

AGENDA

Curriculum WDD

4100_07	REPETITION / SPRACHEN / STATISCHE WEBSITE / SYNTAX / GRUNDGERÜST / TAGS AUFGABE / QUELLCODE und DOM / WEB INSPECTOR / SEMANTIK
	CSS REGEL / CSS EINBINDEN / CSS RESET / CSS MEDIA TYPES / CSS SELEKTOREN + SPEZIFITÄT / MASSEINHEITEN
4100_08	FARBCODES / INLINE- UND BLOCKELEMENTE / FLOAT
	POSITIONIERUNG / BOX-MODEL
4100_09	NAVIGATION / FARBVERLÄUFE / FONTS / TEXT FORMATING / ICON FONT
	BILDER / HINTERGRUND / CSS SPRITES / FAVICON
4100_10	EINMITTEN / TABELLEN / IFRAME / FORMULAR
	VIDEO/AUDIO / RETINA / BARRIEREFREIHEIT
4100_11	REKAPITULIEREN CSS SELEKTOREN / SUBNAVIGATION / ZWISCHENTEST
	HELPER TOOLS / WORKFLOW

WIE LERNE ICH EFFIZIENT?

Kurzgesagt

- Lege chronologische resp. thematische Ordner auf Deinem Rechner an und lade das SAE-Übungsmaterial kurz vor Unterrichtsstart jeweils auf Deinen Rechner.
- Verwende ein elektronisches Lernjournal (z.B. Evernote, Docs), in welchem Du auch nach Stichworten suchen kannst.
- Über grundlegende Englisch-Kenntnisse musst Du verfügen.
- 10-Finger-Tastaturschreiben ist vorteilhaft.
- Denke an die Kontaktzeit von 25-30h/Woche.
Nur durch zusätzliches Üben und Lernen wirst Du reüssieren.

Repetition

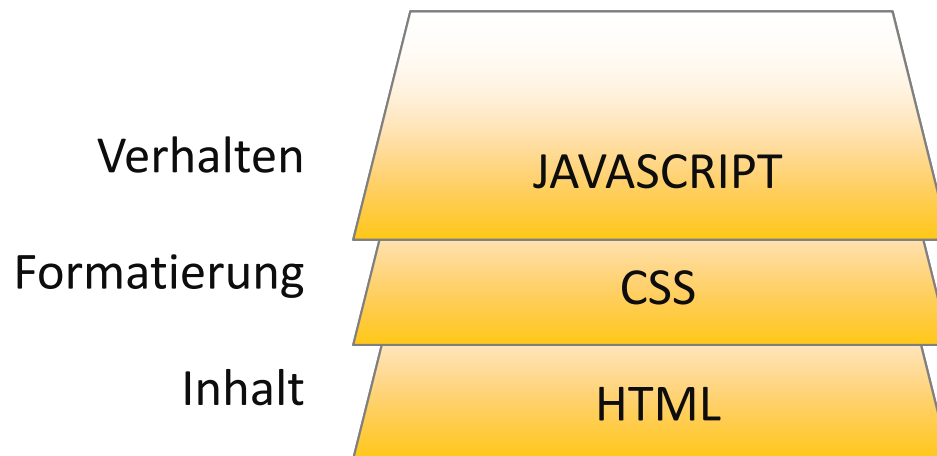
DER BROWSER KANN 3 SPRACHEN

Der sprachliche Aufbau einer Webpage client-seitig

Eine Webpage besteht aus 3 logischen Ebenen.

Jede Ebene braucht ihre eigene Sprache.

Der Server schickt Quellcode mit HTML, CSS und Javascript Code-Teile an den Browser, der diese interpretiert (englisch: Rendering).



Reihenfolge des
Interpretierens
im Browser



SPRACHEN

WIR UNTERSCHIEDEN

Auszeichnungssprache vs. Programmiersprache

Auszeichnungssprache (Markup)	Programmiersprache
<ul style="list-style-type: none"> • strukturiert und formatiert Inhalte 	<ul style="list-style-type: none"> • trifft Entscheidungen nach bestimmten Bedingungen • funktioniert nach folgendem Prinzip: Eingabe, Verarbeitung, Ausgabe
<p>HTML (Inhalte)</p> <p>CSS (Gestaltung)</p>	<p>JavaScript:</p> <p>(clientseitige Scriptsprache für: Verhalten, Ereignisauslöser, Steuerung von Browserfenster, Rechnen, Teile einer Webpage nachladen, Cookie verwalten, Browserverlauf, URL Manipulation, Effekte, Animationen, DOM-Manipulationen von HTML+CSS)</p>

STATISCHE WEBSITE

ERLÄUTERUNG

Eine statische Website ...

- ist zustandslos (kein Gedächtnis, kein Erinnern von Requests)
- ist hardcodiert (Inhalte sind in den HTML-Files abgelegt)
- hat einzelne Webpages, die durch Navigieren im Browser aufgerufen werden können
- jede Webpage kann durch die gleiche zentrale CSS-Datei formatiert werden
- benötigt nicht PHP, da ihre Inhalte NICHT durch eine Datenbank gespiesen wird:
sie ist somit NICHT dynamisch
- das dauerhafte Abspeichern erfolgt clientseitig:
Cookies, JS, Sessions, Local Storage
- hat eine Start-Datei: index.html
- ist nur mit HTML/CSS-Kenntnissen verwaltbar


SYNTAX VON HTML

SYNTAX

Start- und Endtag / Attribute / Wohlgeformter Code

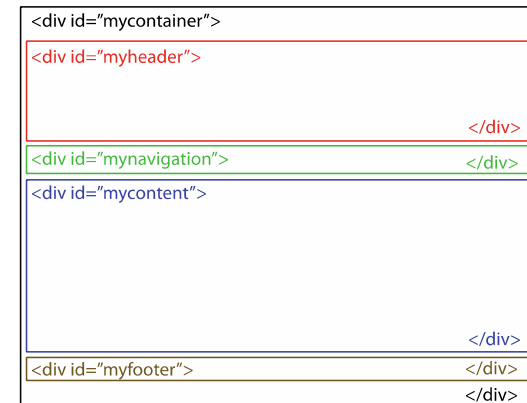
Ein Starttag mit zwei Attributen, Texteingabe, sowie Endtag

`<p id="mycontainer" class="fließtext" >Ich bin Fließtext im Paragraph. </p>`



Unter wohlgeformt versteht man das konsequente Abschliessen und Verschachteln von Tags

```
<section>
  <p>This is a paragraph.</p>
  <p>This is a paragraph.</p>
</section>
```



```
<div id="mycontainer">
  <div id="myheader">
  </div>
  <div id="mycontent">
  </div>
  <div id="myfooter">
  </div>
</div>
```

HTML-ELEMENTE – EINE FAMILIE

Eltern, Kinder, Geschwister, Nachbarn, Nachfahren

<p>Ich bin Text

**** und zeitweise etwas schräg****

im Normalfall normal

****dann wieder schräg****

und ganz Ende bin

****immer

****oft****normal.

</p>

→ ist Mutter von **em**, **em**, **del**

→ ist Nachfahre von **p**

→ ist Kind von Eltern(**p**)

→ ist vorausgehender Nachbar von **em**

→ ist Nachfahre von **p**

→ ist Kind von Eltern(**p**)

→ ist Geschwister von **em**

→ vorausgehender Nachbar von **del**

→ ist Nachfahre von **p**

→ ist Kind von Eltern(**p**),

→ ist Geschwister von **em**

→ ist Mutter von **em**

→ ist Nachfahre von **p**

→ ist Kind von Eltern(**del**)

- Kind: damit sind Elemente der nächsten Einrückungsebene gemeint
- Geschwister/Nachbarn: auf der gleichen Einrückungsebene
- Nachfahren: alle Elemente aller Einrückungsebenen

GRUNDGERÜST

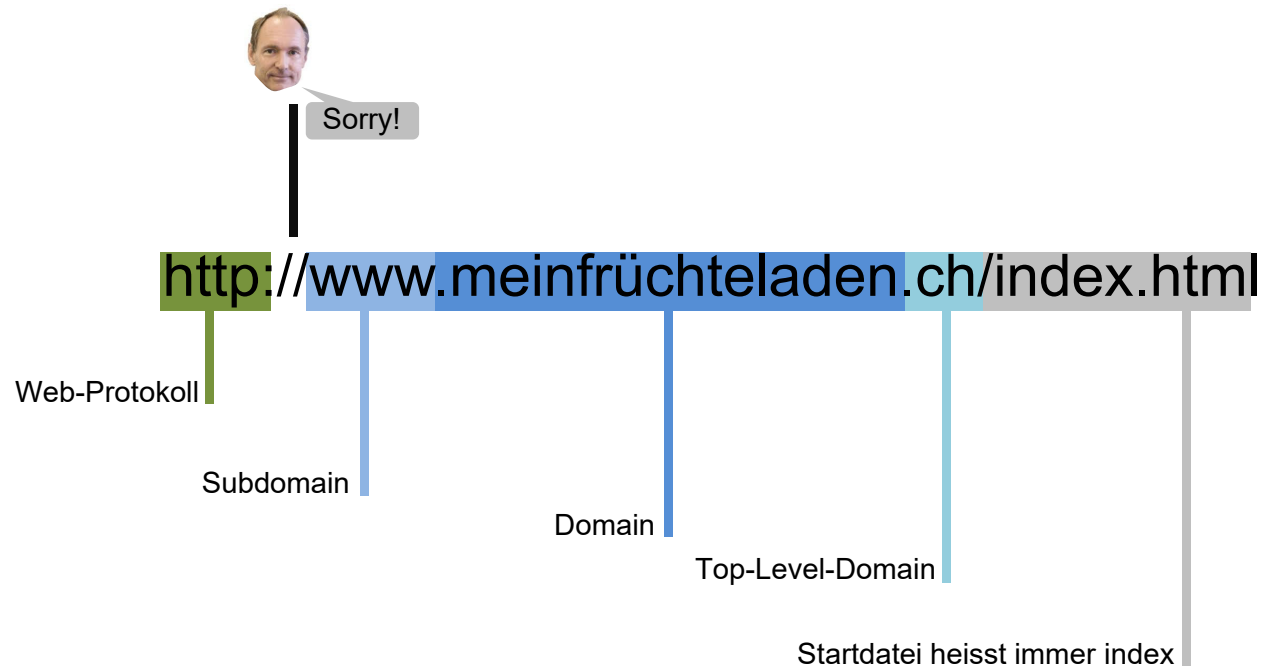
EINE HTML-DATEI

Aufbau (z.B. index.html)

eröffnendes Tag	<!DOCTYPE html>	
	<html lang="de-CH">	
suchmaschinenrelevant für Reiter im Browser für Formatierung für Verhalten	<pre> <head> <meta name="description" content="Die berühmte Maus"> <title>Titel für den Reiter im Browser</title> <link href="css/meine_stile.css" rel="stylesheet" type="text/css"></link> <script type="text/javascript">hier JS </script> </head> </pre>	nicht sichtbar
Division = Platzhalter Linkaufruf bei Klick	<pre> <body> <div id="meine_navi"> Home
 Galerie </div> <div id="inhalt"> <h1>Über mich</h1> <p>Bin Fliesstext im Abschnitt.</p> </div> </body> </pre>	sichtbar
Heading 1 = Überschrift Paragraph		
schliessendes Tag	</html>	

DER AUFRUF DER STARTDATEI

index.html wird standardmässig aufgerufen



DEIN PROJEKT

im Code Editor

Grundgerüst index.html anlegen

1. Überprüfe, ob eine bequeme Entwicklungsumgebung eingerichtet hast (siehe vorheriges Kapitel)
2. Definiere einen Projektordner und die wichtigsten Ordner html / css / js / bilder / _arbeitsdateien
3. Öffne das Projekt in Brackets
4. Lege ein HTML-Gerüst im index.html an. Doctype.
5. Definiere die wichtigsten Tags im Head (Meta-Tags / Title / Favicon) und wichtige, semantische HTML5-Tags und fülle ein wenig Blindtext ab.
6. Überprüfe in der Live-Vorschau
7. Binde ein CSS-Reset ein und definiere, ob das Box-Model border-box zur Anwendung kommt.
8. Definiere ein zentrales CSS-Stylesheet, welches dann aufgesplittet wird in desktop / mobile / print.
9. Baue kleiner JavaScript-Libraries ein, die eine optimale Ausgangslage schaffen:
 1. Modernizr.js
 2. Mit Conditional Format für IE 8: HTML5 Tag-Kompatibilitätstool für ältere Browser: <https://github.com/aFarkas/html5shiv>
 3. Evtl. Prefix Free Framework (wenn ohne @import arbeitend)
10. Überlege dir anhand des Screendesigns, welche semantischen Tags zum Zug kommen, was mit wem verschachtelt wird und wie die Kästenhöhen sich bei überschüssigem Inhalt verhalten sollen (mitwachsen oder overflow).
11. Nimm allenfalls das Browser-Plugin Pixel-Perfect zu Hilfe.
12. Beginne mit den ersten Inline- und Block-Elementen mit einer Navigation: baue sie gemäss Vorlage nach. Nimm Platzhaltertext zur Hilfe.
 1. Verinnerliche die CSS-Selektoren und lege inline-, inline-block und Block-Elemente an
 2. Achte auf korrektes Verschachteln (wohlgeformt)
 3. Positionieren von Elementen / floaten
 4. Schriften setzen und formatieren
13. Fertige eine Navigation mit anklickbaren Links an. Highlighte den aktiven Menüpunkt.
14. Ergreife die nötigen Massnahmen hinsichtlich Barrierefreiheit.
15. Zwischenkontrolle: Überprüfe mit einer W3C Validierung, ob dein Code valide ist.

PROJEKT ANLEGEN

im Brackets

Navigieren zwischen mehreren html-Dateien

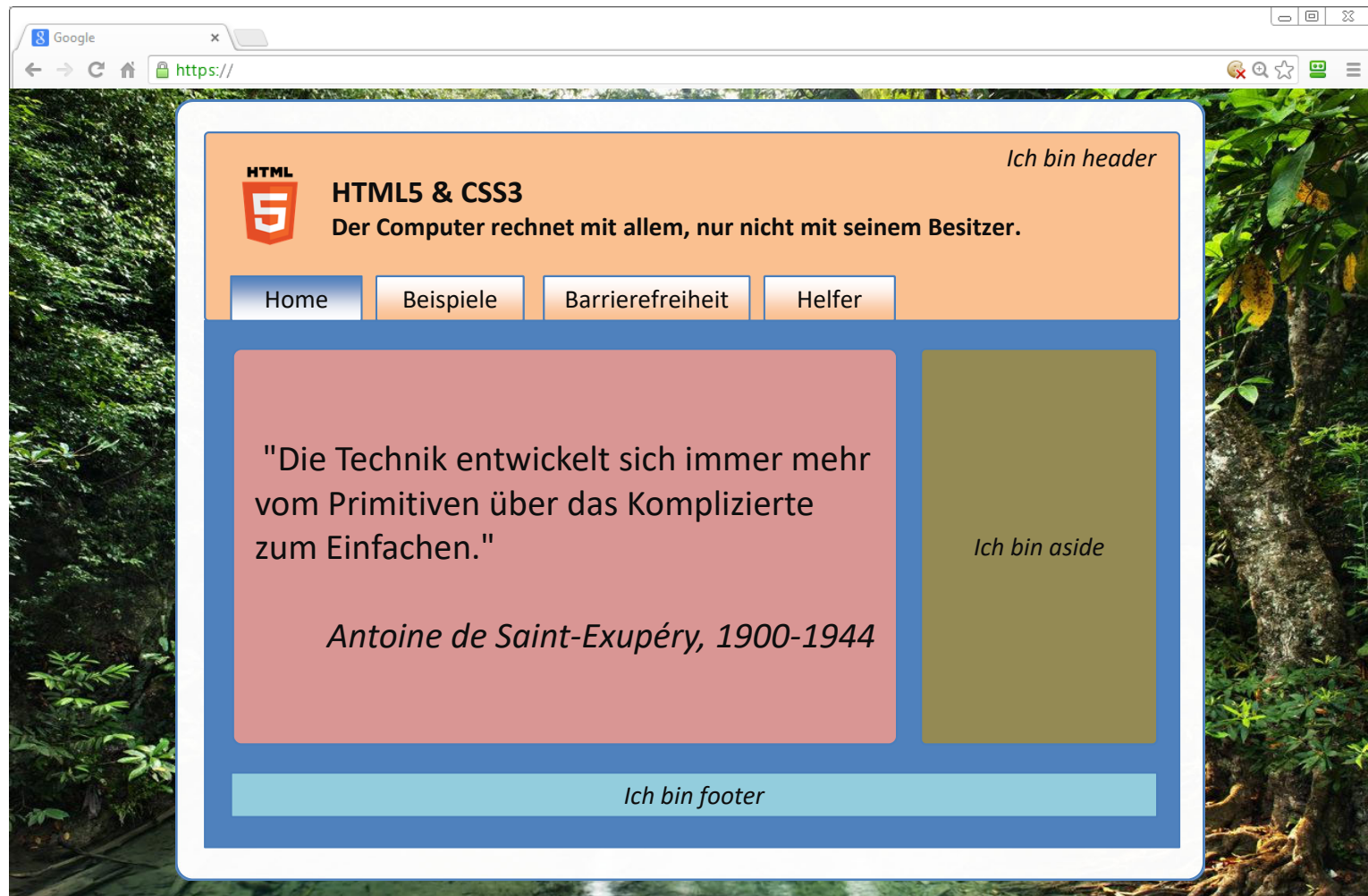
1. Die erste Unterseite erstellen:
Vervielfältige einmal deine Datei index.html, benenne sie nach dem zweiten Navipunkt, verschiebe sie in den Ordner html und passe Pfade, Title-Tag, Navigation und etwas den Inhaltstext der beiden HTML-Dateien an. Navigieren zwischen den Navipunkten index.html und dem zweiten Navipunkt zwecks Kontrolle.
2. Weitere Unterseiten erstellen:
Vervielfältige nun die Datei im html-Ordner gemäss den restlichen Navipunkten und passen jeweils noch die Navigation an, so dass nun zwischen allen Webpages mit der Navi navigiert werden kann. Passe den Platzhalter-Text im Main-Bereich etwas an.

Abfüllen von Inhalten je html-Datei

1. Fülle den Content-Bereich weiter ab, mit Beispielen aus dem Unterricht und berücksichtige dabei immer die Barrierefreiheit.

ZIEL

Statische Website



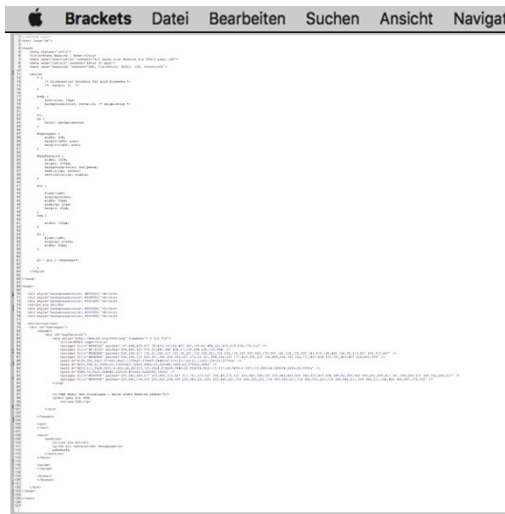
QUELLCODE, DOM und Web Inspector

VOM QUELLCODE ZUM DOM

Der Browser liest den Quellcode zum Document Object Model (DOM) ein

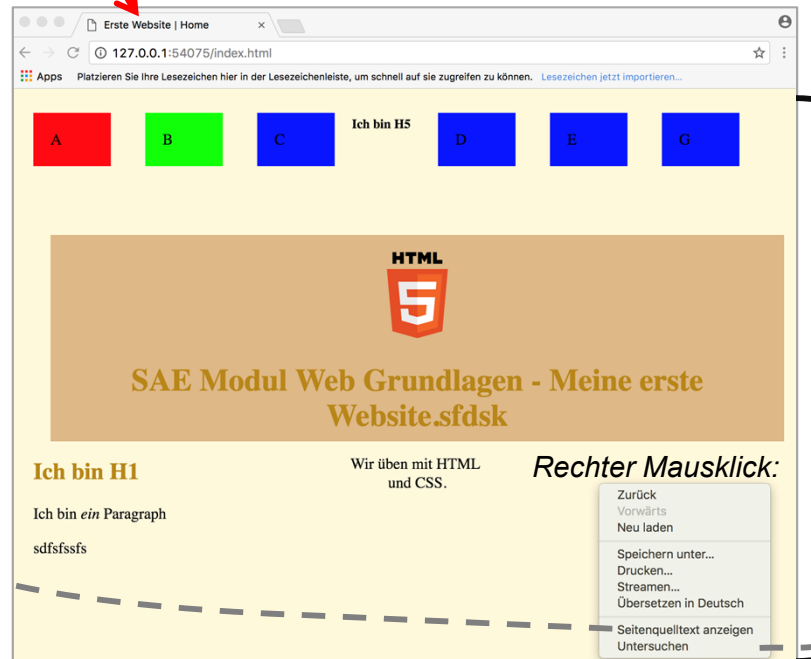
wird geschickt an

Quellcode (index.html)



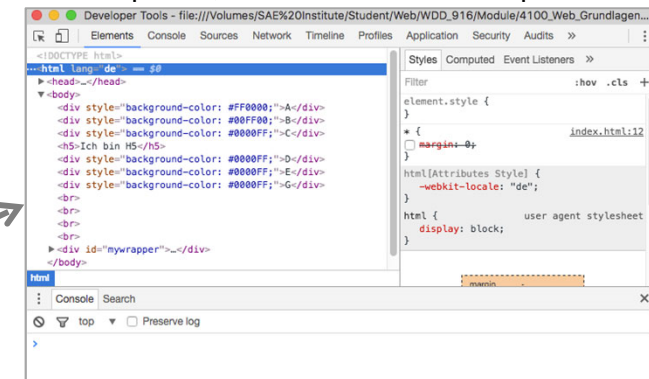
Chrome Datei Bearbeiten Anzeigen Verlauf Lesezeichen Personen Fenster Hilfe

Rendern (berechnen und aufbauen der HTML-Tags zu Modell-Objekten)



Nach dem Rendern haben wir "Baum", den DOM-Baum und die Webpage ist visualisiert.

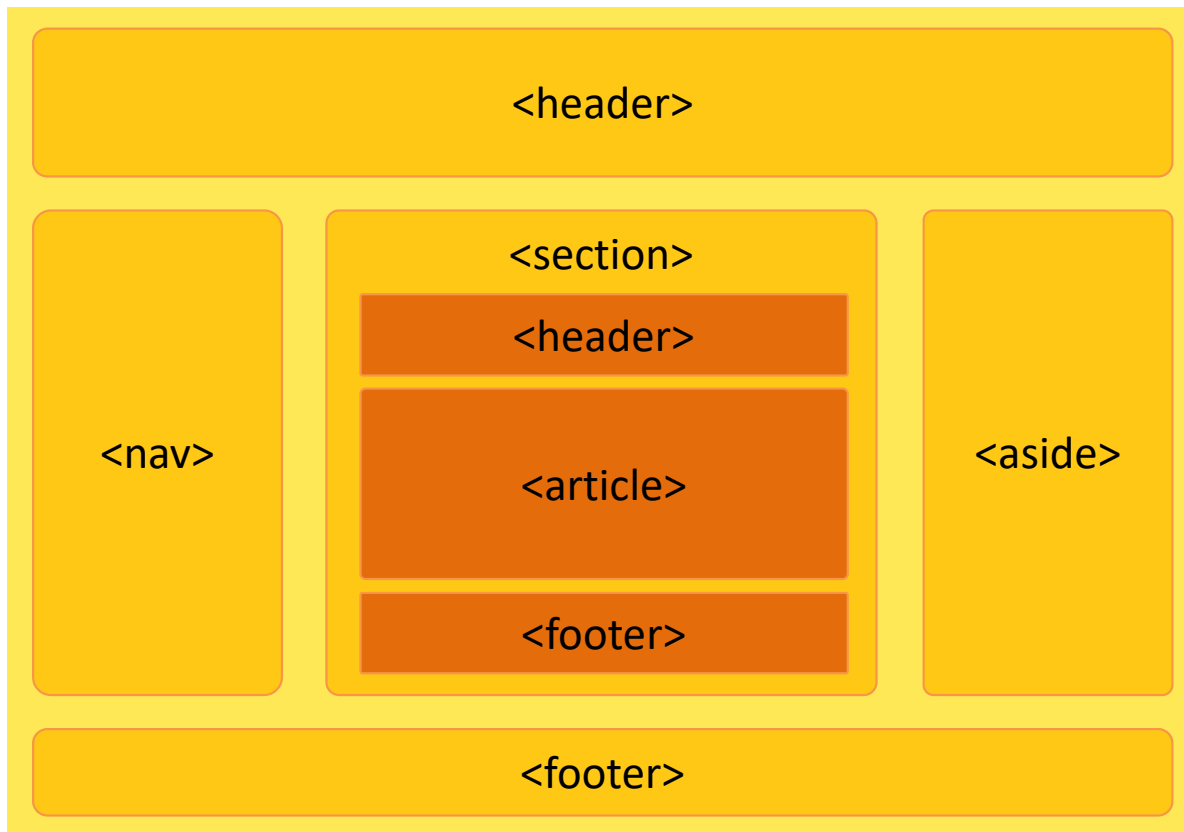
Web Inspector, um den Baum zu manipulieren:



SEMANTIK

SEMANTISCHE TAGS MIT HTML5

Deren Namen haben eine Bedeutung und dienen den Suchmaschinen und uns zur Orientierung.



Weitere semantische Tags:

`<details>`

`<summary>` (innerhalb von `<details>`)

`<figure>` zur Umklammerung von Bild + Erklärung

`<figcaption>` (Bilderklärung)

`<main>`

`<address>`

`<mark>` (hervorgehobener Text)

`<time>` (Datum/Zeitangaben)

`<video>`

`<audio>`

`<svg>`

`<canvas>`