Donnerstag, 22.09.22 - Ablauf

- 9:00 - 10:30 Recap CSS / Einführung JS
- 10:30 - 10:45 Pause
- 10:45 - 11:00 Aktivierungseinheit
- 11:00 - 12:30 JS-Fortsetzung (Theorie)
- 12:30 - 12:45 Pause
- 12:45 - 13:30 JS-Übung
- 13:30 - 14:45 Mittag
- 14:45 - 16:15 JS Ergebnissicherung/Selbststudium (nachmittags)

Donnerstag, 22.09.22 - Recap/Fortsetzung CSS

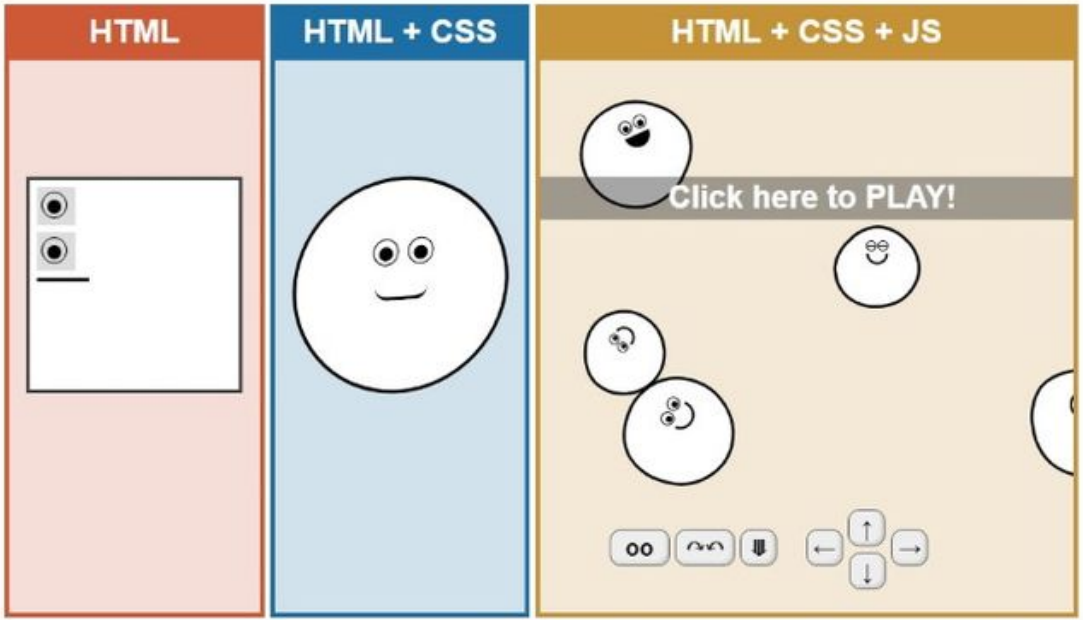
Was haben Sie bisher gelernt / mitgenommen?

Was hat gut / schlecht geklappt?

Haben Sie Fragen bis hier?

Wer möchte seine Webseite zeigen?

Donnerstag, 22.09.22 - Zusammenspiel



Sprachen des Webs: HTML, CSS und JS
Sviat Drach, Pia Geißel, Lena-Luise Stahn, Nadine Sutor

Donnerstag, 22.09.22 - Einführung in JavaScript **(.js)**

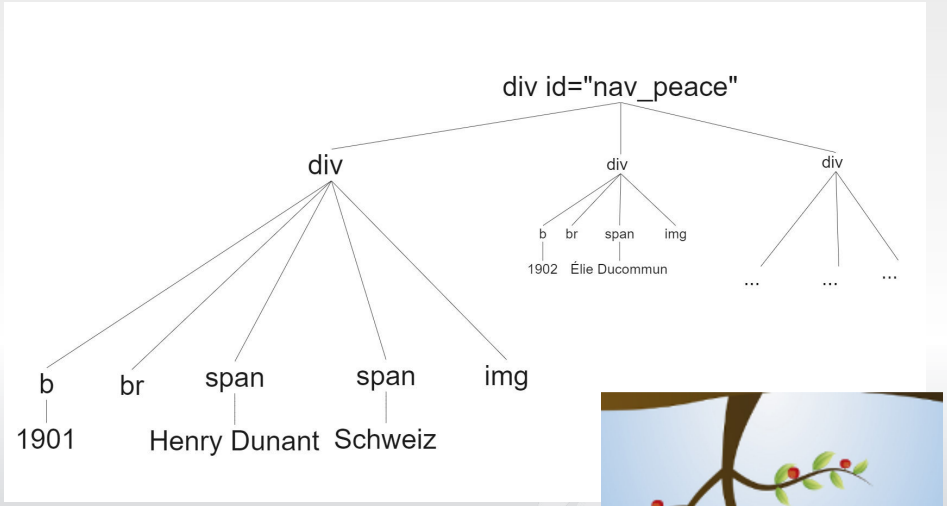
- Wofür wird JavaScript verwendet?
- Funktionsweise/Logik
- erste Übungen
- Ein paar Anmerkungen:
  - In dieser Woche beschäftigen wir uns ausschließlich mit “Vanilla JS”
  - Code “from scratch” vs. Frameworks (Vor- und Nachteile? Was nutzt man wofür? Projektkontexte?)
  - Wir behandeln hier nur client-seitiges Javascript, d.h. nur das, welches vom Browser (=Internet-Client) verarbeitet wird.

Donnerstag, 22.09.22 - JavaScript - Motivation

- wurde 1995 als Skriptsprache von Netscape für Webbrowser entwickelt und 1997 als **ECMAScript Standard** spezifiziert
- gehört mit HTML und CSS zu den **Kerntechnologien** des World Wide Web, von nahezu jeder Webseite verwendet
- Alle gängigen Webbrowser verfügen über eine **JavaScript Engine** zur Ausführung des client-seitigen JS (meist ES7 2016)
- wird inzwischen auch außerhalb von Browsern, d.h. server-seitig, angewendet (z.B. mittels Laufzeitumgebung node.js)
- es ermöglicht die **dynamische Manipulation** von Webseiten über das **Document Object Model**
  - Benutzerinteraktionen werden ermöglicht
  - Inhalte können verändert, nachgeladen oder neu generiert werden
  - ...

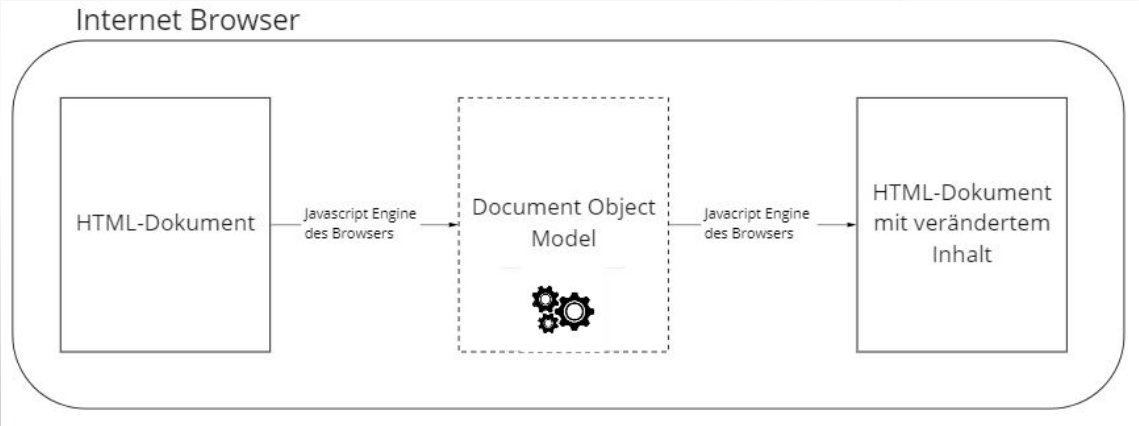
Donnerstag, 22.09.22 - Das Document Object Model (DOM)

- Das DOM ist eine plattform- und **sprachunabhängige Konvention**, W3C Standard
- zur Repräsentation von und Interaktion mit Objekten in HTML und XML Dokumenten
- versteht das Dokument (die Webseite) als **Baummodell**, die Knoten darin sind Objekte (Dokument-, Fragment-, Element-, Attribut-, Textknoten)
- diese werden über Methoden angesprochen und durch Aufrufen ihrer Methoden verändert



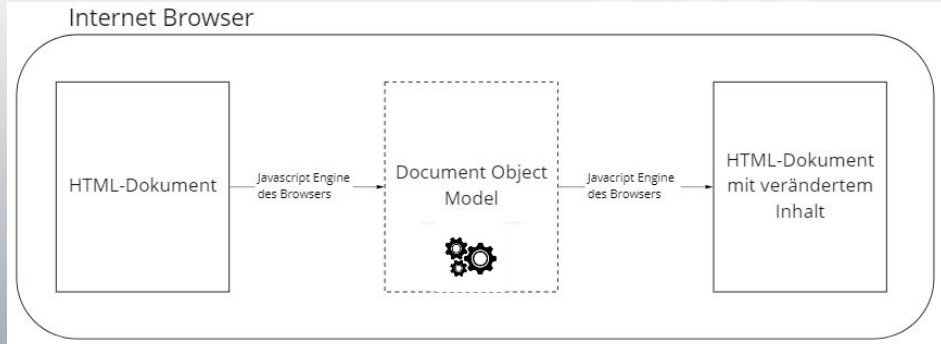
Donnerstag, 22.09.22 - Das Document Object Model (DOM)

- Das Dokument-Objekt wird durch Einlesen eines bestehenden HTML-Dokuments erzeugt, sodass auf die Inhalte, Struktur und Darstellung dieses HTML zugegriffen werden kann;
- insbesondere wird damit möglich
  - die Navigation zwischen den einzelnen Elementen eines Dokuments,
  - das Erzeugen, Verschieben und Löschen von Elementen sowie
  - das Auslesen, Ändern und Löschen von Textinhalten



- Anschließend wird ein “neues” HTML-Dokument generiert, das dynamische Funktionen, z.B. eine Suche, besitzt.

- Die Webseite wird im Browser aufgerufen und geladen.
- Der dazugehörige Code (HTML, CSS, JS) wird **innerhalb des Browser-Tabs** ausgeführt:
  - HTML und CSS werden zur Webseite “zusammengebaut”
  - **anschließend** führt die JS Engine des Browsers das JS aus
- Ausführung des JS erfolgt von oben nach unten (Achten Sie auf die Reihenfolge Ihres Scripts!)



Einbindung: Wo schreiben wir unseren JS-Code hin?

- Javascript-Code wird im HTML-Dokument mit dem tag **`<script>`** eingeleitet (analog zum tag `<style>` für die Einleitung für CSS-Code!).
- Es gibt (wie beim CSS) mehrere Möglichkeiten, den JS-Code mit dem HTML zu verbinden.

1. Inline JS (schlechte Praxis!)

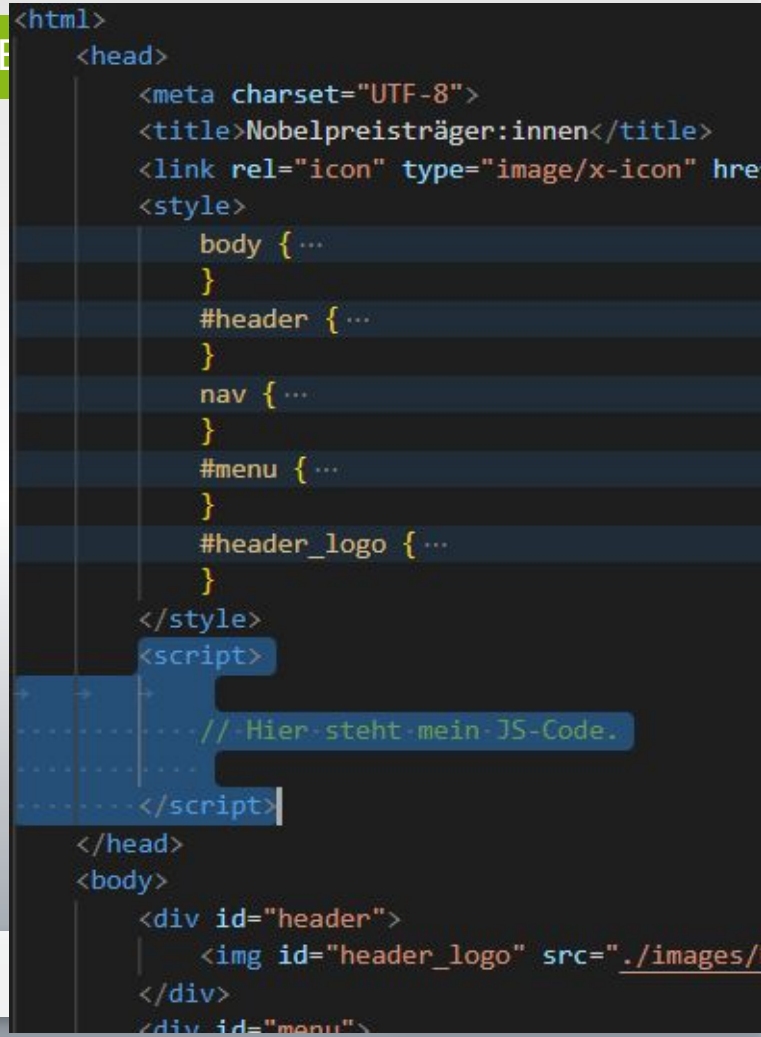
```
<body>
  <button onclick="createParagraph()">Click me!</button> event
  <script>
    function createParagraph() { var para = document.createElement('p'); para.textContent = 'You
    clicked the button!'; document.body.appendChild(para); }
  </script>
</body>
```

2. Internes JS:

Einbindung des Codes direkt im HTML-Dokument mittels <script>-Element:

```
<script>  // hier steht der JavaScript-Code  </script>
```

ACHTUNG: Die Anweisung **kann sowohl im <head>- als auch im <body>-Teil** des HTML-Dokuments geschrieben werden; wo man sie platziert, hängt davon ab, **wann das JS ausgeführt werden soll**: sobald der Browser beim Laden der HTML-Seite auf ein **<script>** stößt, wird dieses ausgeführt, unabhängig davon, ob bereits alle DOM-Elemente aufgebaut wurden, die das JS zur Ausführung benötigt. Daran sollte man denken, wenn man sich für eine Variante der Platzierung des JS-Codes direkt im HTML entscheidet.



3. Externes JS:

Einbindung eines externen JS-Dokuments mittels `<script>`-Element und Referenz auf die externe .js-Datei mit dem `src`-Attribut

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

ACHTUNG: Wie bei Möglichkeit 2 gilt auch hier der Hinweis, auf die Reihenfolge bzw. die **Platzierung des `<script>`-Elements vor oder hinter dem `<body>`-Teil** des HTML zu achten, je nachdem, wann das Script ausgeführt werden soll.

Donnerstag, 22.09.22 - JavaScript - Übung 1

```
alert("Herzlich Willkommen!");
```

1. Öffnen Sie Ihre HTML-Datei im VSCode.
2. Öffnen Sie die Datei im Browser. (*Zusatz: Falls Sie bereits eine externe JS-Datei angelegt haben, prüfen Sie ggf., ob das CSS korrekt eingebunden wird.*)
3. Geben Sie **direkt hinter dem schließenden `</style>`-Element im html-`<head>`** das `<script>`-Element ein, um anzuzeigen, dass hier ausführbarer Code folgt.
4. Schreiben Sie den links stehenden Code in das `<script>`-Element.
5. Speichern Sie die HTML-Datei und laden Sie die im Browser geöffnete Datei erneut (ACHTUNG: in der Vorschau des VS Code funktioniert diese Anweisung nicht!)

Donnerstag, 22.09.22 - JavaScript - Übung 2

```
function greetMe(yourName) {
    alert("Hello " + yourName);
    }

greetMe("World");
```

1. In derselben HTML-Datei geben Sie **direkt vor dem schließenden `</body>`-Element** das `<script>`-Element ein, um anzuzeigen, dass hier ausführbarer Code folgt.
2. Schreiben Sie den links stehenden Code in das `<script>`-Element.
3. Speichern Sie die HTML-Datei und laden Sie die im Browser geöffnete Datei erneut.

Donnerstag, 22.09.22 - JavaScript - function()

Eine **function definition** (auch **function declaration** oder **function statement; dt.:** Funktionsdefinition) besteht aus dem keyword function, gefolgt von:

- dem Namen der Funktion,
- einer Liste an Parametern, die die Funktion benötigt, umschlossen von runden Klammern und durch Kommata getrennt,
- Dem Anweisungsblock,dh. der “Aufgabe”, die die Funktion durchführen soll, umschlossen in geschweiften Klammern, { /* … */ }.
- Der Aufruf der Funktion erfolgt über den Funktionsnamen und ggf. Notwendige Parameter in Klammern dahinter.

```
function greetMe(yourName) {
    alert("Hello " + yourName);
}

greetMe("World");
```

Donnerstag, 22.09.22 - PAUSE

10:30 - 11:00

Donnerstag, 22.09.22 - JavaScript - Debugging

Debugging (oder Debuggen) (nur eine Möglichkeit von vielen):

> “Als **Debuggen** bezeichnet man in der Informatik den Vorgang, in einem Computerprogramm Fehler oder unerwartetes Verhalten zu diagnostizieren und zu beheben.”
>
> [Wikipedia-Artikel “Debuggen” https://de.wikipedia.org/wiki/Debuggen abgerufen am 06.09.22]

D.h.: **Wie teste ich, ob mein Script gelesen wird?**

Donnerstag, 22.09.22 - JavaScript - Übung 3: Debugging

Lassen Sie sich “Haltepunkte” in der Konsole ausgeben mit der Methode **console.log()**.

1. Ergänzen Sie Ihr Script (dh. zwischen den `<script>`-tags in der HTML-Datei) um die folgende Zeile (siehe rechts, Text in Anführungszeichen frei wählbar):

   console.log(“Dies ist ein Test.”);

2. Öffnen Sie die Entwicklertools des Browsers und wählen Sie den **Reiter “Konsole”**:
3. Laden Sie die Seite neu.

```
function greetMe(yourName) {

    console.log(“Dies ist ein Test.”);

    alert("Hello " + yourName);
}

greetMe("World");
```

Aufruf der Entwicklertools s. Folie 22

Donnerstag, 22.09.22 - JavaScript - Variablen

- **const** legt eine Konstante an, die anfangs an einen Wert gebunden wird, welcher später nicht verändert werden kann (immutable)
  - ```
    const facsimiles = document.getElementsByClassName('rs'),
          button = document.getElementsByName('button'),
          PI     = 3.14;
    ```
- **let** legt eine Variable im aktuellen Gültigkeitsbereich an, die nicht gleich deklariert werden muss. Für diesen Fall erhält sie den Wert undefined.
  - ```
    let   text   = 'Hallo Welt!';
    ```
  - ```
    let   alter;   // undefined
    ```
- Der **scope** bezeichnet den **Gültigkeitsbereich** der Variable
  - globale Variable: am Anfang des Dokuments deklariert, innerhalb des gesamten Dokuments gültig
  - lokale Variable: innerhalb einer Funktion deklariert, nur innerhalb dieser Funktion gültig
- Variablennamen in der (guten) Praxis:
  - sprechende Namen
  - case-sensitive
  - camelCase
    ```

Donnerstag, 22.09.22 - JavaScript - Datentypen

- eine bestimmte Menge gleichartiger Werte, mit denen festgelegte Operationen ausgeführt werden können
- **einfache** (primitive) Datentypen
  - Undefined
  - Null
  - Boolean, Wahrheitswert, kann wahr (true) oder falsch (false) sein
  - String, eine Zeichenkette, zum Beispiel "Selfhtml".
  - Symbol, ein spezieller Typ zum Erzeugen eindeutiger Zugriffsschlüssel
  - Number, eine Zahl, mit oder ohne Nachkommastellen.
  - BigInt, ein ganzzahliger Wert beliebiger Größe

- **Objekte**: Sammlung von Eigenschaften und objektgebundenen **Methoden** (Funktionen)

  - Erzeugen eines Objekts

```
const person = {
  firstName: "John",
  lastName: "Doe",
  fullName: function() {
       return this.firstName + " " + this.lastName;
  }
};
```

  - Aufrufen eines Objekts (Bsp.: Ausgabe in der Konsole mithilfe von console.log()):

```
console.log(person.firstName);    // "John"
console.log(person);              // Wiedergabe des gesamten Objekts
console.log(person.fullName());   // "John Doe"
```

Donnerstag, 22.09.22 - JavaScript - Kommentare und Operatoren

Kommentare

- // Ich bin ein einzeiliger Kommentar.
- **/***
  Ich bin ein
  mehrzeiliger Kommentar.
  ***/**
- werden nicht als ausführbarer Code interpretiert
- dienen der Dokumentation des Codes (gute Praxis!)

Donnerstag, 22.09.22 - JavaScript - Kommentare und Operatoren

Operatoren

- = Zuweisungsoperator:
  ```
  const variableEins = ‘Lena’;
  ```
- === Gleichheitsoperator:
  ```
  let variableZwei = 3;
  variableZwei === 4;
  ```
- +, -, *, / mathematische Operatoren, + dient außerdem zum Verbinden von Strings
  ```
  6 + 9; “Hello ” + “world!”
  ```
- !, !== Verneinung, Ungleich
  ```
  let variableDrei = 3;
  variableDrei !== 3;              // liefert false
  !(variableDrei === 3)    //liefert false
  ```

- **Anweisungen**
  - von recht nach links ausgewertet
  - Bsp:
    - Zahl = 42; // Zuweisung von Werten zu Variablen
    - Quadrat = Zahl * Zahl; // Durchführung einer Operation mit Variablen/Werten
    - if (Zahl > 1000) Zahl = 0; // bedingte Ausführung eines Befehls

- **Anweisungsblock**:
  - eine oder mehrere Anweisungen, die innerhalb einer übergeordneten Anweisung oder innerhalb eine Funktion stehen
  - gekennzeichnet durch geschweifte Klammer
  - Bsp:

```
if (Zahl > 1000) {
        Zahl = 0;
        Neustart();
}
```

```
function SageQuadrat (x) {
        var Ergebnis = x * x;
        alert(Ergebnis);
}
```

Bedingte Anweisungen (Auswahl)

- Anweisungsblock wird nur durchlaufen, wenn eine Bedingung erfüllt ist; Alternativ kann beim Nichterfüllen der Bedingung ein zweiter Anweisungsblock durchlaufen werden.
- Die Bedingung wird mithilfe von Vergleichsoperatoren formuliert.

```
if (Ausdruck) {
      //Anweisungsblock des True-Zweigs (wird ausgeführt, wenn Bedingung erfüllt ist)
      Anweisung;
      ...
  } else {
      //Anweisungsblock des False-Zweigs (wird ausgeführt, wenn Bedingung nicht
erfüllt ist)
      Anweisung;
      ...
  }
```

Schleifen (Auswahl)

- **while**
  - Programmanweisungen werden solange wiederholt, wie die Bedingung, die in der Schleife formuliert wird, erfüllt ist. Solche Schleifen eignen sich dann, wenn nicht bekannt ist, wie oft die Schleife durchlaufen werden soll. **Achtung**: Abbruchbedingung muss erfüllbar sein, sonst wird Endlosschleife erzeugt.

```
 while (Bedingung 1 && zaehler <= 3) {
       //Anweisungsblock (wird solange ausgeführt, wie die Bedingung
erfüllt ist)
        Anweisung;
    ...
    zaehler++;
   }
```

Schleifen (Auswahl)

- **for**
  - enthält eine Zählvariable, eine Fortführungsbedingung sowie eine Anweisung zur Änderung der Zählvariable

```
for (var i = 10; i <= 36; i++) {
  //Anweisungsblock (wird so oft ausgeführt, bis die Zählvariable den
Grenzwert erreicht hat)
  Anweisung;
  ...
}
```

Donnerstag, 22.09.22 - JavaScript - das HTML-DOM

- Beim Laden eines HTML-Dokuments im Browser steht dieses als **document object** für die Bearbeitung mit JS Code zur Verfügung. Dieses, wie auch die Objekte, die es beinhaltet, hat spezielle Eigenschaften und Methoden, mit denen gearbeitet und Änderungen am Dokument vorgenommen werden kann.
- Document Object
  - https://www.w3schools.com/jsref/dom_obj_document.asp
  - Beispiele: getElementById(), getElementsByClassName(), getElementsByTagName(), querySelector(): Methode liefert die HTML-Stellen mit den spezifizierten Namen/Klassen/IDs zurück
    - Aufruf: `document.querySelector(".initial");`
- Element object
  - https://www.w3schools.com/jsref/dom_obj_all.asp
  - repräsentiert die einzelnen HTML-Elemente (p, div, a, …)
  - Beispiel: hasAttribute(): gibt `true` zurück, wenn das Element das spezifizierte Attribut besitzt, andernfalls `false`
    - Aufruf: `document.querySelector('div').hasAttribute('class'));`
- Debugging fortgeschritten: API Console (weitere Methoden für die Konsolenausgabe):
  - https://www.w3schools.com/jsref/api_console.asp

FRAGEN ?

Donnerstag, 22.09.22 - PAUSE

12:30 - 12:45

Donnerstag, 22.09.22 - JavaScript - Übung 4

Übung:

Erweitern Sie Ihre Projektwebseite um eine Suchfunktion: …

Ziel: Grundlegendes Verständnis über Logik und Funktionsweise von JavaScript

Tutorials:

https://www.w3schools.com/js/default.asp (W3Schools)

https://wiki.selfhtml.org/wiki/JavaScript (deutsch)

https://developer.mozilla.org/en-US/docs/Web/JavaScript (umfangreicher Leitfaden, teilw. auf dt.)

JS-Grundlagen: Fabian Hiller
https://www.youtube.com/watch?v=3RqKS4AYE9s&list=PLsfa2ln4Ba7i77XaCVnTR7jogF1xP-Yjs

Donnerstag, 22.09.22 - JavaScript Übung 4 Teil 1

1. Ergänzen Sie Ihr **Suchfeld** in der Navigation-Bar um die ID **search_field**. *HTML-tag input Attribut: id*
2. Ergänzen Sie Ihre Navigation-Bar um einen **Button**, der das Attribut onclick besitzt und die Beschriftung “Suchen” trägt. *HTML-tag button Attribut: onclick*
3. Schreiben Sie eine Funktion **suchen()** in den JS-Block Ihres HTML-Dokuments. Die Funktion hat eine leere Parameterliste und der Anweisungsblock bleibt nocht leer.
4. Legen Sie innerhalb der Funktion eine lokale Variable searchField an. Diese soll dasjenige HTML-Element in Ihrem HTML ansprechen, das die ID **search_field** besitzt und dessen property **value**, also den Inhalt des Suchfelds, speichern.

Hilfreiche MDN-Refenrenzen:

- https://developer.mozilla.org/en-US/docs/Web/API/Document/getElementById
- https://www.w3schools.com/jsref/prop_text_value.asp

Donnerstag, 22.09.22 - PAUSE

13:30 - 14:45

Donnerstag, 22.09.22 - JavaScript Übung 4 Teil 2

1. Nutzen Sie die Methode **find()** des **window-Objekts** Ihres HTML-Dokuments, um den in der Variable seachValue gespeicherten string, also das im Suchfeld eingegebene Wort, auf der Webseite zu suchen.
2. Rufen Sie Ihre JS function **suchen()** im <body>-Teil Ihres HTML auf: Die Suchfunktion soll durchgeführt werden, wenn auf den **Button** geklickt wird.

Hilfreiche MDN-Refenrenzen:

- https://developer.mozilla.org/en-US/docs/Web/API/Window/find

FRAGEN ?

Schauen Sie sich die Folien und die MDN- bzw. W3C-Referenzen an und versuchen Sie, Ihre Webseite um weitere Funktionalitäten zu erweitern.

Tutorials:

https://www.w3schools.com/js/default.asp (W3Schools)

https://wiki.selfhtml.org/wiki/JavaScript (deutsch)

https://developer.mozilla.org/en-US/docs/Web/JavaScript (umfangreicher Leitfaden, teilw. auf dt.)

JS-Grundlagen: Fabian Hiller
https://www.youtube.com/watch?v=3RqKS4AYE9s&list=PLsfa2ln4Ba7i77XaCVnTR7jogF1xP-Yjs