Einführung JavaScript

Interpretierte Programmiersprache

- Anders als Cobol, C++, die kompiliert werden müssen
- JavaScript hat nichts mit Java zu tun
  - Name entstand aus Marketinggründen 1995
  - Entwickelt für Netscape
- Alle gängigen Browser können JavaScript interpretieren
- Aber auch Server mit node.js
- (oder ganz neu: deno.js)
- Vereinheitlichung durch ECMA (European Computer Manufacturers Association)
- Standard heute ECMAScript
- https://en.wikipedia.org/wiki/ECMAScript

Bedeutung von JavaScript

- **Webentwicklung: TypeScript**
- In der Webentwicklung gibt es seit Jahren die immer gleichen Sprachen, die ihr drauf haben solltet. Wollt ihr in diesem Segment entwickeln, kommt ihr um JavaScript eigentlich nicht herum. Wer JavaScript beherrscht, ist auf jeden Fall gut aufgestellt. Seit 2012 gibt es einen neuen Player aus dem Hause Microsoft: TypeScript. Im Gegensatz zu JavaScript ist TypeScript statisch typisiert und gibt euch mehr Kontrolle über euren eigenen Code.
- Doch keine Angst! Wer JavaScript beherrscht, wird sich auch schnell in TypeScript einarbeiten können. Denn TypeScript versteht JavaScript und vice versa.

Client- und Serverseitiges JavaScript

- Mit JavaScript können sowohl im Frontend als auch im Backend Anwendungen programmiert werden.
- Es gibt hunderte Frameworks, die auf JavaScript basieren
  - React.js
  - Angular.js
  - Vue.js
  - und tausend andere
- Lange Zeit wurde im Frontend häufig auf Jquery zurückgegriffen. Zur Zeit liegt der Trend darin, wieder natives JavaScript zu schreiben. (Auch vanilla.js genannt … ironisch gemeint.)

JavaScript einbinden

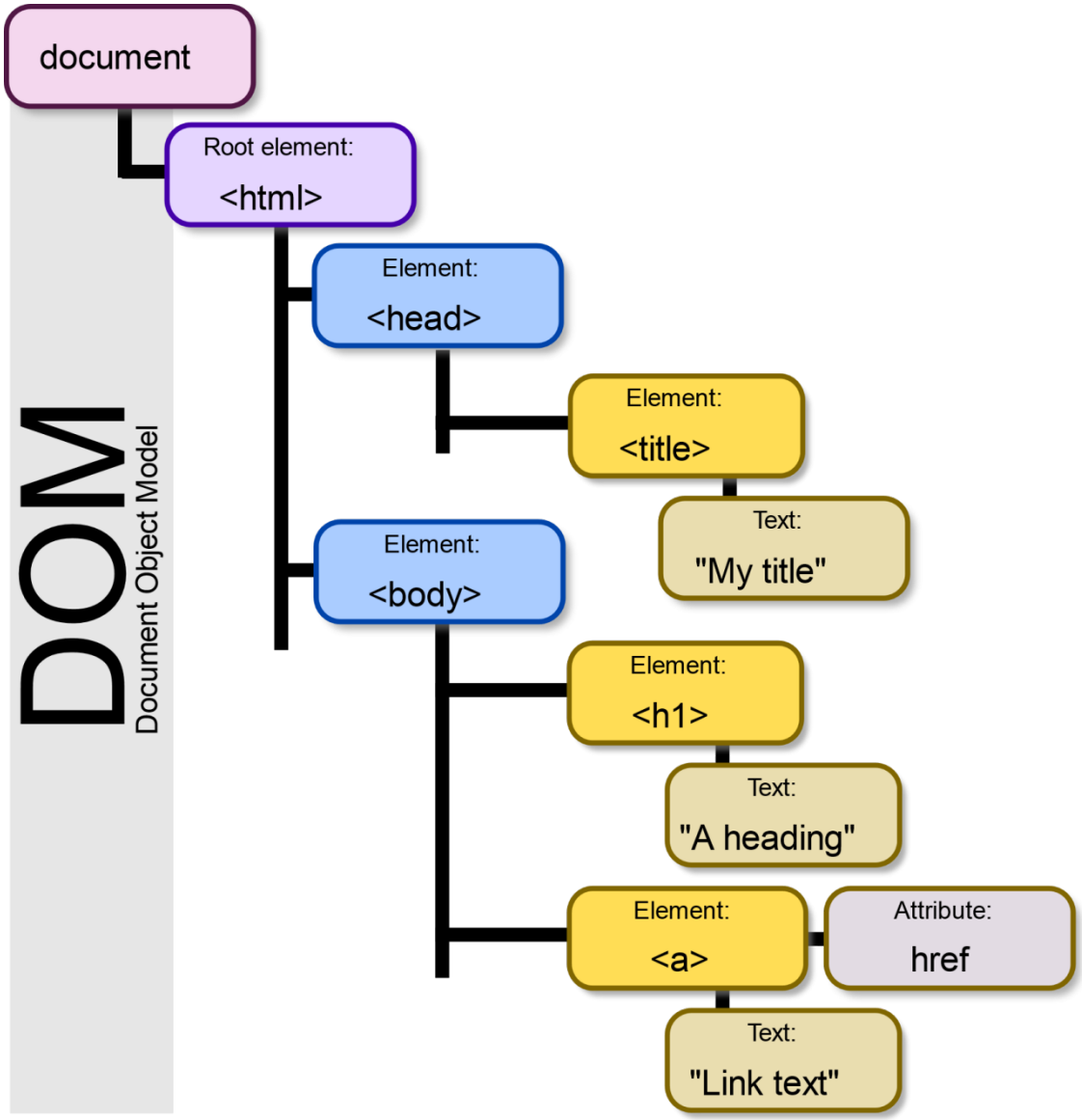
```
...

<script src="js/main.js"></script>
</body>
```

Immer am Ende einbinden, damit HTML-Code ganz geladen ist.

Das DOM – Document Object Model

- The **Document Object Model** (**DOM**) is a cross-platform and language-independent interface that treats an XML or HTML document as a tree structure wherein each node is an object representing a part of the document.

DOM – Beispiel



Warum ist das DOM so wichtig?

- Mit JavaScript lassen sich Webseiten dynamisch verändern.
- Dabei greift man auf vorhandene Elemente (im DOM-Baum vorhanden) zu und verändert diese oder fügt davor oder hinter etwas ein.
- Es gibt folgende Knotentypen, die am häufigsten verwendet werden:
- **Dokumentknoten**, steht für die Wurzel und die gesamte Webseite – globales Objekt document
- **Elementknoten** – repräsentieren HTML-Elemente der Webseite (h1, p, li, ul)
- **Attributknoten** – Attribute von HTML-Elementen (id, href)
- **Textknoten** – Text innerhalb von HTML-Elementen

Elemente selektieren

Was wählt was aus? Überlegen Sie und nennen Sie ein Beispiel:

- getElementbyId()
- Wählt ein Element anhand der id aus.
- getElementsbyClassName()
- Wählt Elemente anhand eines Klassennamens aus
- getElementsbyTagName()
- Wählt Elemente mit angegebenen Elementnamen aus.
- querySelector()
- Gibt das erste Element zurück, das auf einen gegebenen CSS-Selektor passt
- querySelecorAll()
- Gibt alle Elemente zurück, die auf einen gegebenen CSS-Selektor passen

Beispiele

- document.getElementById("reset")
- document.getElementsByClassName("card-header")
- document.getElementsByTagName("p")
- document.querySelector(".container h1")
  - Findet nur das erste Element
- document.querySelectorAll(".card div.card-header")

Elemente selektieren

- parentElement
- parentNode
- nextSibling
- firstChild
- lastChild

- Und und und

Auf Ereignisse reagieren – EventHandler , EventListener

- Ereignisgesteuerte Programmierung

1. Auswahl des Elements auf der Webseite, welches auf die Benutzerinteraktion reagieren soll
2. Angabe des Events, welches abgefangen werden soll: Tastatureingaben, Mausklick, Mausover
3. Angabe der Funktion, die aufgerufen werden soll

Modernes JavaScript

1. HTML-Event-Handler: Event-Handler über HTML-Attribute definieren. <p onclick=„checkAge“> ->veraltetes JavaScript, nicht mehr verwenden
2. DOM-Event-Handler – ebenfalls veraltet

```
function init() {
let element = document.getElementById(„age“);
element.onblur = checkAge
}
window.onload = init
```

DOM Event Listener – Standardweg bei der Ereignisbehandlung
Mehrere Funktionen können für ein Ereignis definiert werden.

```
let element = document.getElementbyId(„age“)
element.addEventListener ( ‚click‘, checkAge);
```

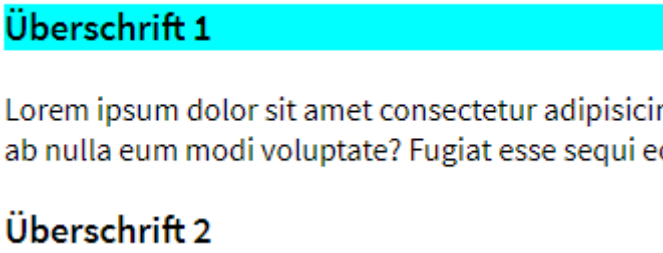
Progressive Enhancement

- Progressive Enhancement (zu deutsch fortschreitende Verbesserung) beschreibt eine Vorgehensweise im Webdesign, die Barrierefreiheit, semantische Auszeichnung und Trennung von Information und Darstellung beinhaltet, um eine Website auch für Geräte benutzbar zu machen, die nur über eingeschränkte Funktionen verfügen – z. B. ohne die Unterstützung von JavaScript, CSS oder Flash.
- Die Idee dahinter ist, dass man die grundlegendste Form einer Websites mit jeder Art von Webbrowser und jeder Internetverbindung aufrufen kann und die erweiterte Funktionalität dann mit besserer Bandbreite und fortgeschritteneren Browsern zusätzlich dargestellt wird.

Übung 1

```
"use strict"

document.addEventListener("DOMContentLoaded", () => {
    // Nachdem DOM geladen, kann die Id welcome gefunden und angesprochen werden
    const welcomeHere = document.getElementById("welcome")
    //Unterschied
    console.log(welcomeHere)
    console.log(welcomeHere.innerText)

    welcomeHere.addEventListener("click", welcome)

    function welcome() {
        welcomeHere.innerHTML = "Willkommen <strong>Klasse I13</strong>";
    }

    /* welcomeHere.addEventListener("click", () => {
     welcomeHere.innerHTML = "Willkommen <strong>Klasse I13!!!!!</strong>";
     })*/
})
```

Übung 2 – für alle Überschriften Text „toggeln“



- Alle Überschriften holen

```
let Headings = document.getElementsByClassName("ueberschriften")
```

- Für alle Überschriften

```
for (const Heading of Headings) { …}
```

- Texte (Paragraph) unter Überschrift holen

```
let inhaltParagraph = Heading.nextElementSibling
```

- Den Inhalt ausblenden (Klasse gibt es schon)

```
inhaltParagraph.classList.add("invisible")
```

Mit **classList.toggle**
kann jetzt die Klasse hinzugefügt und entfernt werden

Übung 3

- Farbe abwechselnd ändern für die Überschriften.
- Wenn Text sichtbar, cyan, sonst weiß
- Ebenfalls mit Klasse oder mit

```
Heading.style.backgroundColor = "cyan"
```

Anonyme Funktionen

- Eine Funktion ohne Namen ist eine anonyme Funktion.

```
let showMessage = function() {
                console.log("Hello World");
            }
showMessage();
```