby

Unsere Beispielwebseite wird sich mit dem Thema *Asterix - der Gallier* beschäftigen. Wir werden die Comicfiguren zu einem Rugby-Team kombinieren und gegen die Römer antreten lassen.

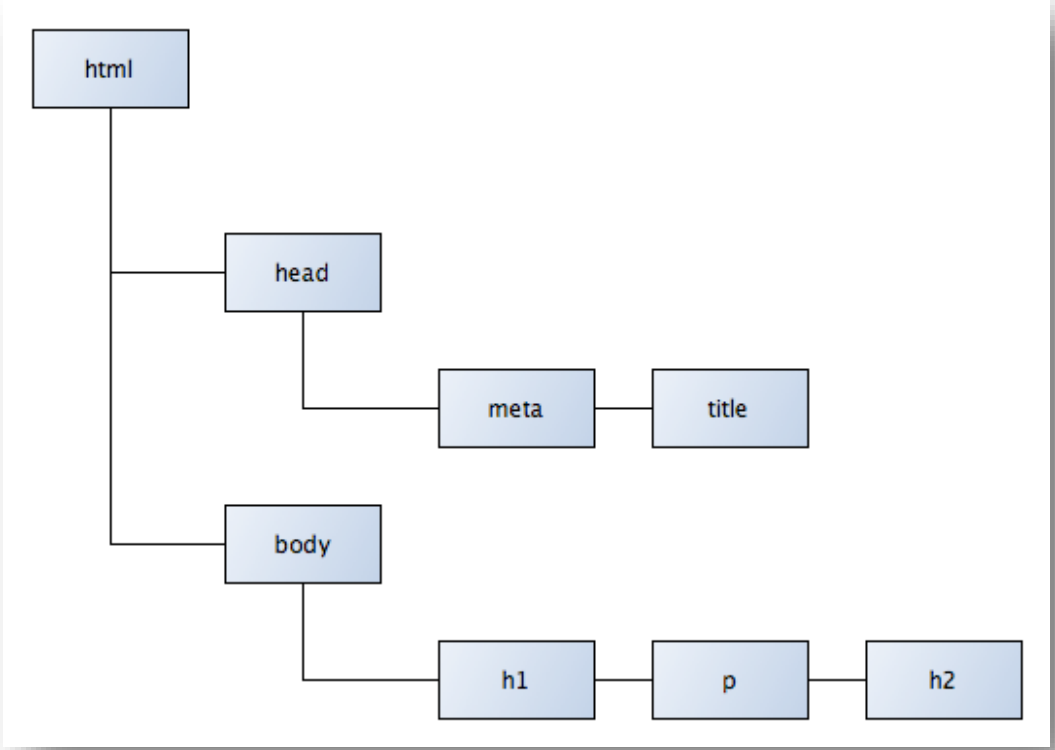
Das Team

UNSER BEISPIEL

Die Webseite mit zwei Überschriften (H1 und H2).

BEISPIEL (DOM)

Zu jeder Webseite gehört ein DOM – es ist die virtuelle Repräsentation der Webseite im Speicher des Computers.



DOCUMENT-OBJECT-MODEL (DOM)

DOM zum vorherigen Beispiel

AUFBAU DER WEBSEITE

Sehen wir uns den Aufbau der Webseite im Einzelnen an und lernen dabei die wichtigsten Elemente von HTML kennen.

ES GIBT 3 BEREICHE:

Metabereich (Dokumententyp)

Kopfbereich (Head)

Dokumentenkörper (Body)

METABEREICH

Der Metabereich kennzeichnet primär den verwendeten Sprachstandard. Im Beispiel entspricht das: HTML 5.

```
<!doctype html>
```

KOPFDATEN

Die Kopfdaten beschreiben allgemeine Informationen zur Webseite.

Sie sind wichtig u.a. für Suchmaschinenoptimierung und Einbindung von clientseitigen Scripts.

KOPFDATEN - ELEMENTE

Die folgende Tabelle zeigt einige typische HTML-Elemente für die Verwendung in den Kopfdaten.

KOPFDATEN

`<title>`	Titel des HTML-Dokuments
`<base>`	setzt die Basis-URL für alle relativen Links auf der Webseite
`<link>`	setzt logische Links zu anderen Dateien, die eingebunden werden sollen
`<style>`	setzt die lokalen Stylesheet-Regeln für das HTML-Dokument
`<script>`	bindet clientseitige Skripte ein
`<meta>`	setzt Metadaten, wie z.B. Schlüsselwörter und Zeichensatz

DOKUMENTENKÖRPER

= darstellbarer Inhaltsbereich des HTML-Dokuments

entspricht dem Inhalt, der vom Browser gerendert wird

Eine gute Webseite ist ordentlich strukturiert!

WEBSEITEN STRUKTURIEREN

Ein guter Startpunkt ist es, die Inhalte einer Webseite zunächst in eine Struktur zu bringen. Dazu bietet HTML einige semantische Möglichkeiten.

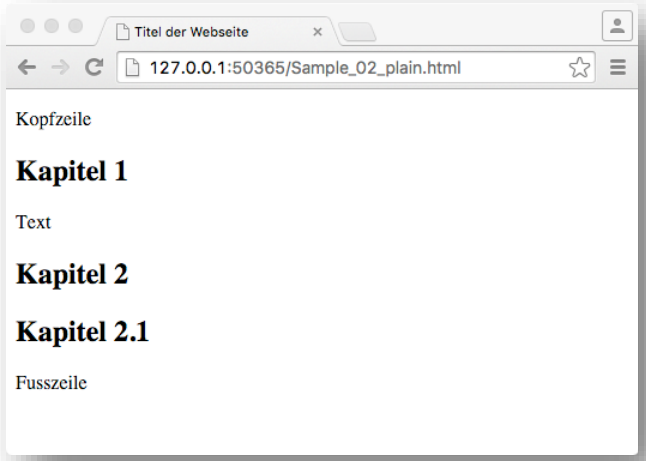
Wir beginnen Top-Down (von der Seite zum Text).

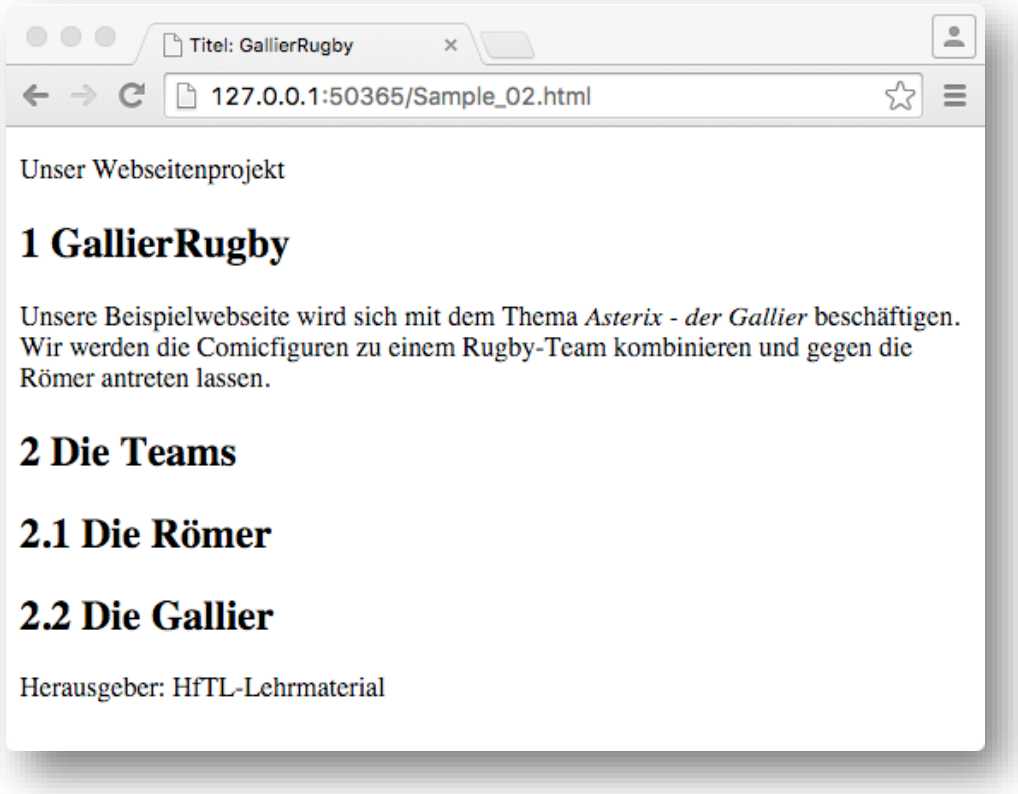
STRUKTURELEMENTE

`<body>`	der darstellbare Inhaltsbereich des HTML-Dokuments
`<section>`	Unterteilung in verschiedene Abschnitte (Sektionen)
`<article>`	Unterteilung von Inhalten in einen in sich geschlossenen themenspezifischen Block
`<nav>`	Element zur Auszeichnung von Navigationsleisten (Sitemap)
`<h1>,<h2>,...`	Überschriftenelemente
`<header>`	Kopfbereich eines Inhalts
`<footer>`	Fußbereich eines Inhalts
`<figure>`	Gruppieren von Inhalten zur gesonderten Beschreibung
`<figcaption>`	Beschriftung von ‚figure'-Inhalten
`<aside>`	gesonderte Darstellung von Inhalt in z.B. Seitenleiste

HTML5 STRUKTURIERUNG ANWENDEN

```
<!doctype html>
<html>
 <head>
<meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
     <header><p>Kopfzeile</p></header>
     <section>
         <h1>Kapitel 1</h1>
         <p>Text</p>
     </section>
     <section>
         <h1>Kapitel 2</h1>
         <section>
             <h2>Kapitel 2.1</h2>
         </section>
     </section>
     <footer><p>Fusszeile</p></footer>
 </body>
</html>
```





UNSER BEISPIEL

Strukturierung des Textinhalts.

ABSÄTZE DEFINIEREN

Die nächsten Elemente beziehen sich auf die Struktur innerhalb eines Absatzes. Dabei steht auch hier die Semantik im Vordergrund – um das Aussehen kümmert sich der Browser bzw. unser Stylesheet.

INHALTE (ABSATZEBENE)

`<p>`	Textabsatz
` `	Zeilenumbruch
`<wbr>`	optionaler Zeilenumbruch (innerhalb eines Wortes)
`<hr>`	thematische Trennung auf Absatzebene
`<blockquote>`	Zitat in Form eines Textabsatzes
`<div>`	Definition eines allgemeinen Bereichs
`<main>`	Definition des Hauptinhalts einer Webseite

BESONDERHEIT: DIV

Eine herausgehobene Stellung nimmt das Element `div` ein. Zunächst als *allgemeiner Bereich* eingeführt, steht es heute als Standardelement für einen Container, dessen Inhalt gemeinsam (intern) und zwar unabhängig von der Umgebung (extern) gestaltet werden soll.

Vor der Definition von HTML 5 wurde mit DIVs die Semantik auf der Webseite gesteuert. Wir sehen dazu später ein Beispiel.

TEXT DEFINIEREN

Abschließend sehen wir uns die Möglichkeiten bzgl. Text an. Analog zu z.B. LaTeX wird die Bedeutung herausgestellt und der Browser erledigt die visuelle Interpretation.

INHALTE (TEXTEBENE)

`<abbr>`	Abkürzungen oder Akronyme
`<cite>`	Kennzeichnung von Quelltext eines Arbeitstitels
`<code>`	Computercode auszeichnen
`<pre>`	präformatierten Text auszeichnen
`<kbd>`	Text als Tastatureingabe auszeichnen
...	

WEITERE ELEMENTE

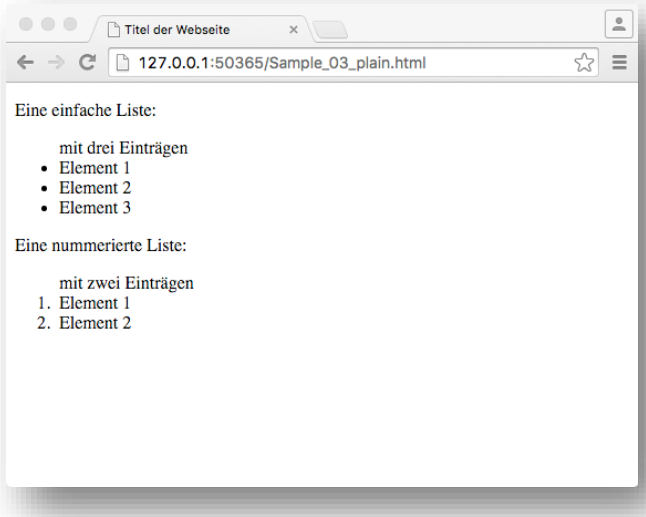
Nach den ersten strukturgebenden Elementen sehen wir uns weiter Möglichkeiten an, ein HTML-Dokument zu strukturieren.

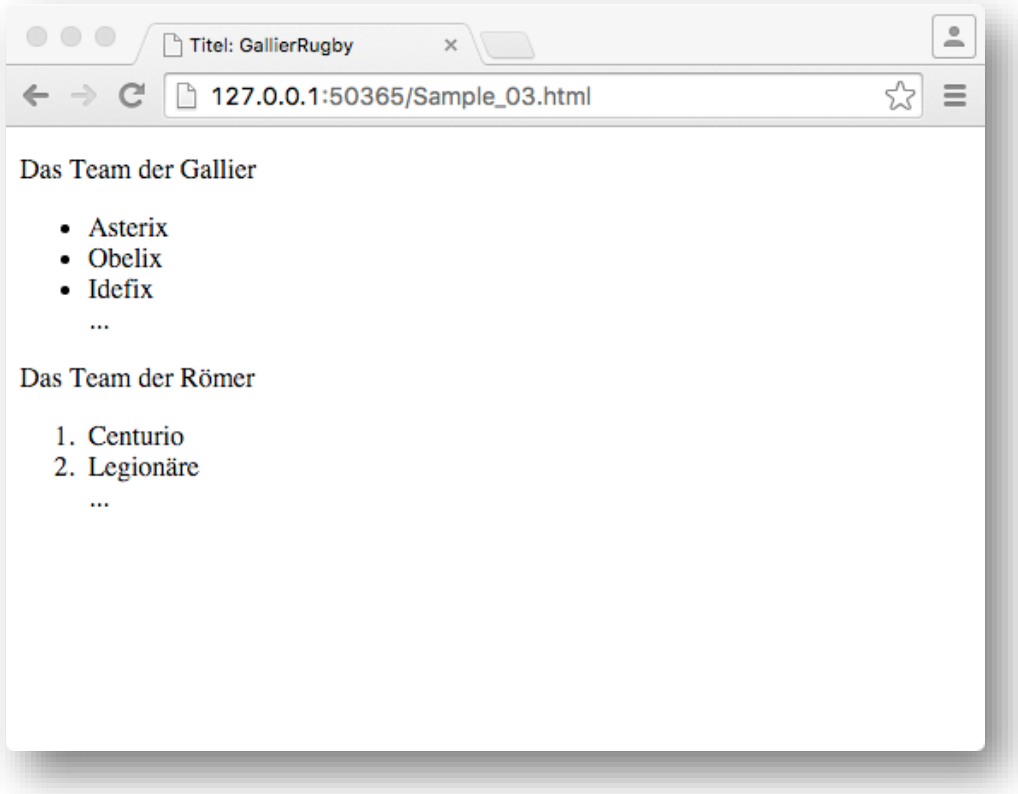
LISTEN

`<ul>`	ungeordnete Aufzählungsliste
`<ol>`	geordnete Aufzählungsliste (nummeriert)
`<li>`	ein Listenelement einer Liste
`<dl>,<dt>, <dd>`	Erstellen von Beschreibungslisten (eingerückte Elemente, z.B. für Glossare)

LISTENELEMENTE

```
<!doctype html>
<html>
 <head>
<meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
 <article>
     <p>einfache Liste</p>
     <ul>
         <li>Element 1</li>
         <li>Element 2</li>
         <li>Element 3</li>
     </ul>
     <p>nummerierte Liste</p>
     <ol>
         <li>Element 1</li>
         <li>Element 2</li>
     </ol>
 </article>
 </body>
</html>
```





UNSER BEISPIEL

Integration von einfachen Listen.

TABELLEN

Tabellen wurden vor der Einführung von HTML5 und CSS oft zur optischen Gestaltung von Webseiten genutzt (Grid).

HTML5 verbietet viele dieser Formatierungsmöglichkeiten!

TABELLEN

`<table>`	Tabelle
`<tr>`	Tabellenzeile
`<td>`	Tabellenzelle
`<th>`	Tabellenkopf (Kopfzelle)
`<thead>`	Tabellenkopfbereich
`<tbody>`	Tabellenkörper
`<tfoot>`	Tabellenfußbereich
`<colgroup>`	Gruppe von Tabellenspalten
`<col>`	Tabellenspalte
`<caption>`	Tabellenbeschriftung

LINKS

Ohne Links wären Webseiten ziemlich langweilig. Es gibt die Möglichkeit, auf interne Seiten desselben Webservers oder auf externe Seiten zu verlinken.

HYPERLINKS

```
<a href="http://www.ix.de">Link zu IX</a>
```

- externe und interne Links
- wichtig für Navigation
  - siehe `<nav>`
- absolute und relative Pfade
  - Relative Pfade beziehen sich auf: `<base>`
- Target definiert, wo der Link geöffnet wird

```
<a target="_blank" href="http://www.ix.de">IX</a>
```

- Hyperlinks können nicht nur das Protokoll `http` aufrufen:

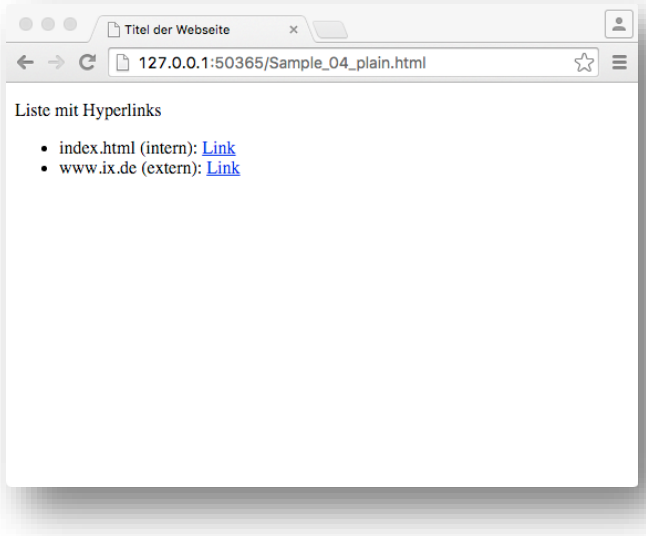
```
<a href="mailto:kontakt@webseite.de">E-Mail senden</a>
```

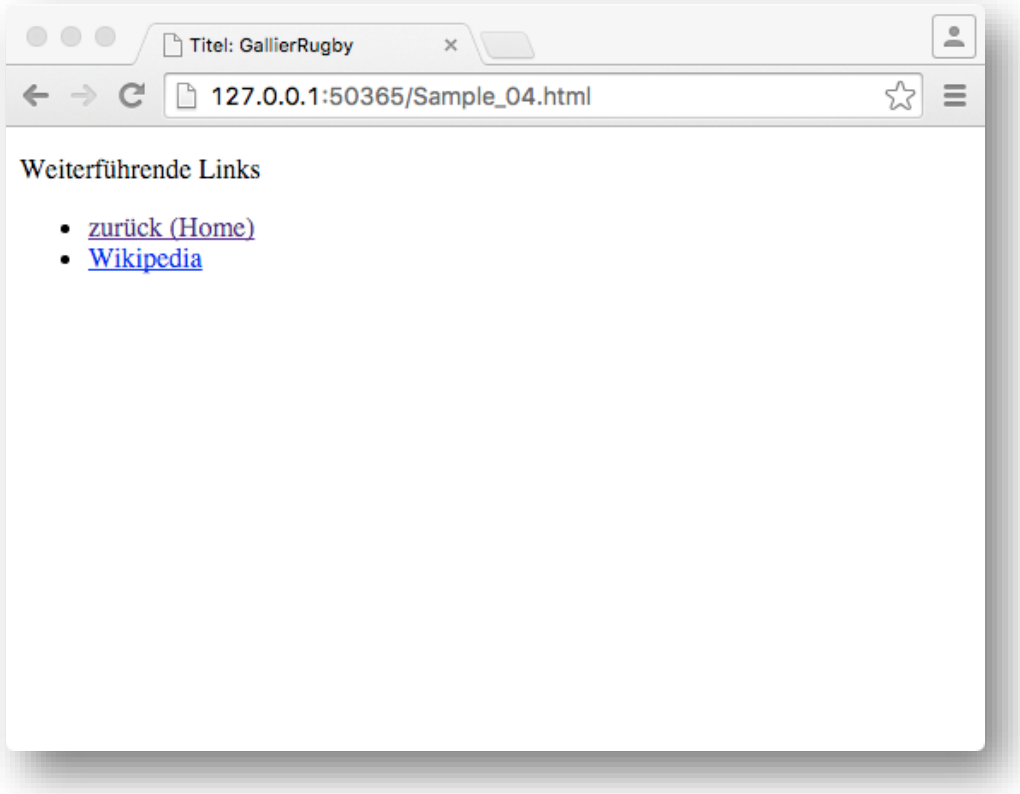
ATTRIBUTE FÜR <a>

download	Link wird automatisch als Download angeboten
href	URL für das Ziel
media	Angaben zum Medientyp des Ziels
target	Definition wie der Link geöffnet wird
type	Definition des MIME-Typs des Ziels

HYPERLINKS

```
<!doctype html>
<html>
 <head>
<meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
 <article>
    <p>Liste mit Hyperlinks</p>
    <ul>
     <li><a href="index.html">
      Hyperlink 1 (intern)</a>
     </li>
     <li><a href="http://www.ix.de">
      Hyperlink 2  (extern)</a>
     </li>
    </ul>
 </article>
 </body>
</html>
```





UNSER BEISPIEL

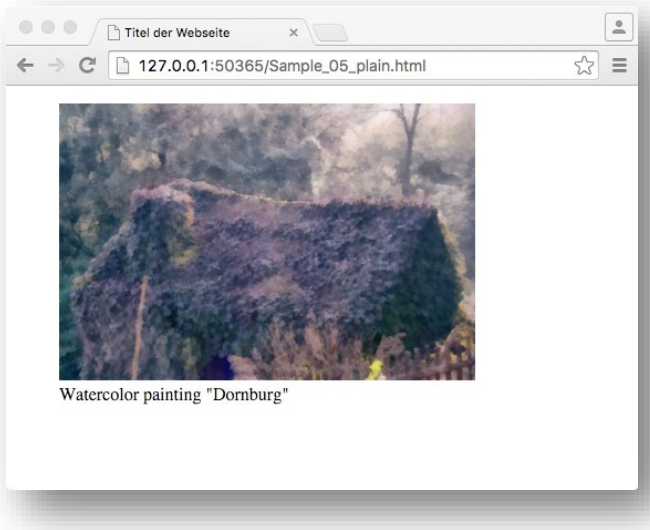
Links, die wir später im Fußbereich platzieren.

BILDER

Viele Webseiten leben von ihren interessanten Bildern. Ohne Bilder würden Informationen ausschließlich in Textform zur Verfügung stehen und das Internet wäre dann doch recht „öde“.

BILDER

```
<!doctype html>
<html>
 <head>
<meta charset=“UTF-8”>
  <title>Titel der Webseite</title>
 </head>
 <body>
  <img src="image.jpg" height="250"
alt="Dornburg">
 </body>
</html>
```



UNSER BEISPIEL

Unsere wichtigsten Charaktere als 3 Bilder in einer (Gruppe) Figure.

ATTRIBUTE FÜR <IMG>

alt	Angabe eines alternativen Textes, wenn Bilder nicht angezeigt werden können
height	Angabe für die Höhe des Bildes
ismap	Angabe, ob es sich um eine serverseitige Image-Map handelt
src	Verweis auf die Bilddatei
usemap	Angabe einer Image-Map, die mit dem Bild verknüpft wird
width	Angabe für die Breite des Bildes

BILDFORMATE

Typische Bildformate sind JPG, GIF oder PNG. Moderne Browser können eine Vielzahl von Bildformaten verarbeiten und korrekt anzeigen.

DAS CONTENTMODELL

Wie gesagt, ist HTML 5 deutlich stärker auf die Semantik ausgerichtet. Es definiert daher ein umfangreiches Contentmodell, in dem die einzelnen HTML-Elemente unterschiedlichen Kategorien zugeordnet werden.

CONTENT-MODELL VON HTML5

- Das Content-Modell definiert die Semantik.
- Alle HTML-Elemente werden mindestens einem Content und somit semantisch zugeordnet.
- HTML5 standardisiert stärker als frühere HTML-Versionen:
  - HTML4 kannte nur sogenannte *Block*- und *Inline*-Elemente
  - HTML5 definiert die Semantik mit insgesamt 7 Kategorien

KATEGORIEN

Die nachfolgende Tabelle zeigt links die Kategorien und rechts die entsprechend zugeordneten Elemente. Natürlich dient dieses Modell dazu, die Kategorien unterschiedlich zu steuern (z.B. bei Umbrüchen).

CONTENT-KATEGORIEN

`Flow`	Fließtext-Elemente, z.B. `a, article, aside, code, footer, h1`
`Sectioning`	Elemente zur Festlegung von Sektionen im Dokument, z.B. `article, aside, nav, section`
`Heading`	Überschriften-Elemente, `h1...h6`
`Phrasing`	Elemente zur Auszeichnung von Text, z.B. `abbr, b, i, em`
`Embedded`	eingebundene und importierte Elemente, `audio, canvas, embed, iframe, img, math, object, svg, video`
`Interactive`	Elemente dieser Kategorie bieten dem Anwender eine Interaktionsmöglichkeit, z.B. `button, input, video`
`Metadata`	Elemente zur „Meta“-Steuerung des HTML-Dokuments, z.B. `base, link, meta, script, style`

SEMANTIK MIT DIVS

Vor HTML5 fehlten eindeutige Strukturelemente, wie `<nav>`. Zum Einsatz kamen daher Container, die mittels `<div>` definiert wurden. Lediglich eine nutzerspezifische ID wurde zur Unterscheidung eingesetzt.

ANWENDUNG DER SEMANTIK UNTER HTML4

```
<div id="Name"> </div>
```

das erhöhte die Komplexität, denn die Semantik liegt nicht in HTML sondern in selbst definierten Zeichenketten (hier: "Name")

```
<div id="header">…</div>
<div id="footer">…</div>
<div id="section">…</div>
```

ANWENDUNG DER SEMANTIK UNTER HTML5

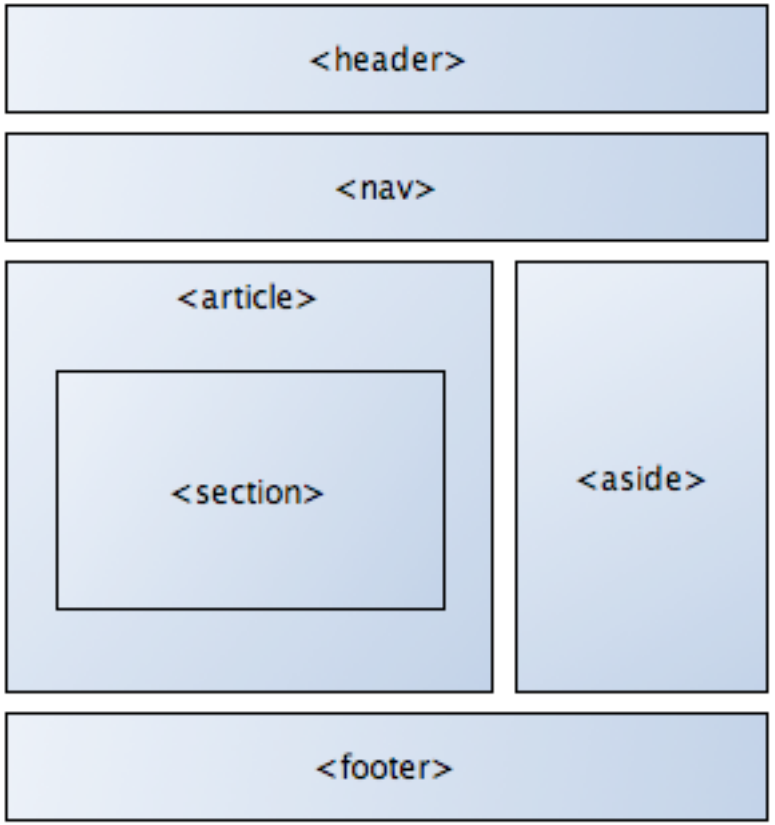
Die wichtigsten Strukturelemente müssen nicht über eine eigene ID angesprochen werden.

```
<header>...</header>
<footer>...</footer>
<section>...</section>
<nav>...</nav>
```

Viel einfacher, denn die Semantik wird jetzt durch HTML selbst vorgegeben.

SEMANTIK IM BROWSER

Wir haben jetzt die Semantik unter HTML5 kennengelernt. Aber wie „übersetzt“ der Browser diese Elemente und wie stellt er sie strukturiert dar?



SEMANTIK VON HTML5

Typische Darstellung einer einfachen Webseite

KOMMUNIKATION

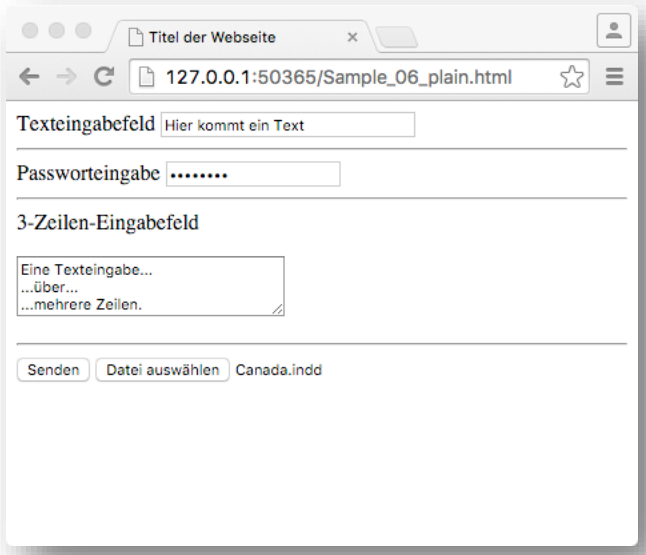
Alle bisher gezeigten Elemente dienen der Strukturierung und Darstellung von Informationen, die von einem Webserver zum Client gesendet werden.

BEIDSEITIGE KOMMUNIKATION

- Bisher ging es um die Darstellung von Inhalten/Informationen (Webserver, unidirektional zum Client).
- Um mit dem Anwender vollständig zu kommunizieren, müssen Informationen vom Client zum Webserver gesendet werden.
- Dafür können Formulare/Auswahllisten eingesetzt werden.
- Scripte sind ebenfalls möglich (später mehr dazu).

FORMULARE

```
<!doctype html>
<html>
 <head>
<meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
  <form action="/php/doit.php" method="post">
    Texteingabefeld
     <input type="text" size="30"><hr>
    Passworteingabe
     <input type="password" size="20"><hr>
    3-Zeilen-Eingabefeld <p>
     <textarea name="text" rows="3"
     cols="30"></textarea><hr>
    <button type="submit">Senden</button>
    <input type="file">
   </form>
 </body>
</html>
```



ATTRIBUTE

Zunächst schauen wir uns eine Übersicht der Attribute an, die im HTML-Element `<form>` eingesetzt werden.

ATTRIBUTE FÜR `<FORM>`

`action`	Angabe einer URL, die beim Absenden des Formulars aufgerufen wird (häufig: Serverscript)
`method`	Definition der HTTP-Request-Methode; bestimmt wie die Daten an den Server gesendet werden (*get*/*post*)
`enctype`	(MIME)-Kodierung der zu versendenden Daten (Standard: *application/x-www-form-url-encoded*)
`accept-charset`	Zeichenkodierung der zu versendenen Daten
`target`	Angabe, wo die Antwort des Server ausgegeben wird (z.B. in einem neuen Fenster - `_blank`)

FORMULAR-ELEMENTE

Als nächstes betrachten wir die einzelnen Elemente, wie sie in einem Formular eingesetzt werden können.

ELEMENTE FÜR <FORM>

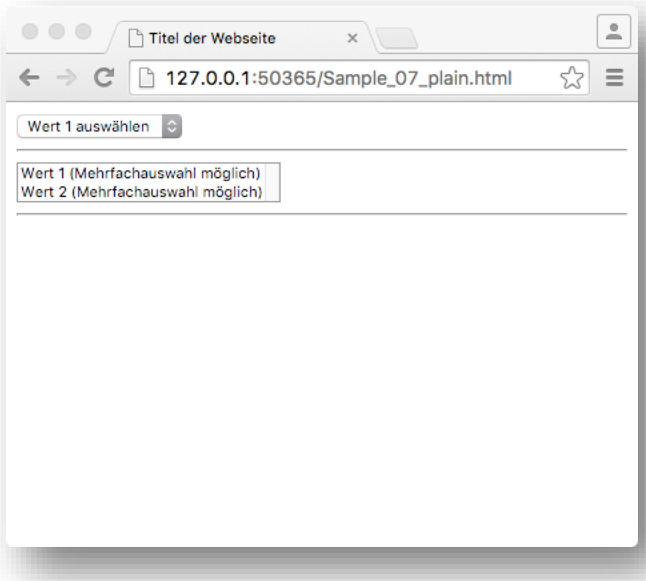
<input type=„text“>	einzeiliges Texteingabefeld
<input type=„password“>	einzeiliges Texteingabefeld mit verdeckter Eingabe
<textarea>	mehrzeiliges Texteingabefeld
<select>	Dropdown-Liste
<option>	Auswahlfeld innerhalb von <select>...</select>
<input type=„radio“>	Radiobuttons
<input type=„checkbox“>	Checkboxen
<input type=“file“>	Datei-Upload
<button type=„submit“>	Schaltfläche

WEITERE HTML-ELEMENTE

Neben Formularen kennen Sie sicher noch weitere Möglichkeiten, wie der Nutzer Informationen an den Server kommunizieren kann. Dabei handelt es sich oft um eine Auswahl aus vorgegebenen Einträgen.

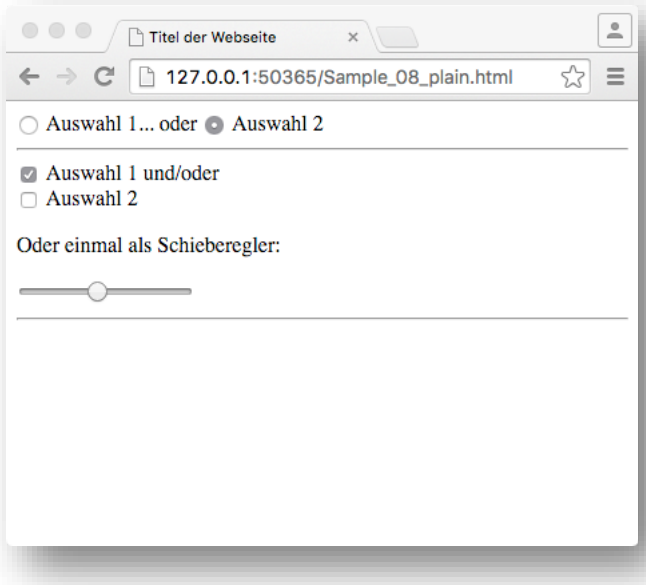
LISTENAUSWAHL

```
<!doctype html>
<html>
 <head>
<meta charset=“UTF-8”>
  <title>Titel der Webseite</title>
 </head>
 <body>
  <form>
   <select>
    <option value="val1">Wert 1 auswählen
    </option>
    <option value="val2">Wert 2 auswählen
    </option>
   </select>
  </form><hr>
  <form>
   <select multiple size="2">
    <option value="val1">Wert 1
     (Mehrfachauswahl möglich)</option>
    <option value="val2">Wert 2
     (Mehrfachauswahl möglich)</option>
   </select>
  </form><hr>
 </body>
</html>
```



WEITERE AUSWAHL-ELEMENTE

```
<!doctype html>
...
<body>
 <form>
  <input type="radio" value="1">
   Auswahl 1... oder
  <input type="radio" value="2">
   Auswahl 2
 </form><hr>
 <form>
  <input type="checkbox" value="1">
   Auswahl 1 und/oder <br>
  <input type="checkbox" value="2">
   Auswahl 2
  <p>Oder einmal als Schieberegler:</p>
  <input type="range" min="1" max="10"
   step="1" value="3">
  </form><hr>
 </body>
</html>
```



FEHLERQUELLE: NUTZER

Eine häufige Fehlerquelle ist der Anwender selbst. Wie soll nochmal das Datum eingegeben werden? Dazu bieten HTML5 und moderne Browser eine „narrensichere“ Unterstützung…

MEHR INPUT MIT HTML5

`<input type=„text“>`	einzeiliges Texteingabefeld
`type=„color“`	Farbauswahl
`type=„date“`	Datumsangabe
`type=„email“`	E-Mailadresse
`type=„month“`	Monatsangabe
`type=„range“`	Zahl in einem bestimmten Bereich
`type=„search“`	Suchfeld
`type=„tel“`	Telefonnummer
`type=„url“`	URL-Adresse
...	...

DER HTTP-REQUEST

Beachten Sie, dass bei der Kommunikation zwischen Client und Server das HTTP-Protokoll zum Einsatz kommt. Sie haben gelernt, dass für jeden HTTP-Request die Methode bestimmt werden muss.

GET UND POST

- Mit einem HTTP-Request stellt der Browser (Client) eine Anfrage an den Webserver (Server). Die Methode bestimmt, wie die Daten vom Client zum Server übertragen werden.
- Standard: GET
  - Daten werden mit einem führenden ? an die URL angehangen
  - Query-String
- POST
  - Daten werden in einem getrennten Bereich des HTTP-Request übertragen
  - für größere Datenmengen besser geeignet

WEBSEITENPROJEKT

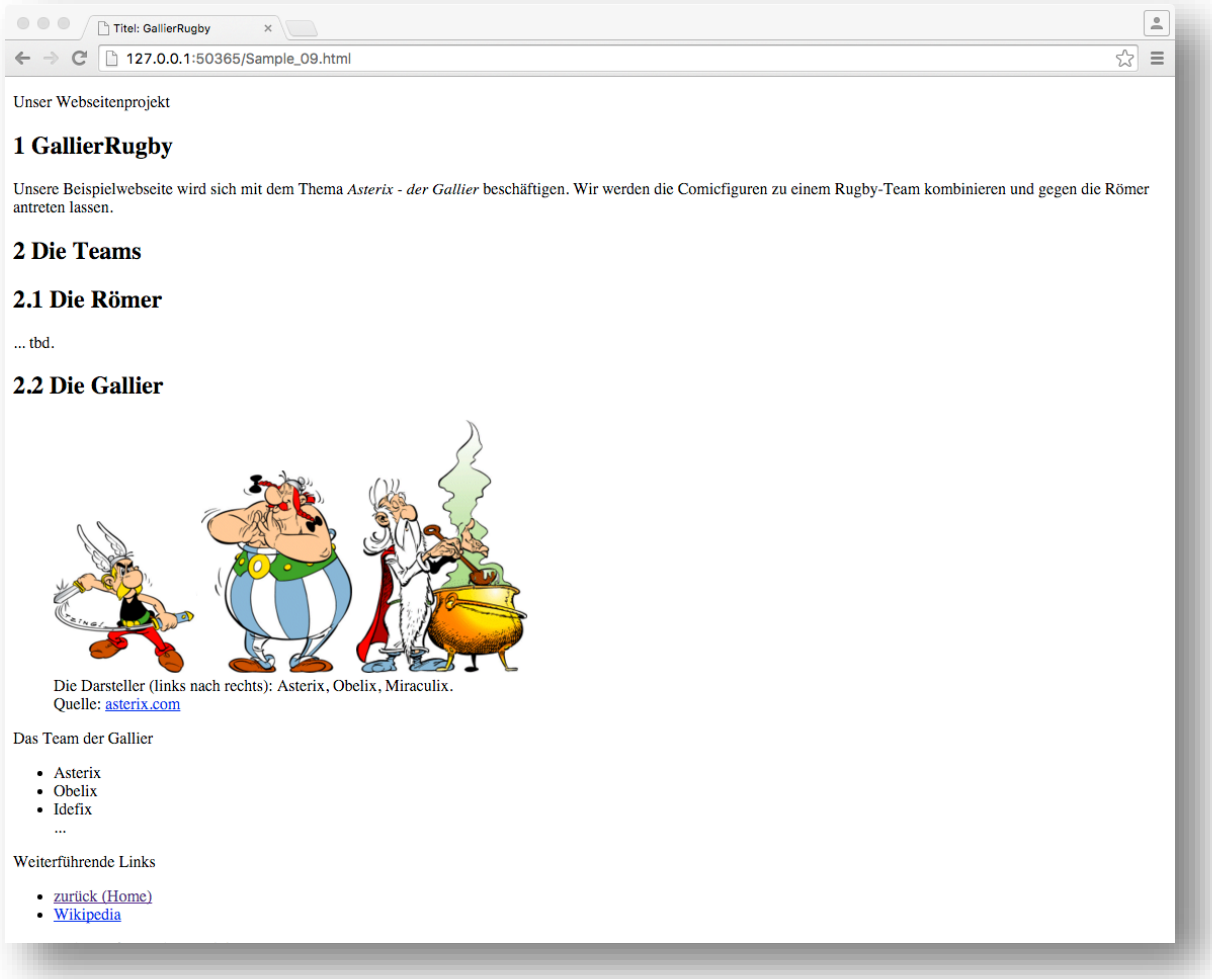
Aber was macht eigentlich unser Webseitenprojekt?

Schauen wir uns das Zwischenergebnis einmal an.

Sieht noch etwas kahl aus, oder? Dann müssen wir uns der Gestaltung widmen → CSS.

UNSER BEISPIEL

Unsere zunächst grob strukturierte Webseite.



ENDE

HTML-EINFÜHRUNG

CSS

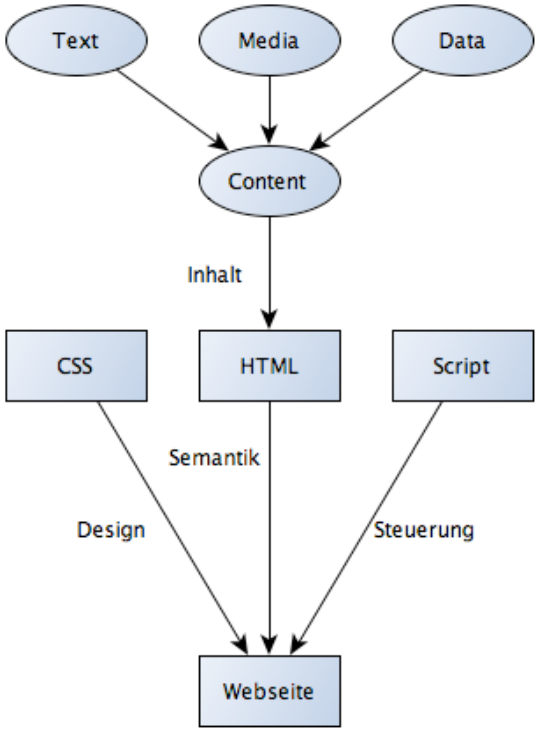
LAYOUT, CSS3

STYLESHEETS

Sie haben mit HTML kennengelernt, wie Webseiten logisch strukturiert und mit Inhalt befüllt werden. Über Stylesheets kann jetzt eine gezielte Steuerung der Darstellung erfolgen. Interaktion werden wir unter dem Stichwort Script näher beleuchten.

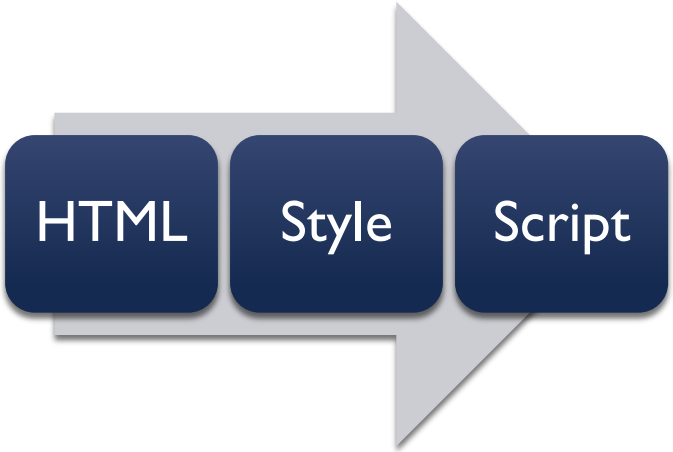
Als vollständige Webseite betrachten wir immer die Summe aus HTML, Stylesheet und Script.

CSS, HTML + SCRIPT = WEBSEITE



MODEL-VIEW-CONTROL

Diese Trennung der Belange kennen wir bereits aus der Programmierung. Obwohl das Prinzip im Kontext von Webseiten nicht immer eingehalten wird, hilft es die Qualität nachhaltig zu steigern. Halten Sie sich nach Möglichkeit daran.



CSS VERWENDEN

- HTML definiert die Bedeutung des Inhalts (Semantik)
- CSS definiert die Darstellung des Inhalts
- CSS wird in ein HTML-Dokument eingebunden
- Die aktuelle Version ist CSS3.
- Zusammen mit HTML5 ist das der aktuelle Stand der Technik.

BEISPIEL

Betrachten wir wieder unser Beispiel aus dem vorherigen Lernabschnitt.

HTML5 OHNE CSS

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Titel der Webseite</title>
 </head>
 <body>
     <header><p>Kopfzeile</p></header>
     <section>
         <h1>Kapitel 1</h1>
         <p>Text</p>
     </section>
     <section>
         <h1>Kapitel 2</h1>
         <section>
             <h2>Kapitel 2.1</h2>
         </section>
     </section>
     <footer><p>Fusszeile</p></footer>
 </body>
</html>
```

BEISPIEL MIT STYLE

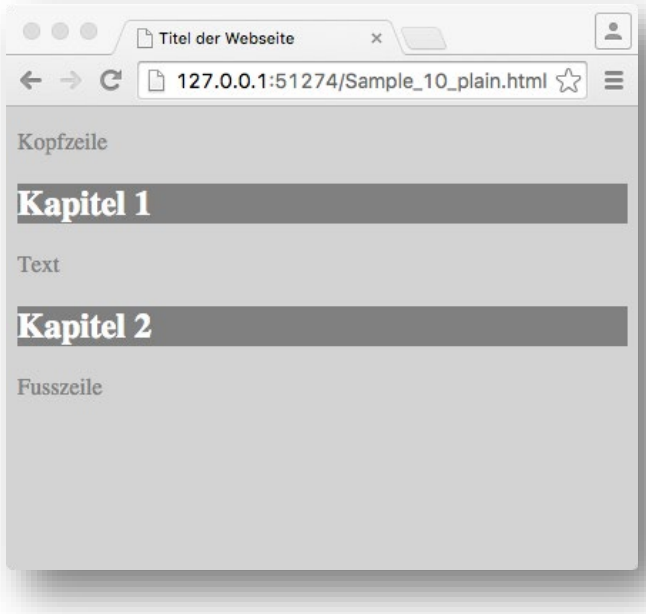
Jetzt wird eine separate Datei mit der Endung *.css erstellt. Darin definieren wir das Aussehen der Webseite. Zu den einzelnen Befehlen kommen wir später und schauen uns zunächst das Ergebnis an.

HTML5 MIT CSS

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Titel der Webseite</title>
  <link rel="stylesheet"
    href="sample_10_plain.css">
 </head>
 <body>
     <header><p>Kopfzeile</p></header>
     <section>
       <h1>Kapitel 1</h1>
       <p>Text</p>
     </section>
     <section>
       <h1>Kapitel 2</h1>
       <p>Text</p>
     </section>
     <footer><p>Fusszeile</p></footer>
 </body>
</html>
```

```
html {
    color: grey;
    background: lightgrey;
}

h1, h2 {
    color: white;
    background: grey;
}
/* sample_10_plain.css */
```



ERLÄUTERUNG ZUM BSP

Wie Sie sehen, wird der Style im Header über das Tag `<link>` eingebunden. Die Box zeigt den Inhalt des Stylesheets, der Vorder- und Hintergrundfarbe der Webseite definiert (grau, hellgrau).

STYLE JE ELEMENT

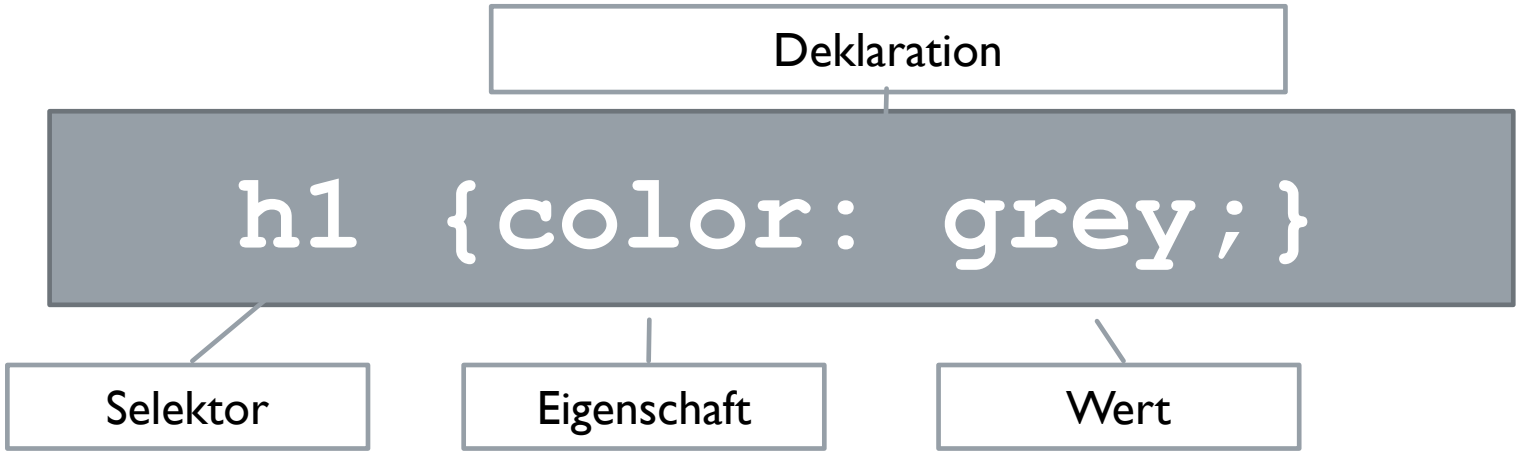
Sie sehen am Beispiel, dass im Stylesheet offenbar einzelne HTML-Elemente *selektiert* werden und für diese das Aussehen gezielt definiert wird.

Das ist das Grundprinzip von CSS. Jede Anweisung in einem Stylesheet bezieht sich auf ein Element im HTML-Dokument. Schauen wir uns die Syntax an...

CSS - ANWEISUNGEN

Der ***Selektor*** bestimmt das HTML-Element, auf das das Design angewendet wird.

Die ***Deklaration*** bestimmt ***Werte*** für einzelne ***Eigenschaften***.



```
html {
    color: grey;
    background: lightgrey;
}

h1, h2 {
    color: white;
    background: grey;
}
/* sample_10_plain.css */
```

1. deklariert Schriftfarbe „grau“ und Hintergrund „hellgrau“ für das gesamte Dokument
2. deklariert Schriftfarbe „weiß“ und Hintergrund „grau“ für Überschriften-Elemente

SELEKTION

Prägen Sie sich ein, wie die Selektion eines HTML-Elements erfolgt. Wir lernen später noch Möglichkeiten der Kombination von Selektoren kennen.

EINBINDUNG IN HTML

- Die Einbindung erfolgt über das bereits bekannte Tag `link` im Header des HTML-Dokuments.

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

- Alternativen sind:
  - als CSS direkt im HTML-Tag
  - als CSS im Header des HTML-Dokuments
- Beide Alternativen sind sorgfältig zu wählen und nur bei Bedarf einzusetzen. Beispielsweise, wenn der Style exakt nur einmal verwendet wird.

STYLES FÜR UNTERSCHIEDLICHE MEDIEN

- Es macht durchaus Sinn, unterschiedliche Stylesheets anzulegen und dem Anwender entsprechend bereitzustellen.
- Dazu werden medienspezifische Stylesheets eingesetzt:

```
<link rel=„stylesheet“ media=„screen and (min-width: 1080px)“ href=„style1080.css“>
```

- Das wäre die Auswahl für ein Anzeigegerät mit FullHD.

Mehr dazu später bei *Responsive Design*.

SELEKTOREN

Die Selektion kann nicht nur über die bekannten HTML-Tags erfolgen. Es sind auch selbstdefinierte IDs und Klassennamen möglich.

CSS SELEKTOREN

Selektor	Bezeichnung	Auswahl	Beispiel
`element {}`	Typselektor	HTML-Element mit dem Namen `element`	`<element>`
`.kname`	Klassenselektor	Elemente mit der Klasse `kname`	`<p class=„kname“>`
`#elemid`	ID-Selektor	Elemente mit der ID `elemid`	`<p id=„elemid“>`
`*`	Universal-selektor	Jedes Element	`<p>`

Der Klassenselektor kann für mehrere Elemente angewendet werden.
Der ID-Selektor gilt immer nur für ein Element.

DER UNIVERSALSELEKTOR

Der Universalselektor * sieht zunächst etwas unnötig aus. Seine Bedeutung wird in Kombination mit anderen Selektoren deutlich: z.B. „*selektiere alle Elemente in einer bestimmten Sektion*“.

BEISPIEL

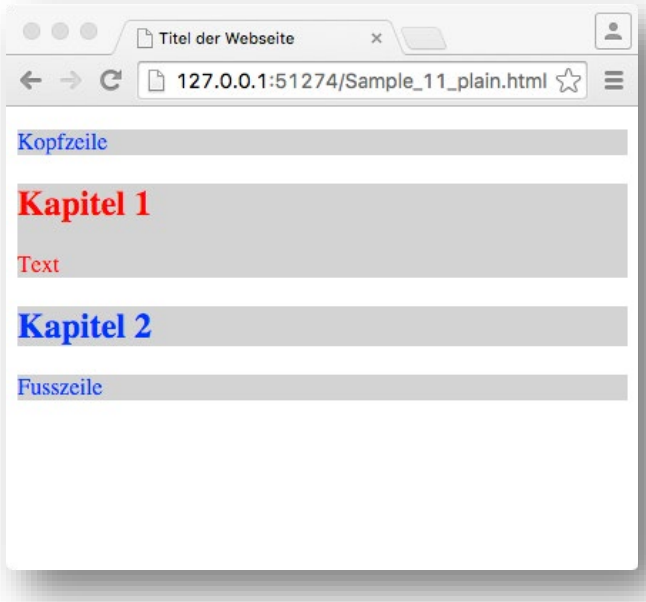
Wir bestrachten wieder unser Beispiel und fügen zwei Klassen für die farbliche Gestaltung ein. Beachten Sie dabei, wie den HTML-Elementen ein Attribut hinzugefügt wird, welches die Klassenzugehörigkeit festlegt.

VERWENDUNG DES KLASSENSELEKTORS

```
<body>
 <header class="blueclass">
  <p>Kopfzeile</p></header>
 <section class="redclass">
  <h1>Kapitel 1</h1>
  <p>Text</p>
 </section>
 <section class="blueclass">
  <h1>Kapitel 2</h1>
 </section>
 <footer class="redclass">
  <p>Fusszeile</p>
 </footer>
</body>
```

```
.redclass {
    color: red;
    background: lightgrey;
}

.blueclass {
    color: blue;
    background: lightgrey;
}
```



BEISPIEL

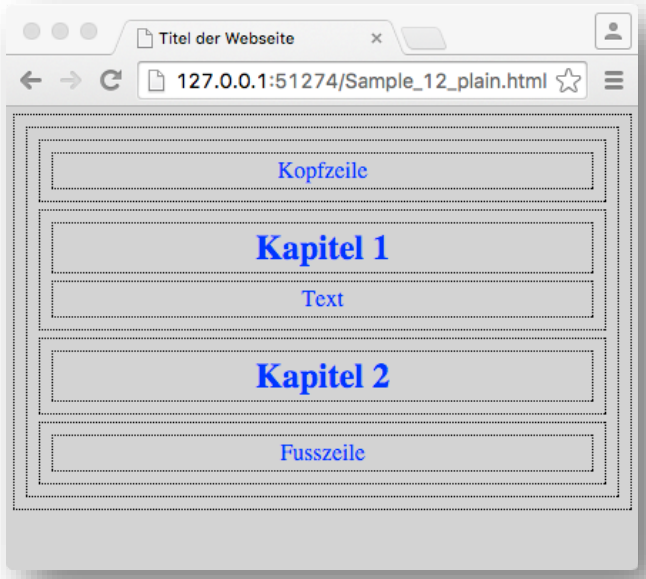
Im folgenden Besipiel setzen wir den Universalselektor ein und geben jedem HTML-Element einen Rahmen. Sie sehen hier sehr schön, wie der Browser alle Elemente als Box rendert.

Tipp: Vergleichen Sie die Boxen einmal mit dem DOM.

VERWENDUNG DES UNIVERSALSELEKTORS

```
<body>
 <header>
  <p>Kopfzeile</p></header>
 <section>
  <h1>Kapitel 1</h1>
  <p>Text</p>
 </section>
 <section>
  <h1>Kapitel 2</h1>
 </section>
 <footer>
  <p>Fusszeile</p>
 </footer>
</body>
```

```
* {
    color: blue;
    background: lightgrey;
    margin: 5px;
    padding: 3px;
    border: 1px dotted black;
    text-align: center;
}
```



PSEUDOKLASSEN

Neben eigenen Klassen liefert CSS einige vordefinierte Pseudoklassen. Das sind Klassen, die nicht direkt einem HTML-Element zugeordnet sind.

CSS PSEUDOKLASSEN

- Mit Pseudoklassen bietet CSS eine Möglichkeit, Darstellung und Verhalten (...des Anwenders) zu verbinden.
- Der Einsatz muss sorgfältig abgewägt werden!
- Pseudoklassen haben vordefinierte Namen, z.B. `:hover`.
- Im folgenden Beispiel erhalten alle Elemente, über die der Mauszeiger steht, die Schriftfarbe rot und Hintergrund grau.

```
*:hover{
    color: red;
    background: lightgrey;
}
```

CSS PSEUDOKLASSEN

Es gibt noch weitere Pseudoklassen:

- Hyperlinks (`visited, link`)
- Dokumentenstruktur (`root, first-child`)
- Sprache (`lang(de)`)
- Negations-Pseudoklasse (`not(p)`)

Setzen Sie Pseudoklassen nur dann ein, wenn es einen expliziten Grund dafür gibt. Oft bietet sich jedoch eine bessere Lösung an.

SELEKTIONEN KOMBINIEREN

Wir hatten es bereits angesprochen. Selektoren können beinahe beliebig miteinander kombiniert werden. Somit sind Gruppen von Elementen auf einer Webseite adressierbar. Nutzen Sie dafür die Semantik von HTML.

CSS KOMBINATOREN

Kombinator	Bezeichnung	Bedeutung
E F	Nachfahrenselektor* (*decendant*)	F wird selektiert, wenn es ein Nachfahre eines E-Elements ist
E > F	Kindselektor (*child*)	F wird nur selektiert, wenn es Kind eines E-Elements ist
E + F	Nachbarnselektor (*adjacent sibling*)	F wird nur selektiert, wenn es direkt nach (neben) E vorkommt (gleiche Eltern)
E ~ F	Geschwisterselektor (*general sibling*)	F wird nur selektiert, wenn es nach E vorkommt (gleiche Eltern)

** Die Selektoren beziehen sich auf das DOM.*

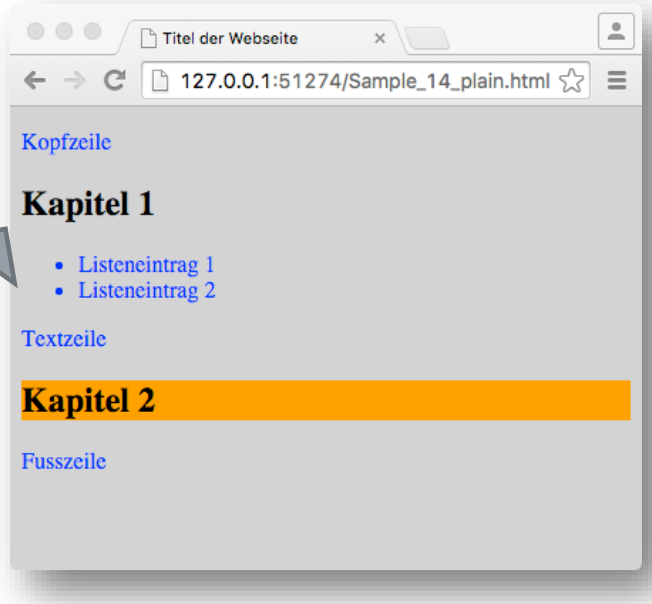
VERERBUNG

- Vererbung erleichtert die Entwicklung von Webseiten.
- Die Vererbung bezieht sich wieder auf das DOM (Baum).
- Es gibt eine automatische und eine erzwungene Vererbung.
  - Vererbung erzwingen: `section * {background: inherit;}`
- Im folgenden Beispiel *erben* die Kinder des `section`-Elements die Farbgestaltung.

VERERBUNG

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Titel der Webseite</title>
  <link rel="stylesheet"
    href="Sample_14_plain.css">
 </head>
 <body>
     <header><p>Kopfzeile</p></header>
     <section id="section1">
       <h1>Kapitel 1</h1>
       <ul>
         <li>Listeneintrag 1</li>
         <li>Listeneintrag 2</li>
       </ul>
       <p>Textzeile</p>
     </section>
     <section>
       <h1>Kapitel 2</h1>
     </section>
     <footer><p>Fusszeile</p>
     </footer>
 </body>
</html>
```

```
html {
    color: blue;
    background: lightgrey;
}
h1 {
    color: black;
    background: orange;
}
#section1 h1 {background: inherit;}
```



VERERBUNG

Im vorigen Beispiel wird zuerst der Hintergrund auf grau gesetzt (für `<html>`). Danach werden die Überschriften orange gefärbt. Ohne die Anweisung in der letzten Zeile würden jetzt in beiden Sektionen orange-farbige Balken angezeigt.

In der letzten Zeile jedoch *erbt* die Sektion den Hintergrund und somit die Farbe vom Elternelement. Sie bleibt grau.

MEHRERE STYLESHEETS

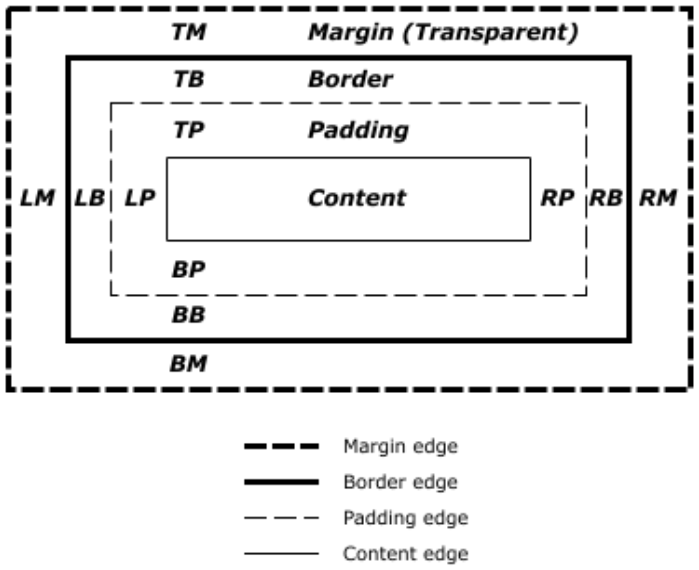
Das Prinzip der Mehrfachverwendung wird auch auf komplette Stylesheets angewendet. Es findet eine Kombination bzw. Kaskadierung statt – daher auch der Name *Cascading Style Sheets*.

KASKADE

- CSS soll durch Wieder- bzw. Mehrfachverwendung den Entwicklungsaufwand reduzieren und die Fehlerquote veringern.
- Stylesheets können mehrfach verwendet werden, HTML-Dokumente können aber auch mehrere Sytesheets einbinden und entsprechend kombinieren.
- CSS definiert Prioriäten, wenn es dabei zu Konflikten kommt.

Siehe: http://www.w3.org/TR/css3-cascade/

- HTML-Elemente werden vom Browser in Boxen dargestellt (Rechteck).
- Alle Boxen werden innerhalb des Browser-Fensters arrangiert.
- Boxen werden geschachtelt (entsprechend DOM).



Quelle: http://www.w3.org/TR/CSS21/box.html, 18.09.2015

BOXEN GESTALTEN

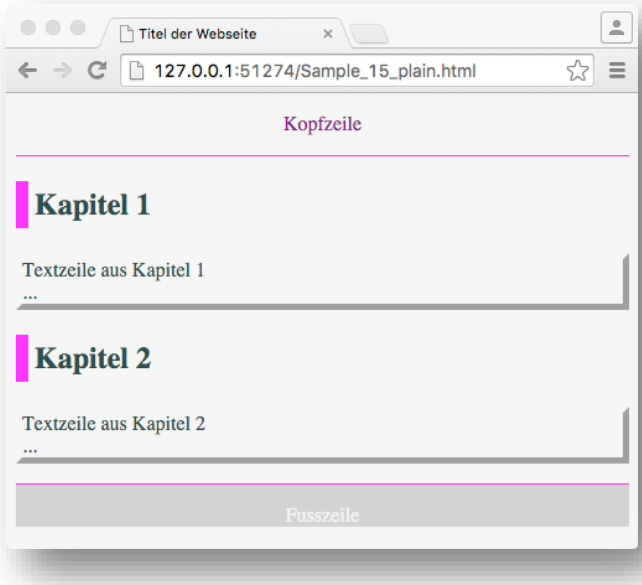
- Rahmen
- Abstände
- Hintergrund (Farbe, Transparenz)
- Schatten

RAHMEN SETZEN

Nehmen wir wieder unser Anfangsbeispiel. Hier gibt es einen Header, Footer und Textbereiche. Mittels eines Stylesheets geben wir dem tristen Aussehen etwas mehr Farbe und Dynamik, indem wir lediglich die Boxen gestalten.

BOXEN GESTALTEN

```
<!doctype html>
<html>
html {
    color: darkslategray;
    background: whitesmoke;
}
header{
    color: darkmagenta;
    text-align: center;
    border-bottom: 1px solid magenta;
}
h1 {
    border-left: 10px solid magenta;
    background: whitesmoke;
    padding: 5px;
}
h1 + p {
    border: 5px outset whitesmoke;
}/*Der Text nach h1 wird selektiert*/
footer {
    text-align: center;
    color: whitesmoke;
    background: lightgrey;
    border-top: 1px solid magenta;
}
```



POSITION

Nachdem die Boxen Farbe gewonnen haben, wollen wir sie noch zurechtrücken. Dazu können Boxen positioniert werden. CSS unterscheidet dabei 4 mögliche Positionierungsarten...

BOXEN POSITIONIEREN

`position: static;`	Standardeinstellung, alle Elemente werden nacheinander im Fluss (*floating*) angeordnet – entsprechend HTML-Dokument
`position: relative;`	Element wird relativ zur Standardposition positioniert (top, bottom, left, right). Nachfolgende Elemente bleiben auf ihrer Position.
`position: absolute;`	Element wird *freischwebend* an eine neue Position gestellt. Nachfolgende Elemente *rücken nach*.
`position: fixed;`	Element verhält sich analog zu `absolute`, wird aber aus dem Fluss herausgelöst. Bezugspunkt ist hier die linke obere Ecke des Fensters.

POSITIONIERUNGSARTEN

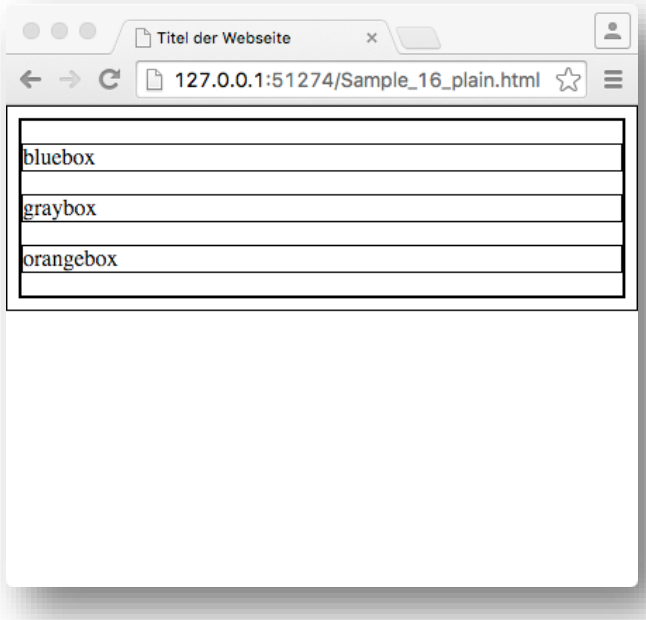
Nun setzen wir die Positionierung ein, indem wir die Art der Positionierung in der CSS-Anweisung festlegen.

Um das Verhalten der Boxen vergleichen zu können, beginnen wir mit einem einfachen Beispiel.

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Titel der Webseite</title>
  <link rel="stylesheet"
    href="Sample_16_plain">
 </head>
 <body>
    <article>
     <p class="bluebox">bluebox</p>
     <p class="magbox">magbox</p>
     <p class="graybox">graybox</p>
     <p class="orangebox">orangebox</p>
    </article>
 </body>
</html>
```

```
* {
    border: 1px solid black;
}
```

HTML-CODE FÜR DIE FOLGENDEN BEISPIELE

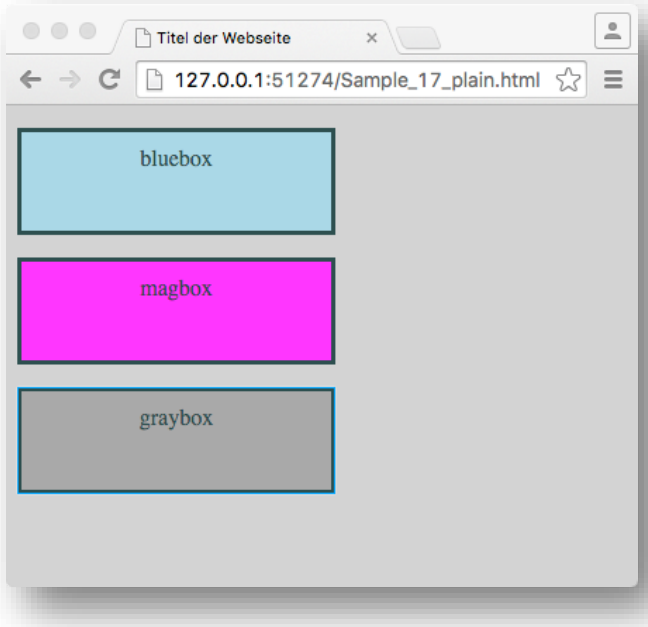


STATISCHE POSITION

Die statische Positionierung entspricht dem bekannten Verhalten. Die drei Boxen werden untereinander dargestellt – passend für das schmale Fenster.

```
html {
    color: darkslategray;
    background: lightgrey;
}
p {
    border: 3px solid;
    width: 200px;
    height: 50px;
    text-align: center;
    padding: 10px;
}
.bluebox {
    background: lightblue;
}
.graybox {
    background: gray;
}
.magbox {
    background: magenta;
}
```

POSITION: STATIC

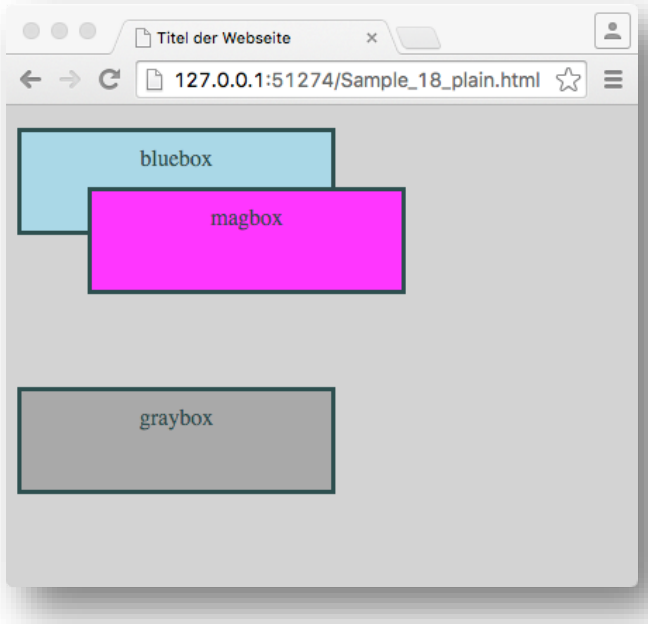


RELATIVE POSITION

In diesem Beispiel wurde die zweite Box relativ zu ihrer ursprünglichen Position verschoben. Beachten Sie, dass die dritte Box nicht „nachrückt“.

```
...
p {
    border: 3px solid;
    width: 200px;
    height: 50px;
    text-align: center;
    padding: 10px;
}
.bluebox {
    background: lightblue;
}
.magbox {
    background: magenta;
    position: relative;
    top: -50px;
    left: +50px;
}
.graybox {
    background: darkgray;
}
```

POSITION: RELATIVE



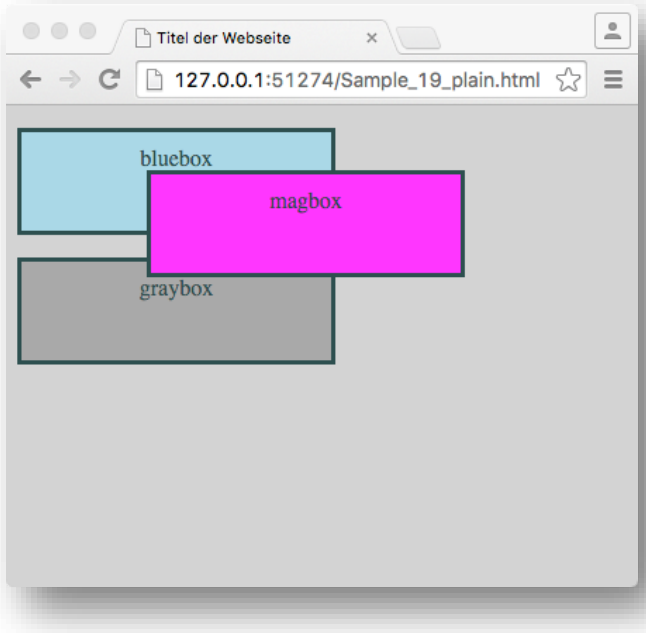
ABSOLUTE POSITION

Jetzt bekommt die zweite Box eine absolute Position im Fenster, wobei sie den Fluss verlässt. Sie beobachten, dass die dritte Box jetzt „nachrückt“ und die Position der zweiten Box im Fluss einnimmt. Dabei kommt es zu einer Überlagerung.

Bei *absolute* würde diese Position jedoch mit Scrollen entsprechend „mitbewegt“ werden. Letzteres unterscheidet sie von *fixed*, wo die Box immer am selben Bereich im Fenster angezeigt wird.

```
...
p {
    border: 3px solid;
    width: 200px;
    height: 50px;
    text-align: center;
    padding: 10px;
}
.bluebox {
    background: lightblue;
}
.magbox {
    background: magenta;
    position: absolute;
    top: 30px;
    left: 100px;
}
.graybox {
    background: darkgray;
}
```

POSITION: ABSOLUTE



* graybox *rückt nach*

Z-INDEX (DIE 3. DIMENSION)

- Wir haben gesehen, dass sich Boxen überlagern können.
- Mit `z-index` wird die 3. Dimension definiert, d.h. die Position der Box auf einer dritten Achse.
- Der `z-index` funktioniert nicht bei `position: static`.

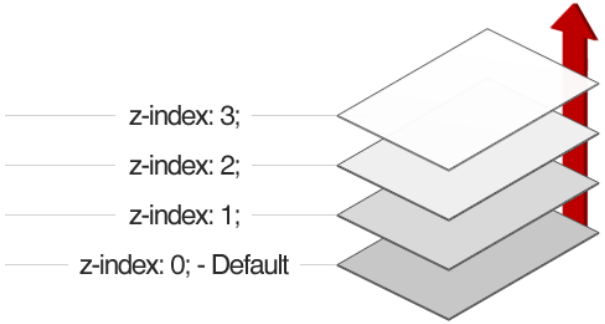
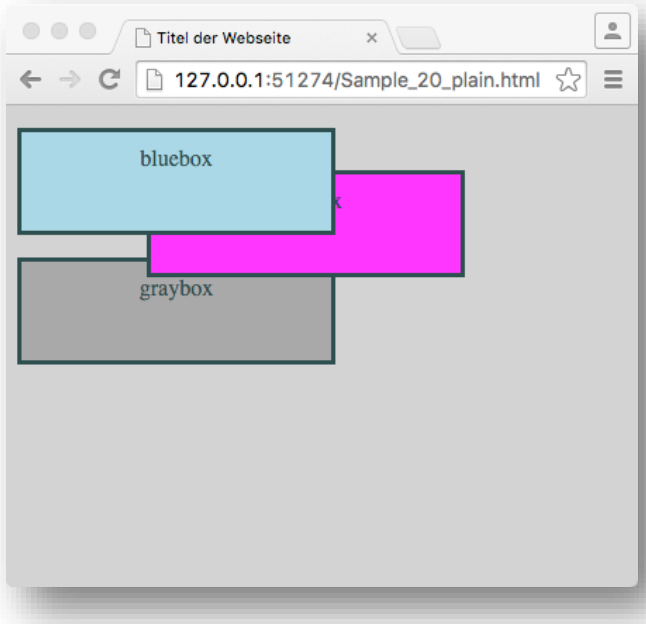


Bild Quelle: http://www.annyhe.com/zindex/img/basic-z-index.gif

```
...
p {
    border: 3px solid;
    width: 200px;
    height: 50px;
    text-align: center;
    padding: 10px;
}
.bluebox {
    background: lightblue;
    position: relative;
    z-index: 2;
}
.magbox {
    background: magenta;
    position: absolute;
    top: 30px;
    left: 100px;
    z-index: 1;
}
.graybox {
    background: darkgray;
}
```

Z-INDEX

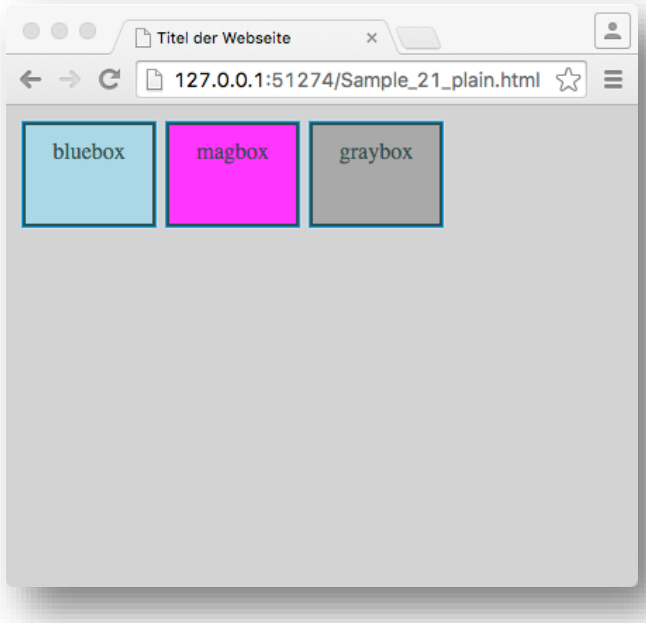


FLIEßTEXT

- Fließtext kennen wir bereits von MS Word bzw. LaTeX.
- Ein Element wird *schwebend* platziert und nachfolgende Elemente *umfließen* es.
- Die CSS-Eigenschaft lautet: `float`.

```
...
p {
    border: 3px solid;
    width: 70px;
    height: 50px;
    text-align: center;
    padding: 10px;
    margin: 3px;
}
.bluebox {
    background: lightblue;
    float: left;
}
.magbox {
    background: magenta;
    float: left;
}
.graybox {
    background: darkgray;
    float: left;
}
```

FLOAT

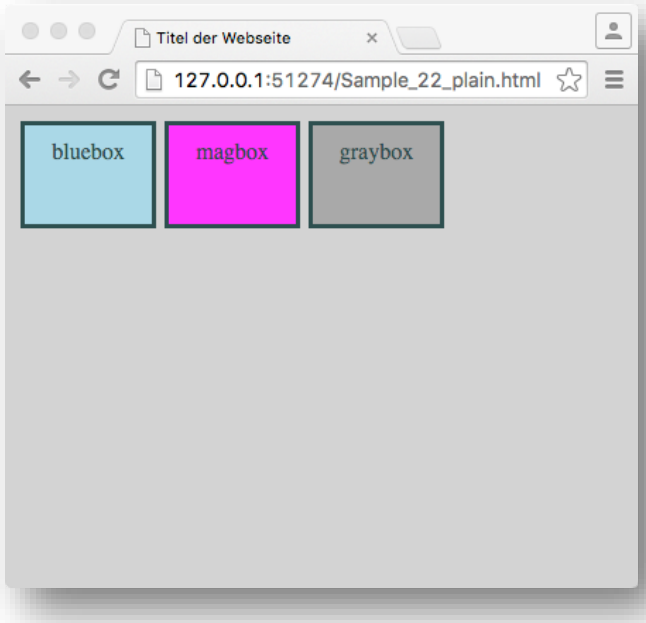


FLEXBOXEN

- Flexboxen machen die Gestaltung von Fließelementen einfacher. HTML-Elemente können als Flexbox definiert werden und dienen somit als Container für darin liegende Elemente.
- Die Positionierung der Boxen (floating) wird dabei einmalig im Elternelement bestimmt und angewendet.
- Probieren wir es aus: Flexbox im Element `article`

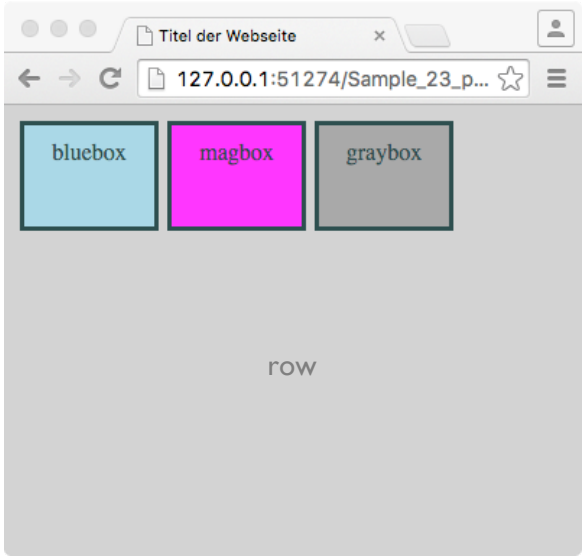
```
...
p {
    border: 3px solid;
    width: 70px;
    height: 50px;
    text-align: center;
    padding: 10px;
    margin: 3px;
}
article {
    display: flex;
    flex-direction: row;
}
.bluebox {
    background: lightblue;
}
.magbox {
    background: magenta;
}
.graybox {
    background: darkgray;
}
```

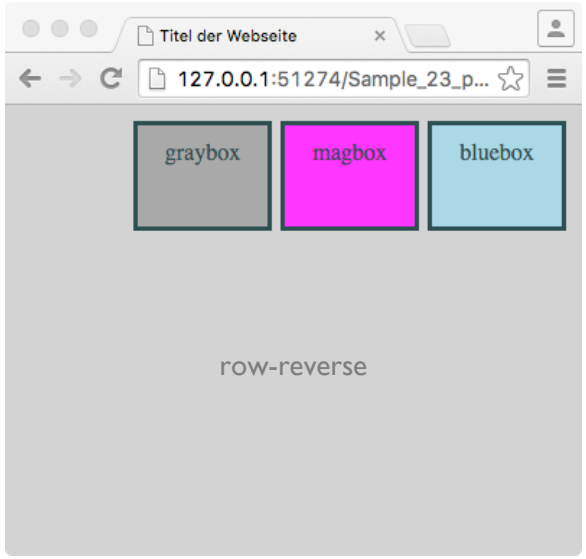
FLEXBOX

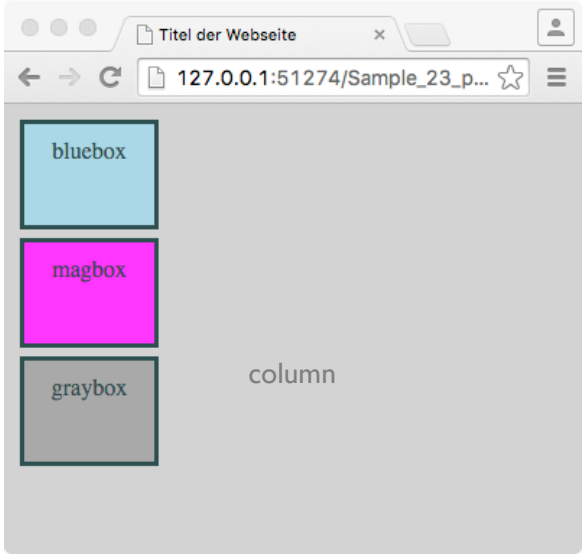


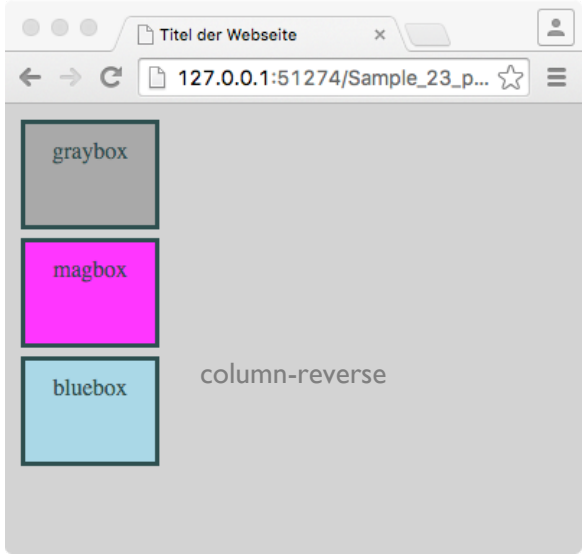
FLEXBOXEN NUTZEN

Die Reihenfolge kann nun sehr einfach über die Flexbox gesteuert werden. Im folgenden Beispiel schauen wir uns einige Parameter an.


Titel der Webseite
127.0.0.1:51274/Sample_23_p...
bluebox
magbox
graybox
row


Titel der Webseite
127.0.0.1:51274/Sample_23_p...
graybox
magbox
bluebox
row-reverse


Titel der Webseite
127.0.0.1:51274/Sample_23_p...
bluebox
magbox
graybox
column


Titel der Webseite
127.0.0.1:51274/Sample_23_p...
graybox
magbox
bluebox
column-reverse

ENDE

NÄCHSTER TEIL: ÜBUNGSAUFGABEN

AUFGABE 1

- Öffnen Sie die Einführung des HTML-Editors Brackets
  - index.html
- Schreiben Sie das Dokument in LaTeX nach
  - verwenden Sie nach Möglichkeit die gleiche Semantik
- Erstellen Sie eine Tabelle mit einem Vergleich
  - Mit welchen Elementen/Tags wird welche Semantik beschrieben?
  - Was sind die Unterschiede zwischen den drei verschiedenen Formaten?
  - Die Tabelle bezieht sich auf das erstellte Dokument.
- Erstellen Sie jeweils eine Druckausgabe im PDF-Format