Mit CMD+F suchen!!!

EiA1 – Zusammenfassung

Auszeichnungssprache HTML:

- *Hyper Text Markup Language*
- Aktuell HTML 5
- Statische vs. dynamische Websites
    - Statische: Vorgefertigtes HTML wird vom Server bereitgestellt
    - Dynamisch: Dynamisch erzeugte Webinhalte vom Server, bsp. Suchergebnisse; generierendes Programm bleibt Nutzer verborgen
- Die Stufen im Entwicklungsprozess eines W3C Standards sind:
    - Arbeitsentwurf (Working Draft)
    - Empfehlungskandidat (Candidate Recommendation)
    - Empfehlungsvorschlag (Proposed Recommendation)
    - W3C Empfehlung (W3C Recommendation)
- Block-Elemente, nehmen gesamte zur Verfügung stehende Breite ein:
    - div, header, footer, nav, h1-6, p,…
- Inline-Elemente, kein neuer Absatz, nur so viel Platz wie es braucht:
    - a, em, strong, img, span, …
- **Wichtige HTML-Elemente:**
    - : Block Element ohne semantische Bedeutung
    - : Inline Element ohne semantische Bedeutung
    - <article>: In sich geschlossene Komposition, die unabhängig wiederverwendbar sein soll (z.B. Blockeintrag, Zeitungsartikel, etc.)
    - <section>: In sich geschlossener Abschnitt eines Dokuments, sollte mind. Eine Überschrift haben
    - <aside>: Seitenabschnitt, nicht/nur indirekt mit restlichem Inhalt verknüpft

Mit CMD+F suchen!!!

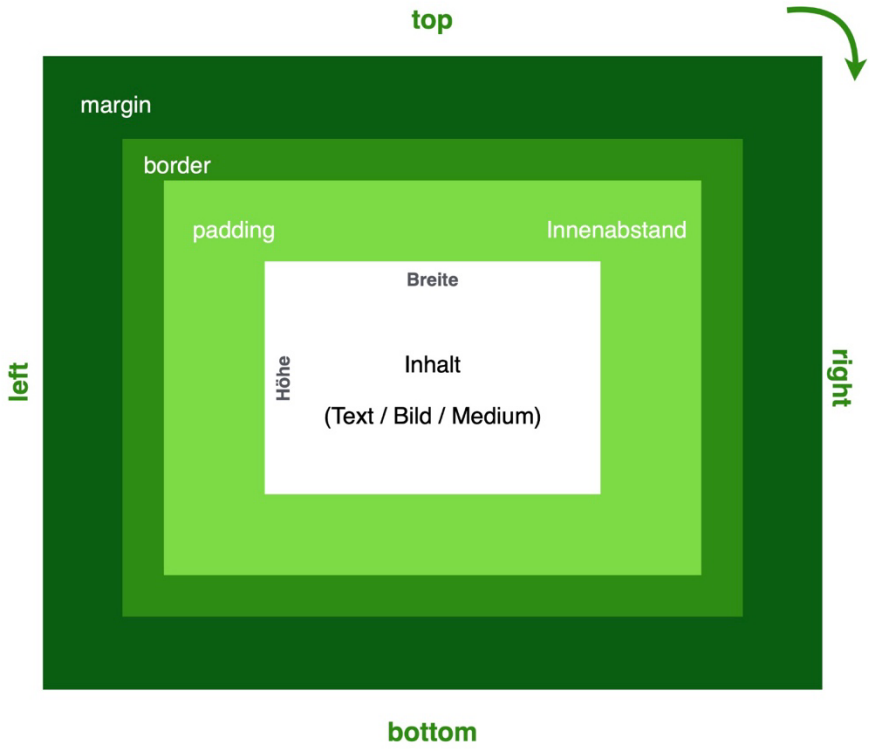
- **HTML-Elemente Tabellen:**
  - <table>: Tabelle selbst
  - <thead>: Die Kopfzeile der Tabelle
  - <tbody>: Hauptteil der Tabelle
  - <tr>: Eine Tabellenzeile
  - <th>: Eine Tabellenzelle (Kopfzeile)
  - <td>: Eine Tabellenzelle (Hauptteil)
- **HTML-Elemente Listen:**
  - : Unsortierte Liste (unorderdList)
  - : Sortierte Liste (orderdList)
  - : Listenelement
- **HTML-Elemente Formulare:** (Eingabemöglichkeiten)
  - <form>: Das gesamte Formular
  - <fieldset>: Eine Gruppierung von Elementen innerhalb der Formulars
  - <input>: Bietet eine große Bandbreite an verschiedenen Eingabeoptionen, ausgewählt durch das „type“-Attribut, z.B.Text, Passwort, Zahl,…
  - <button>: Schaltfläche
  - <select>: Drop-Down oder Listenauswahl
  - <label>: Beschreibungselement, welches einem Formularelement zugewiesen wird (Kann Schaltfläche für zugewiesenes Input Element verändern, nicht nur Box klickbar, sonder auch Text)
  - <output>: Ausgabeelement, welches vom Nutzer nicht verändert werden kann
- Jedes HTML-Tag kann „Attribute“ haben. Jedem Attribut wird ein Wert zugewiesen
  - <element attribut=“wert“> Inhalt </element>
- Allgemeine Attribute (können auf jedes Tag angewendet werden):
  - id => muss einmalig sein
  - class
  - hidden => blendet ein Element aus
  - style => CSS-Anweisungen direkt im HTML nutzbar
  - title => zeigt beim hovern mit der Maus ein Tooltip an

Mit CMD+F suchen!!!

Cascading Style Sheets: CSS

- **Einbindungsvarianten**:
  - Inline => Nicht empfohlen!
  - Über <style> -Tag im Head =>bedingt Empfohlen
  - <link> -Tag im Head Bereich zeigt auf separate CSS-Datei
- **CSS-Anweisung** besteht aus:
  - Selektor
    - Tag-Selektor; Bsp. p für <p>
    - Class-Selektor; Bsp. .klasse für <p class="klasse">
    - ID-Selektor; bsp. #id für <p id="id">
  - Deklaration: gesamte Anweisung, besteht aus:
    - Eigenschaft; bsp. color
    - Und Wert; bsp. red
- **Kaskade** => Gewichtung für Regel und Eigenschaften
  - ID Selektor: Faktor der Kaskade 100
  - Klassen Selektor: Faktor der Kaskade 10
  - Element Selektor: Faktor der Kaskade 1
- **Vererbung und Überschreibung**:
  - Kinder erben automatisch Style-Anweisungen
  - Neue Anweisungen für Kinder, überschreiben vererbte
- **CSS Box Model:**

```css
div {
    background: grey;
    color: white;

    width: 300px;

    padding: 10px 10px 40px 10px;
    border: 5px solid red;
    margin: 10px 10px 40px 10px;
    /*
    margin-top: 10px;
    margin-right: 10px;
    margin-bottom: 40px;
    margin-left: 10px;
    */
}
```



Mit CMD+F suchen!!!

- **CSS Maßeinheiten:**

Einheit	Beispiel	Anwendung	Eigenschaften
px	`margin: 16 px;`	Bildschirm	Pixelorientiert – abhängig von der Ausgabe und dem Betriebssystem
pt	`margin: 16 pt;`	Bildschirm / Druck	72pt entsprechen einem Inch/Zoll 1pt => 0,35mm
em	`font-size: 1 em;`	Flexible Bildschirmgrößen	Relative Angabe im Verhältnis zum **übergeordneten** Element
rem	`font-size: 1 rem;`	Flexible Bildschirmgrößen	Relative Angabe im Verhältnis zum **root** Element
Prozent	width: 70%	Flexible Bildschirmgrößen	Relative Angabe im Verhältnis zum **übergeordneten** Element

- **Komplexe Selektoren:**

Art	CSS-Beispiel	Zugriff auf
Nachfahren	`div h3 {color: red}`	alle <h3> Elemente unterhalb eines
Eltern/Kind	`div>h3 {color: red}`	alle <h3> Elemente, die ein unmittelbares Kind von sind.
Folgeelement	`h1+p {…}`	ein <p> Element, das unmittelbarer Nachbarknoten von <h1> ist.
Geschwister	`div ~ p`	alle <p> Geschwisterknoten von div
Universal	`main * {…}`	alle Unterelemente von <main>

Art	CSS-Beispiel	Zugriff auf
Attribut	`img[alt=kaiserschmarrn]`	ein Bild mit dem alt-Attribut „kaiserschmarrn“
Pseudoklassen	`a:hover {color: red}`	einen Anker im hover-Zustand (MouseOver)
Pseudo-Elemente	`p::before`	einen Inhaltsknoten eines <p> Elements. Einsprung vor dem ersten Zeichen.
Wiederholung	`:n-th-child(2n)`	jeden zweiten Kindknoten

Mit CMD+F suchen!!!

- **Flussverhalten:**

Position	Style Attribute	Verhalten
float	none I left I right	Legt fest, ob und wie ein Element umflossen wird. Achtung: Element wird dabei aus dem Dokumentenfluss herausgenommen!
clear	none I left I right I both	Beendet das Floaten. Die Zuweisung erfolgt im nachfolgenden Element.
display	inline I block I table I list I none	Legt fest, welchen Darstellungs-Typ die Box um das Element bekommt.
overflow	visible I hidden I scroll I auto	Definiert das Verhalten eines Container-Elements, wenn der Inhalt des Containers größer ist, als durch den Container festgelegte verfügbare Platz.

Position	Style Attribute	Verhalten
grid	grid I inline-grid	Bietet die Möglichkeit, Elemente in einem nativen Grid darzustellen.
flexbox	-	Bietet eine effizientere Möglichkeit um Elemente zu positionieren.

- **Positionierung:**

Position	Style Attribute	Bezugspunkt	Flussverhalten
static	–	–	Standardflussverhalten
relative	top I left I bottom I right I **z-index**	Relativ zur ursprünglichen Position	Standardflussverhalten
absolute	top I left I bottom I right I **z-index**	Relativ zum ersten positionierten (nicht: static) Vorfahren-Element	Element wird aus dem Standardfluss herausgenommen. Nachfolgende Elemente rutschen nach.
fixed	top I left I bottom I right I **z-index**	Linke obere Ecke des Browserfensters	Element wird aus dem Standardfluss herausgenommen. Nachfolgende Elemente rutschen nach.

Mit CMD+F suchen!!!

- **Responsive Design**: Mobile First Ansatz

```css
/* Allgemeine Anweisungen = Smartphone Darstellung */
h1 {
    color: red;
    font-size: 20px;
}
/* Besondere Anweisungen für größere Auflösungen */
@media screen and (min-width: 1024px) {
h1 {
    font-size: 40px;
}
}

/* Besondere Anweisungen für andere Medien */
@media print {
h1 {
    color: black
}
}
```

- **Responsive CSS** Bausteine:

Festlegung Viewport	`<meta name="viewport" content="width=device-width, initial-scale=1.0">`
Mediaquery Screen	`@media screen and (min-width: 768px) {…}`
Mediaquery Print	`@media print {…}`
Alternative Einbindung	`<link rel="stylesheet" media="screen and (min-width: 768px)" href=„tablet-portrait.css">`
Mobile first!	Zuerst mobile CSS-Definition, dann Tablet, Desktop,…

Mit CMD+F suchen!!!

TypeScript:

- **Script Einbindungsmöglichkeiten:**
  - Siehe CSS
  - <script src="script.js"></script>
- TypeScript wird zu regulärem JavaScript kompiliert
- **Variablen Deklaration**:
  - Variablen Deklaration (var, let, const,…)
  - Bezeichnung der Variable
    - keine Ziffer am Anfang
    - keine Leerzeichen
    - Keine Bindestriche
    - Keine Verwendung von Schlüsselwörtern wie „string“
  - „:“ => syntaktischer zuordnungsoperator
  - Daten Typ:
    - string
    - number
    - boolean
    - array
    - any
    - void

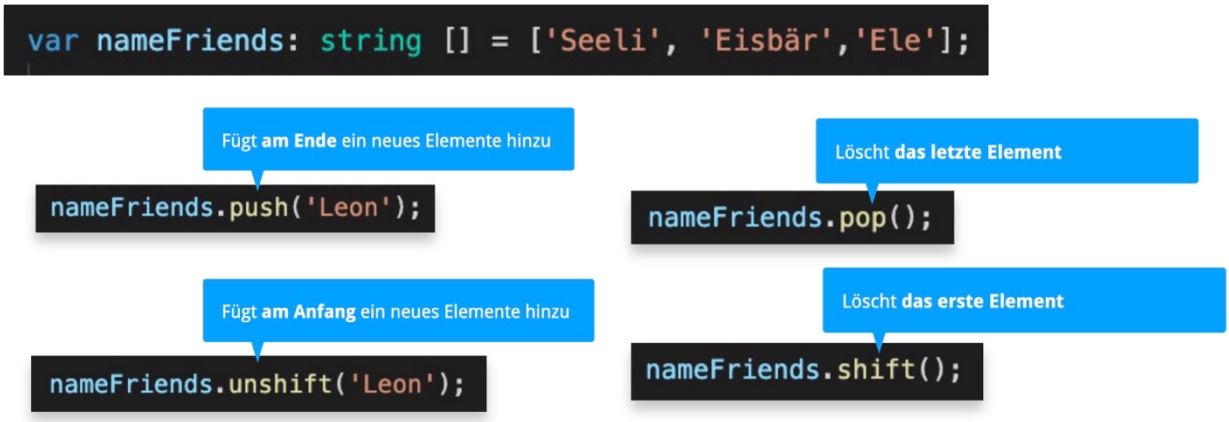

```
var animal : string = "Pingu";
var age : number = 2;
var likesFish : boolean = true;

var city : string = "Furw@ngen";
var weight : number = 5.3;
var enabled : boolean = false;
```

- **Mathematische Operatoren:**

Addition	+	17+3	20
Subtraktion	-	10-3	7
Multiplikation	*	3*5	15
Division	/	15/3	5
Divisionsrest / Modulo	%	12%5	2
Incrementor	++	var i=10 i++;	11
Decrementor	--	var i=10 i--;	9
Zuweisungsoperator	+= -= *= /=	var i=10; i+=5;	15

Mit CMD+F suchen!!!

- **Funktionen Deklarationen**
  - Keyword „function“
  - Funktionsname, Benennungskonvention wie bei Variablen
  - Funktionsklammern „()“, ggf. mit Argumenten
  - Anweisungsklammern „{}“
- **Funktionsaufruf:**
  - Funktionsname
  - Funktionsklammern, ggf. mit Argumenten
  - Semikolon
- Funktionen: **Argumente**
  - Argument wird innerhalb der Funktionsklammern festgelegt
    - Mit Name
    - Und Typ (string, number etc.)
  - Argument wird innerhalb der Anweisungsklammern wie eine Variable verwendet.
  - Beim Funktionsaufruf wird ein Wert für jedes Argument übergeben. Überall wo Argument in Anweisungsklammer verwendet wird, wird der übergebene Wert eingesetzt.
  - Mehrere Argumente werden Komma separiert angegeben.
- Deklaration **einfaches Array**:



Mit CMD+F suchen!!!

- **Vergleichsoperatoren**

Gleichheit der Werte	==	5 == 5	true
		6 == 5	false
Ungleichheit der Werte	!=	6 != 5	true
		5 != 5	false
Größer als	>	5 > 2	true
Kleiner als	<	5 < 8	true
Größer oder gleich	>=	6 >= 5	true
Kleiner oder gleich	<=	5 <= 5	true

- **Logische Operatoren**

Logisches „UND“	&&	x = 5; y = 2;	x > 3 && y < 5	true
			x > 5 && y < 5	false
Logisches „ODER“	||	x = 5; y = 2;	x > 3 || y < 5	true
			x > 5 || y < 5	true
Logisches „NOT“	!	x = 5; y = 2;	!(x == y)	true

- **Variablen Deklaration**
  - let-Variable mit eingeschränktem Gültigkeitsbereich => kann nicht mehrfach deklariert werden/neu deklariert werden
  - var-Variable kann mehrfach deklariert werden => Fehleranfälliger
  - const-Variable, lokaler Gültigkeitsbereich, kann nicht durch Zuweisung/ neu Deklaration verändert werden

Mit CMD+F suchen!!!

Debugging:

```
console.log("Eine Logausgabe");
console.info("Eine Info");
console.warn("Eine Warnung");
console.error("Ein Error");

console.group("Eine Gruppe");
console.log("Gruppenelement 1");
console.log("Gruppenelement 2");
console.groupEnd();
```

Konsolenausgaben können in verschiedenen Formaten ausgegeben werden.