Cheat Sheet – Basics

Executing JS

JavaScript can be executed in different places. It always needs to be placed within <script> tags though.
Option 1: Code between <script> tags

```
<script>console.log('Hello there!')</script>
```

JavaScript code will always be executed once the browser reaches this <script> tag when loading the page. Therefore it is a good practice to place <script> tags at the bottom of the <body> section (right before </body>) to make sure, that the loading of the page isn't blocked by long-running scripts.
Option 2: External File

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

Still, the placement of <script> matters, since the file will be loaded and the script inside the file will be exectuted once the browser reaches the <script> tag.
Also, don't forget the closing tag, the following Syntax will not work:

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

The same is true for this syntax:

```
<script src="script.js">console.log('Hello there!')</script>
```

Only the code in the script file will be executed in this case.

Besides the browser, NodeJS (a server-side language) is also using JavaScript and therefore can execute JavaScript code.

Variables & Types

JavaScript ships with different data types, though it's no strongly typed language. That means, that you don't explicitly assign a variable to be a specific type. Instead, you may switch types as you like (but you shouldn't do that in most cases).
You create a variable using the *var* keyword:

```
var myVariable = 'a string variable';
```

This would create a variable, initialized with a string. You don't have to initialize a variable immediately, you can also only declare it:

```
var onlyDeclaredVariable;
```

In JavaScript you can even set a variable (myVar = 5) before declaring it. This behavior is called **hoisting**. When executing JavaScript code, all your declarations will get pulled to the top of the file/ code and therefore you may have a different order when writing your code.

Notice the semicolon at the end of the line? That's optional but it is a good practice to end your lines with a semicolon. You can also enforce this (and get warnings if your forget it) by adding this "text" to the top of your file/ code:

```
"use strict"
```

This enforces the strict mode which allows browsers to run your code more efficiently.
Learn more about strict mode here:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Strict_mode

As types, you have couple of possibilities available:

- String (text)
- Number (both integer and floating point numbers)
- Boolean (true or false)
- Array
- Null
- Undefined
- NaN (Not a Number)
- Object

Functions

Functions are not executed immediately but have to be called. Therefore, you may put code into functions which you don't want to execute by default but which instead should be called at a time of your choice/ during runtime.
Functions also take arguments to work with and may return a value.

```
function fn(number) {
    return number * 2;
}
```

You can also write functions like this:

```
var fn = function(number) {
    return number * 2;
}
```

A function may then be called like this:

```
console.log(fn(10)); // prints 20 to the console
```

Functions have their own type: Function.

Control Structures

Most of the time you don't want to execute your code from top to bottom. You may only want to execute parts of your code depending on some conditions or loop through some code to execute it more often than once. Control Structures help you with that.

In JavaScript you have the following Control Structures available:

if

```
if (condition == true) {}
```

switch

```
switch (value) {
    case 5:
        // do something
        break;
    case 10:
        …
    default:
        …
}
```

for

```
for (var i = 0; i < 5; i++) {}
```

while

```
while (i < 5) {}

// Alternative, execute code first

do {} while (i < 5);
```

There also exists a for...in loop which will be explained in the Objects section.

Operators

Of course you need operators to really work with data. JavaScript includes mathematical operators (addition, subtraction, ...) as well as comparison operators (==, <, ...). And some more.

Learn more about JavaScript operators here:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Expressions_and_Operators
http://www.w3schools.com/js/js_operators.asp

Learn more about operator precedence here:
https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Operators/Operator_Precedence