Cheat Sheet – Syntax Changes

Strict Mode

Using strict mode is the default in ES6 Modules, therefore specifying “use strict” is not required.

let & const

In ES6, you may declare your variables with **var**, **let** or **const**. Which one should you choose?
const should be chosen if you’re not planning on changing the value of this variable (which then becomes a constant). This should be your default choice unless you absolutely need to re-assign it during runtime. Why should it be the default? Because it keeps your code explicit and clear.
let should be your second choice – use it whenever you need to re-assign variables. This will mostly be the case in mathematical alogrithms or loops.
var should probably never be chosen. When using it, you may have a constant or a variable – but no one knows! Therefore, use **const** or **let** instead.

Function / Arrow Functions

There were two major additions regarding functions: (Fat) **Arrow Functions** (() => {}) and **default arguments** (doSmth(arg = 1)).

Fat Arrow Functions allow you to use a shorter syntax to create functions:

```
const fn = (arg1, arg2) => return arg1 + arg2
```

You may leave out the parenthesis around the arguments if only one argument is passed. If no argument is passed, empty parentheses are required. The function body may be written inline and without curly braces if you’re only writing a return statement (return then also may be left out). Besides the shorter syntax, fat arrow functions also change the behavior of the **this** keyword. Inside a fat arrow function, the **this** keyword will always refer to the context of the function instead of the caller of the function.

Default parameters allow you specify default values for parameters passed to functions.

```
function fn(arg1 = ‘default string’, arg2 = 23) { return arg1 + arg2 }
```

More info on Arrow Functions:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Arrow_functions

More info on Default Parameters:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Default_parameters

Literal Notation Extensions

There are some improvements to Literal Notation syntax.

```
let width = 10;
let length = 50;

let lengthField = 'length';
rectangle = {
   width, // width is automatically assigned => takes above value
   [lengthField]: length, // Dynamic field name using variable
   "outputSize"() { // shorter syntax + possibility of string name
      console.log(this.width * this.length); // this refers to object
   }
};
```

More information may be found here: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Object_initializer

Rest & Spread Operators (...)

ES6 added two important new operators: Rest and Spread. Technically they look the same (... => three dots) but they are used in different places.

Rest:

```
function sumUp(start, ...toAdd) {}
```

Transforms a list of arguments (1, 2, 3) into an array ([1, 2, 3]) which may be used inside the function. This behavior is triggered when used inside of a function argument list.

Spread:

```
let ids = [1, 2, 3, 4, 5, 6];
console.log(Math.max(...ids)); // prints 6
```

Transforms an array into a list of arguments. This behavior is triggered when used outside of a function argument list.

More information on Rest:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/rest_parameters
More information on Spread:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Spread_operator

For-of Loop

A new loop was added: The for-of loop which allows you to loop through an array of items.

```
let testResults = [5.30, 2.84, 9.11, 1.01, 3.99];


for (let testResult of testResults) {
   if (testResult > 5) {
      console.log(testResult); // prints 5.30, 9.11
   }
}
```

More information may be found here:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Statements/for...of

Template Literals

Template literals allow you to easily write more complex text in strings (e.g. including line breaks) as well as to output variables in a string.

```
let name = 'Max';
description = `
     This is a description, which may wrap across multiple lines.
     Written by ${name}.
`;
```

Template literals are used by starting and ending the string with backticks (`) instead of quotation marks.
More information can be found here:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Template_literals

Destructuring

Destructuring is a cool new feature which allows you to easily extract values from an object or an array.
Array:

```
let numbers = [1, 2, 3, 4, 5];
let [a, b] = numbers; // a => 1, b => 2
```

Object:

```
let person = {
    name: 'Max',
    age: 27
};
let {name, age} = person; // Notice the {} instead of []
```

More information may be found here:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment