getting-started.md

Getting Started with JavaScript

JavaScript im Browser

- Es gibt verschiedene Möglichkeiten js im Browser auszuführen
- Dabei wird jedoch immer ein `<script></script>` Tag verwendet
- **Das Script-Tag kann sowohl im Head als auch im Body auftauchen! Im Head, wird das Script ausgeführt, wenn noch nicht das gesamte HTML geladen wurde, im Body erst danach**

JavaScript direkt im Script Tag

- Der Inhalt zwischen dem Script Tag, wird als JavaScript interpretiert und ausgeführt
- Das wird allerdings in echten produktiven Anwendungen schnell unübersichtlich und deshalb wird davon abgeraten

```
<html>
    <head>
    </head>
    <body>
        <script>
            console.log("Hallo Welt");
        </script>
    </body>
</html>
```

JavaScript aus externer Datei laden

- Es ist außerdem auch möglich, eine externe Js Datei einzubinden
- Hierfür wird das `src`-Attribut des Script Tags benutzt
- Als Beispiel ist folgende Ordner Struktur gegeben:
  - my-project-folder/
    - index.html
    - my-js-file.js

```
<!-- index.html -->
<html>
    <head>
    </head>
    <body>
        <script src="/my-js-file.js"></script>
    </body>
</html>
```

```
// my-js-file.js
console.log("Hallo Welt");
```

- **Wenn du das `src`-Attribut verwendest, wird der Inhalt zwischen dem Script-Tag ignoriert

```
<!-- index.html -->
<html>
    <head>
    </head>
    <body>
```

```
        <script src="/my-js-file.js">
            console.log("Wird nicht ausgeführt, wegen src=!!!!");
        </script>
    </body>
</html>
```

Order of Operations

- Da man mit Js oft, den HTML content manipulieren will, ist es wichtig auf eine ordentliche Reihenfolge zu achten.
- Wie schon angemerkt, kann das unter anderem dadurch geschehen, ob das Script-Tag im Head oder Body eingebunden wird
- Eine weitere Möglichkeit sind die `defer` und `async` Attribute

Async

- Der Browser stoppt den Ladeprozess der Seite nicht, um den JS Content herunterzuladen
- Prozess wird jedoch zum parsen ("lesen") des Scripts gestoppt
- `<script async src="/script.js"></script>`

Defer

- Der Browser stoppt den Ladeprozess der Seite nicht, um den JS Content herunterzuladen
- Prozess wird auch nicht zum parsen ("lesen") des Scripts gestoppt
- Defer impliziert async, deshalb macht es keinen Sinn beide zusammen zu verwenden
- `<script defer src="/script.js"></script>`

JavaScript Statements

- Ein Statement ist wie ein vollständiger Satz in Js
- Kann zum Beispiel, das Deklarieren einer Variable sein, oder der Aufruf einer Funktion
- **Jedes Statement wird in Js mit einem Semikolon beendet**
- Wenn zwischen 2 Zeilen ein \n -> Newline Charakter (Enter drücken) auftaucht, wird von Js automatisch ein Semikolon erzeugt
- In diesen Fällen kann auf das Semikolon verzichtet werden, **davon rate ich aber eher ab, da es auch zu Problemen führen kann**

```
console.log("Semikolon am Ende der Zeile");
console.log("Es können auch zwei Statements in einer Zeile passieren"); console.log("2. Statement");

console.log("Kein Semikolon am Ende der Zeile (funktioniert nur, weil kein weiteres Statement in der gleichen
console.log("Nächstes Statement auf nächster Zeile")

// Folgende Zeile ist FALSCH
console.log("Funktioniert NICHT!! Da kein Semikolon") console.log("Test")
```

JavaScript Comments

- Ein Comment ist Text im Source Code, der nicht ausgeführt wird
- Kann zur Dokumentation und zum deaktivieren von Code genutzt werden

Single-Line Comment

- Zählt nur für aktuelle Zeile
- **Benutze "//"**

```
// console.log("wird nicht ausgeführt");
```

Multi-Line Comment

- Zählt für mehrere Zeilen, bis zum Ende des Comments
- /* für Start des Comments
- */ für ende des Comments

```
/*
console.log("Wird nicht ausgeführt");
*/
```