JavaScript Grundlagen - CheatSheet

1. Variablen

In Variablen werden beliebige Werte gespeichert, um sie später wieder zu verwenden.

Konstante Variablen

```
const myVar = <Wert>;
```

So deklarierte Variablen behalten ihren `<Wert>` und können nicht geändert werden.

Veränderbare Variablen

```
let myVar = <Wert>;
```

So deklarierte Werte können ihren `<Wert>` beliebig oft ändern.

Variablen können verschiedene Werte annehmen.

- Text: `"ein wenig text"`
- Zahlen: `1` oder `5.32`
- Arrays: `[1, 2, 3, 4]`
- Objekte: `{a: "text", b: 1}`

2. Bedingungen

Soll unterschiedlicher Code abhängig von einer Bedingung ausgeführt werden, gibt es zwei Möglichkeiten.

If, Else If, Else

```
if (<Bedingung>) {
        <Aktion 1>
} else if (<andere Bedingung>) {
        <Aktion 2>
} else {
        <Aktion 3>
}
```

Zahlen vergleichen mit:

- kleiner als (`<`)
- kleiner gleich (`<=`)
- gleich (`==`)
- ungleich (`!=`)
- größer gleich (`>=`)
- größer als (`>`)

Texte vergleichen mit:

- gleich (`===`)
- ungleich (`!==`)

Switch

```
switch (<Variable>) {
        case <Fall 1>:
                <Aktion 1>
                break;
        case <Fall 2>:
                <Aktion 2>
                break;
        default:
                <Aktion 3>
}
```

Die Aktion in einem `case` wird genau dann ausgeführt, wenn `<Variable> == <Fall x>`.

3. Schleifen

Mehrfaches Ausführen des gleichen Codes geht durch Schleifen.

For-Schleife

```
for (let i = 0; <Bedingung>; <i ändern>) {
        <Aktion>
}
```

For-Schleifen haben feste Grenzen.

While-Schleife

```
let i = 0;
while (<Bedingung>) {
        <Aktion>
        <i ändern>
}
```

While-Schleifen können ihre Grenzen ändern.

4. Funktionen

Gleicher Code, der an verschiedenen Stellen wiederverwendet werden soll, sollte in einer Funktion zusammengefasst werden.

```
function <Name>() {
        <Aktion>
}
```

Im weiteren Code kann diese Funktion dann mit `<Name>();` aufgerufen werden. Soll die Funktion Parameter übergeben bekommen, geht das so:

```
function <Name>(<parameter 1>, <parameter 2>) {
        <Aktion>
}
```

Im weiteren Code kann diese Funktion dann mit `<Name>(<a>, <b>);` aufgerufen werden. Soll die Funktion einen Wert zurückgeben, geht das so:

```
function <Name>() {
        <Aktion>
        return <Wert>;
}
```

Im weiteren Code kann diese Funktion dann mit `let myVar = <Name>();` aufgerufen werden.