JavaScript Cheat Sheet

Variablen

Um eine Variable mit dem Namen `nummer` anzulegen, musst du folgendes schreiben:

```
var nummer;
```

Danach kannst du dieser Variablen Werte hinzufügen, in dem du diese mit `=` zuweist:

```
nummer = 3;
```

Du kannst diese zum Beispiel auch hoch zählen

```
nummer = nummer + 1;
```

oder eine andere Funktion aufrufen:

```
nummer = prompt("Gib eine Nummer ein.", "");
```

Vergesst vor allem nicht die Semikolons ;

If-Else Bedingungen

Mit `if` kannst du Bedingungen festlegen. Du kannst damit Befehle umsetzen wie: “Wenn die Nummer gleich 0 ist, dann möchte ich die Nummer um eine Zahl erhöhen.” Der Code dazu wäre dafür:

```
if (nummer == 0) {
    nummer = nummer + 1;
}
```

Etwas allgemeiner ausgedrückt, werden If-Bedingungen also so geschrieben:

```
if (BEDINGUNG) {
    BEFEHLE, falls BEDINGUNG wahr ist;
}
```

Man kann auch weitere Fälle abfragen. Wenn du zum Beispiel die Nummer unterschiedlich erhöhen möchtest, je nachdem, was Nummer ist, geht das so:

```
if (nummer == 0) {
    nummer = nummer + 1;

} else if (nummer == 1) {
    nummer = nummer + 2;

} else if (nummer == 2) {
    nummer = nummer + 3;
}
```

Falls du z.B. die Nummer erhöhen möchtest, wenn sie 0 ist, und sie in allen anderen Fällen eins abziehen möchtest, kannst du `else` benutzen.

```
if (nummer == 0) {
    nummer = nummer + 1;
} else {
    nummer = nummer - 1;
}
```

Beschreiben der Bedingungen

Du kannst die Bedingungen unterschiedlich beschreiben:

- `==` , wenn etwas *gleich* sein soll
- `!=` , wenn etwas *ungleich* sein soll
- `<=` , wenn etwas *kleiner oder gleich* sein soll
- `>=` , wenn etwas *größer oder gleich* sein soll
- `>` , wenn etwas *größer* sein soll
- `<` , wenn etwas *kleiner* sein soll

Außerdem kannst du auch mehrere Bedingungen abfragen:

- `&&` bedeutet *und*
- `||` bedeutet *oder*

Beispiele:

- Wenn die Nummer **größer** als 4 **oder gleich** 0 sein soll:

  ```
  if (nummer > 4 || nummer == 0) {
      ...
  }
  ```

- Wenn die Nummer **nicht** 1 **und** auch **nicht** 2 sein soll:
  `if (nummer != 1 && nummer != 2) { ... }`

Schleifen

Wenn du einen Befehl hast, der sehr oft aufgerufen werden soll, kannst du Schleifen benutzen.

```
for (var counter = 0; counter < 100; counter = counter + 1) {
    BEFEHL
}
```

In diesem Fall wird der `BEFEHL` 100x ausgeführt.

Als Erklärung, was genau passiert:

1. Wir setzen einen Counter auf 0 (`var counter = 0`).
2. Wir erhöhen den Counter um 1 (`counter = counter + 1`).
3. Wir führen den `BEFEHL` aus.
4. Wir schauen nach, ob der Counter kleiner 100 ist (`counter < 100`)
    - falls ja, wiederholen wir Schritt 2-4
    - falls nein, hören wir auf

Anmerkung: `counter = counter + 1` macht übrigens genau das gleich wie `counter++` .

Funktionen

Funktionen sind Codeabschnitte oder zusammengefasste Befehle. Wenn du gewisse Befehle hast, die du immer wieder ausführen willst, oder wenn du Befehle hast, die du über einen Button ausführen möchtest, benutzt du Funktionen:

```
function showName() {
    document.getElementById("name").textContent = "Lena";
}
```

Dabei ist `showName` der **Name** der Funktion. Der Inhalt oder die Befehle der Funktion sind in den geschweiften Klammern `{ }` .

Wenn du nun `showName()` irgendwo aufrufst, dann werden alle Befehle der Funktion ausgeführt.

Arrays

Du kannst dir Arrays vorstellen wie einen Schubladenschrank:

```
var zahlen = [7, 2, 4, 5];
```

Die Variable `zahlen` ist nun ein Schubladenschrank.

- In der ersten Schublade ist die Zahl 7.
- In der zweiten Schublade ist die Zahl 2.
- In der dritten Schublade ist die Zahl 4.
- In der vierten Schublade ist die Zahl 5.

```
var fruechte = ["apfel", "banane", "birne"];
```

Wäre entsprechend ein Array mit den Wörtern `apfel`, `banane` und `birne`.

Wenn ihr nun aus dem Schubladenschrank, also aus dem Array ein Element rausnehmen wollt, geht das mit eckigen Klammern:

```
var frucht = fruechte[1];
```

Würde bedeuten, dass ihr aus dem Array `fruechte` das 1. Elemente rausnehmt und in die Variable `frucht` reinschreibt. In `frucht` steht also nun `banane` drin.

Ihr könnt auch Sachen in die Schublade reintun bzw. Sachen in das Array schreiben:

```
zahlen[2] = 3;
```

Damit schreibt ihr in die 2. Schublade von Zahlen den Wert 3. Und das Array verändert sich von:

- [7, 2, 4, 5] zu
- [7, 2, 3, 5]

Vergesst nicht, dass bei Arrays das Zählen bei 0 anfängt.

JavaScript in HTML

Datei einbinden

Um eine Funktion aus einer JavaScript-Datei in deiner HTML-Datei zu benutzen, muss die HTML-Datei irgendwoher wissen, wo diese JavaScript-Datei liegt. Das macht ihr mit diesem Code:

```
<script src="PFAD_ZUR_JAVASCRIPT_DATEI"></script>
```

Wenn ihr eine JavaScript-Datei habt, die z.B. `code.js` heißt und im gleichen Ordner liegt, schreibt ihr:

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

Wenn die JavaScript-Datei im Unterordner `js` liegt, schreibt ihr:

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

JavaScript-Funktion aus Datei einbinden

Wenn ihr mit einem Button eine JavaScript-Funktion aufrufen wollt, könnt ihr das so machen:

```
<button onclick="FUNKTIONSNAME()">Drück mich</button>
```

Als Beispiel: Wenn ihr eine Funktion, die `showName()` heißt und ungefähr so aussieht:

```
function showName() {
    name = prompt("Gib deinen Namen ein", "");
}
```

dann müsst ihr im Button folgendes aufrufen:

```
<button onclick="showName()">Drück mich</button>
```