ES6/ESNext Cheat Sheet für die Webentwicklung

Hier die wichtigsten Funktionalitäten ab ECMAScript 6_

1. Let und Const

`let` und **`const`** bieten Block-Scope und ersetzen weitgehend **`var`**. **`const`** bietet Unveränderbarkeit des Werts zwecks Fehlervermeidung und Laufzeit-Optimierung.

```
let a = 10;
const b = 20;
```

2. Arrow Functions

Kürzere Syntax für Funktionen. Erhalten automatisch den **`this`**-Kontext.

```
const add = (a, b) => a + b;
```

3. Template Literals

Verwendet Backticks (`) für Zeichenketten mit JS-Ausdrücken/Variablen und mehrzeiligen Strings.

```
const name = "Max";
const greeting = `Hello, ${name}!
Very nice to meet you!`;
```

4. Standard-Parameterwerte

Parameter können Standardwerte haben, wenn kein Argument übergeben wird.

```
function greet(name = "guest") {
  return `Hello, ${name}`;
}
```

5. Destructuring

Einfaches Extrahieren von Werten aus Arrays oder Objekten.

Beispiel (Array):

```
const [x, y] = [1, 2];
```

Beispiel (Objekt):

```
const {name, age} = {name: 'Max', age: 30};
```

6. Spread Operator

Einfaches "Entpacken" von Arrays oder Objekten.

Beispiel (Array):

```
const arr1 = [1, 2];
const arr2 = [...arr1, 3, 4];
```

Beispiel (Objekt):

```
const obj1 = {a: 1, c: "t" };
const obj2 = {...obj1, b: 2};
```

7. Rest Operator

Erfassung mehrerer Funktions-Parameter in einem Array.

```
function sum(initialValue, ...numbers) {
  return numbers.reduce((acc, num) => acc + num, initialValue);
}
```

8. Promises

Eine elegantere Möglichkeit, mit asynchronem Code umzugehen. Voraussetzung für **`async/await`**.

```
function delay(time) {
  return new Promise((resolve, reject) => {
    if (time < 0) {
      reject("Invalid time!");
    }

    setTimeout(() => {
      resolve("Timeout finished!");
    }, time);
  });
}
```

9. Async/Await

Einfachere Schreibweise für asynchrone Operationen die Promises verwenden.

```
async function fetchData() {
  const response = await fetch("https://url");
  const data = await response.json();
  return data;
}
```

10. Modules (Import/Export)

Teile Code in separate Dateien auf und importiere/exportiere Funktionen und Variablen.

Export:

```
export const myVar = 42;
export function myFunc() {}
```

Import:

```
import * as myModule from "./myModule";
import { myVar, myFunc } from "./myModule";
```