Chapter 08

Introduction to **Web Design**

JS

Basic Javascript

Prepared by **Hok Tin**

Content

- Introduction to Javascript
- Basic Javascript

Introduction to Javascript

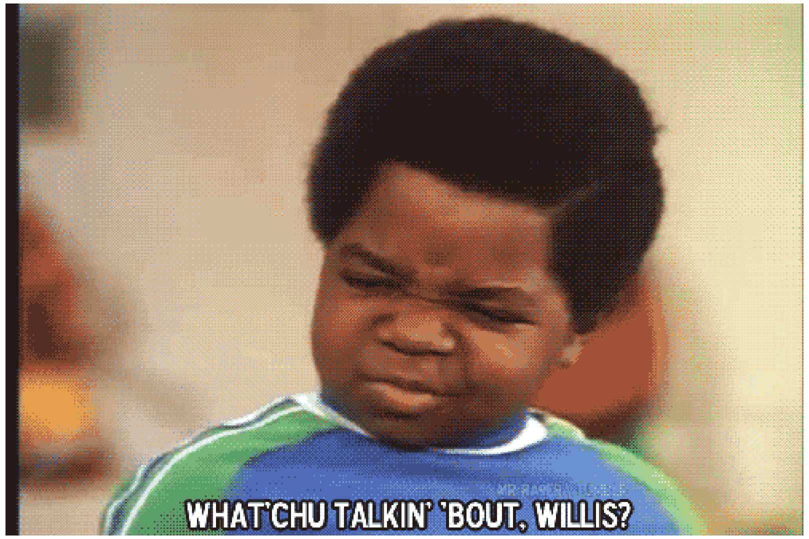
❑ What is JavaScript

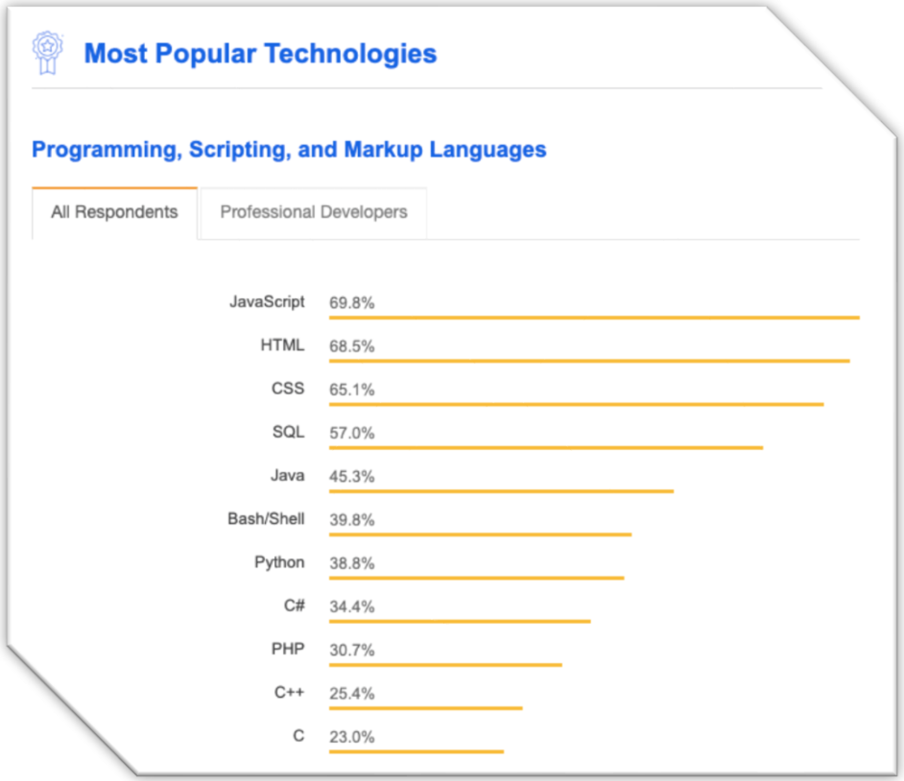
For now, let's simply say that JavaScript is a tool for developers to add interactivity to websites

See, your browser needs three things for allowing you to consume this content:

- HTML structures the content
- CSS styles it

JavaScript *makes your page come to life!*




Most Popular Technologies

Programming, Scripting, and Markup Languages

All Respondents | Professional Developers

JavaScript	69.8%
HTML	68.5%
CSS	65.1%
SQL	57.0%
Java	45.3%
Bash/Shell	39.8%
Python	38.8%
C#	34.4%
PHP	30.7%
C++	25.4%
C	23.0%

Introduction to Javascript

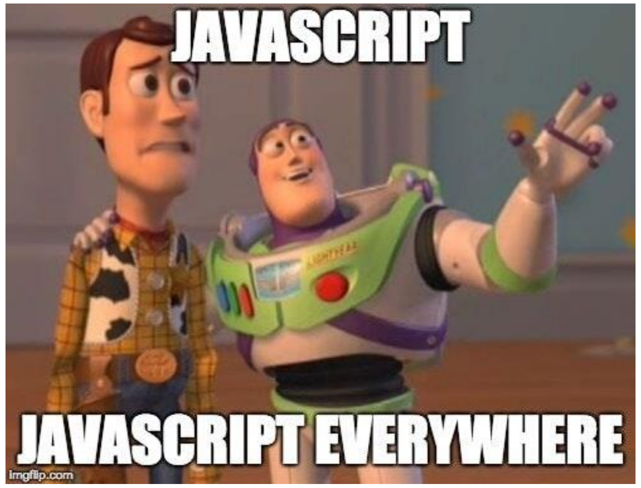
❑ Your future with Javascript

You might have heard of **Angular**, backed by Google, and **React**, backed by FB. Also we have to mention **Vue** and **Svelte** here that, even if not supported by a tech powerhouse, completes the triad of important JS frameworks.

...

The language of... many, many things

Server-side code
Mobile apps
API integrations
Web frontend



Introduction to Javascript

❑ What Javascript can do?

- JavaScript can *Change HTML Content*

```
document.getElementById("demo").innerHTML = "Hello JavaScript";
```

- JavaScript can *Change HTML Attribute Values*

```
document.getElementById("demo").setAttribute("class", "democlass");
```

- JavaScript can *Change HTML Styles (CSS)*
- JavaScript can *Show HTML Elements*

```
document.getElementById("demo").style.display = "none";
```

- And many more ….

Introduction to Javascript

❑ How to write your Javascript in HTML

- The **<script>** Tag : In HTML, JavaScript code is inserted between **<script>** and **</script>** tags.

index.html

```html
<!DOCTYPE html>
<html>
<body>

  <h2>Use JavaScript to Change Text</h2>
  <p>This example writes "Hello JavaScript!" into an HTML element with id="demo":</p>
  <p id="demo"></p>

  <script>
    document.getElementById("demo").innerHTML = "Hello JavaScript!";
  </script>
</body>
</html>
```

- You can place any number of scripts in HTML document, either in **<head>** or **<body>**
- **External javascript**: External scripts are practical when the same code is used in many different web pages. JavaScript files have the file extension **.js**

index.html

```html
<!DOCTYPE html>
<html>
<head>
  <script src="myScript.js"></script>
</head>
<body>
  <h2>Use JavaScript to Change Text</h2>
  <p>This example writes "Hello JavaScript!"
     into an HTML element with id="demo":
  </p>
  <p id="demo"></p>
</body>
</html>
```

myScript.js

```javascript
document.getElementById("demo").innerHTML = "Hello JavaScript!";
```

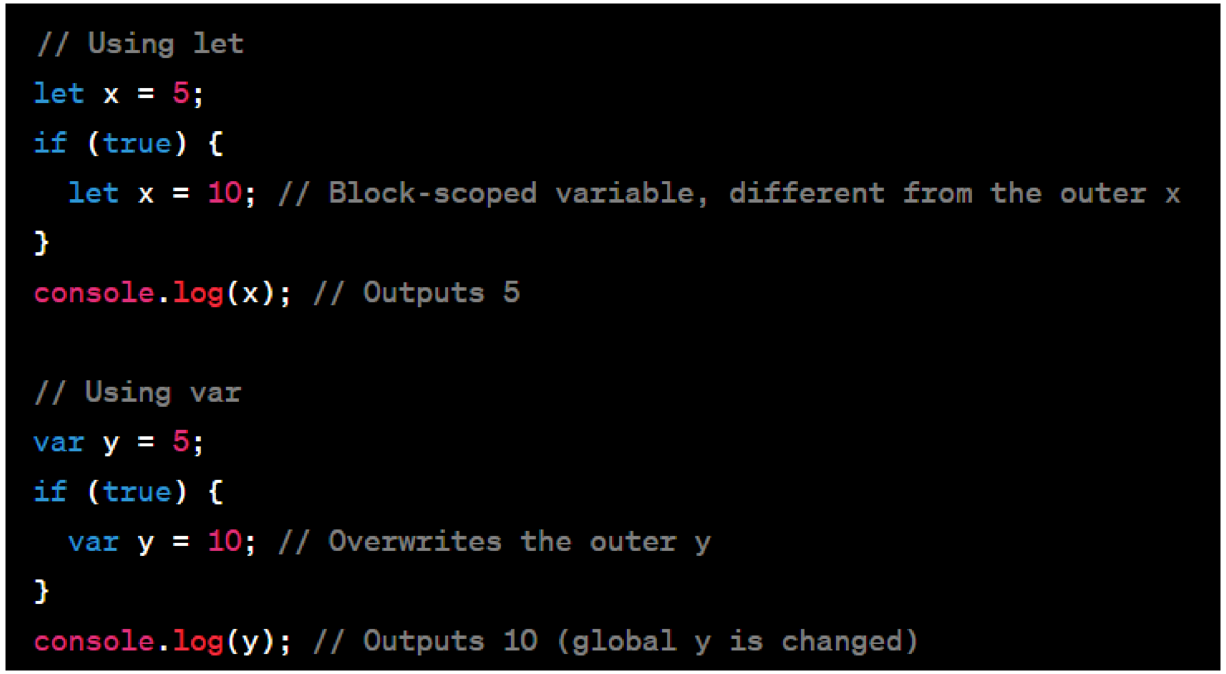
Basic Javascript

❑ **Variable**

A variable is a “named storage” for data. We can use variables to store goodies, visitors, and other data.

You can either use let or var to declare a variable

So what is the different between let and var?



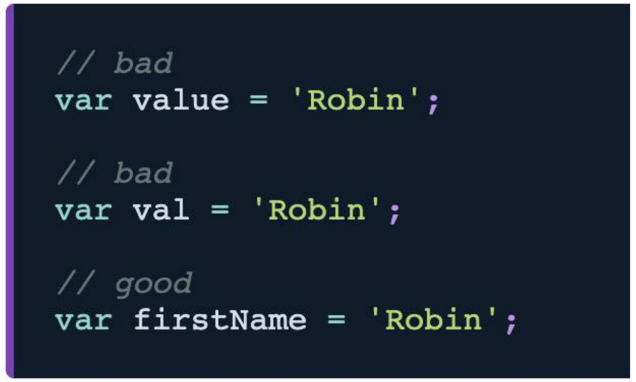
```
// Using let
let x = 5;
if (true) {
  let x = 10; // Block-scoped variable, different from the outer x
}
console.log(x); // Outputs 5

// Using var
var y = 5;
if (true) {
  var y = 10; // Overwrites the outer y
}
console.log(y); // Outputs 10 (global y is changed)
```

Basic Javascript

❑ Variable Naming & Naming Convention

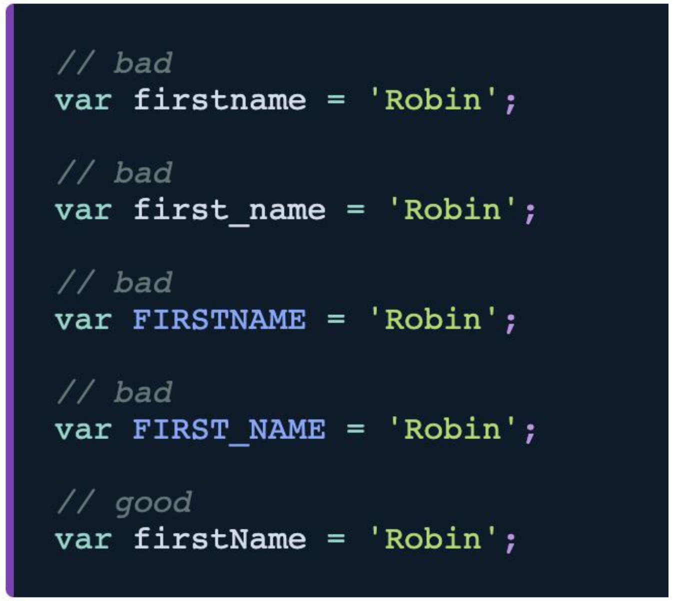
Variable naming and naming conventions are important in programming for several reasons:

- ✓ Readability and Maintainability:
- ✓ Self-documentation
- ✓ Preventing Bugs
- ✓ Collaboration
- ✓ Avoiding Conflicts
- ✓ Etc.



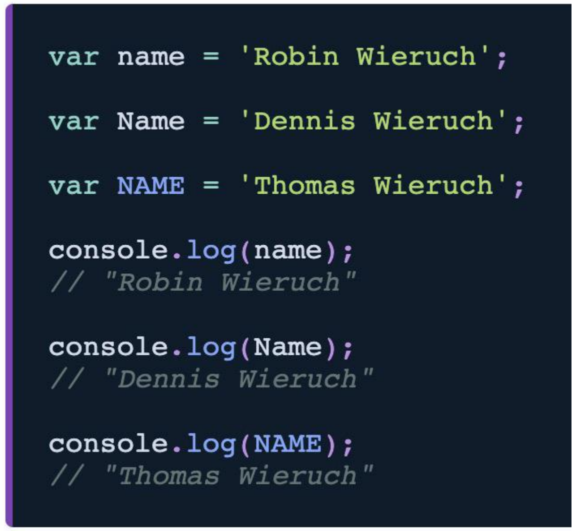
```
// bad
var value = 'Robin';

// bad
var val = 'Robin';

// good
var firstName = 'Robin';
```

Meaningful



```
// bad
var firstname = 'Robin';

// bad
var first_name = 'Robin';

// bad
var FIRSTNAME = 'Robin';

// bad
var FIRST_NAME = 'Robin';

// good
var firstName = 'Robin';
```

camelCase recommended



```
var name = 'Robin Wieruch';

var Name = 'Dennis Wieruch';

var NAME = 'Thomas Wieruch';

console.log(name);
// "Robin Wieruch"

console.log(Name);
// "Dennis Wieruch"

console.log(NAME);
// "Thomas Wieruch"
```

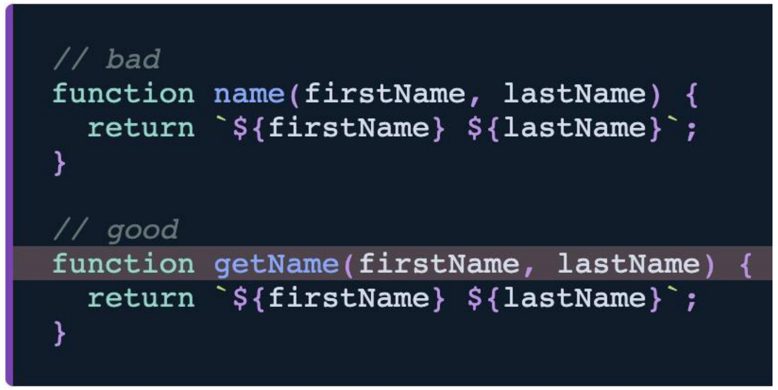
Case Sensitive

Basic Javascript

❑ Variable Naming & Naming Convention

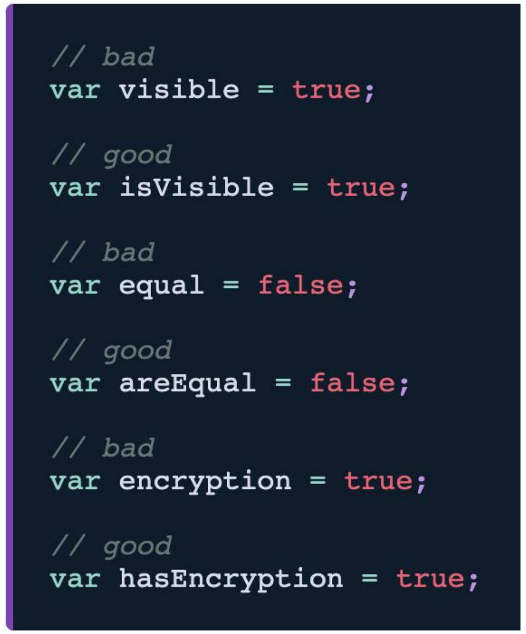
Variable naming and naming conventions are important in programming for several reasons:

- ✓ Readability and Maintainability:
- ✓ Self-documentation
- ✓ Preventing Bugs
- ✓ Collaboration
- ✓ Avoiding Conflicts
- ✓ Etc.



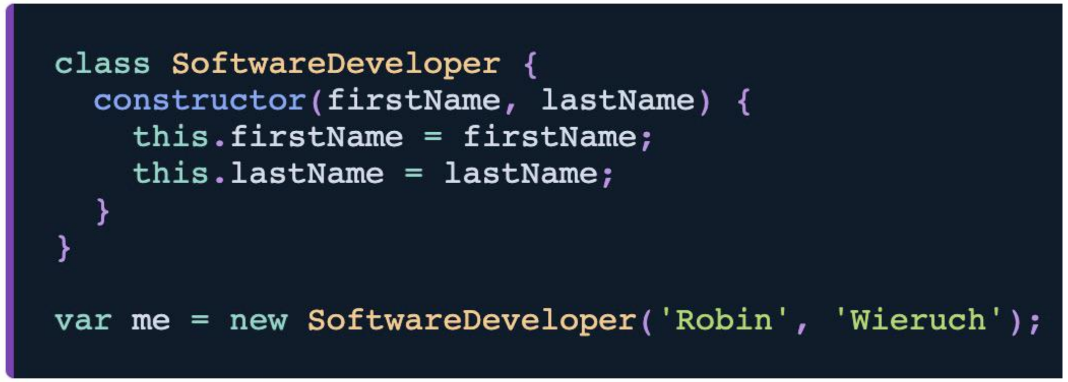
```
// bad
function name(firstName, lastName) {
  return `${firstName} ${lastName}`;
}

// good
function getName(firstName, lastName) {
  return `${firstName} ${lastName}`;
}
```

How you should name function



```
// bad
var visible = true;

// good
var isVisible = true;

// bad
var equal = false;

// good
var areEqual = false;

// bad
var encryption = true;

// good
var hasEncryption = true;
```

How you should name boolean variable



```
class SoftwareDeveloper {
  constructor(firstName, lastName) {
    this.firstName = firstName;
    this.lastName = lastName;
  }
}

var me = new SoftwareDeveloper('Robin', 'Wieruch');
```

Use PascalCase to name class

Basic Javascript

❑ Types of types

Types in JavaScript group together similar kinds of values. We can further categorize these types into the following:

- **Primitive values**
- **Objects and functions**

Following are all types that fall under the ***primitive category***:

- **Boolean**

```
var bool_true = true;
var bool_false = new Boolean(false);
```

- **Number**

```
var num1 = 120;
var num2 = new Number(0.002);
```

- **Undefined**

```
var num1;
var num2 = undefined;
```

- **String**

```
var str1 = "String-1";
var str2 = new String("String-2");
var str3 = new String(1234);
```

How to concatenate two strings

```
var str1 = "Str1";
var str2 = new String("Str2");
var str1_2 = `str1:${str1} and str:${str2}`;
```

- **Null**

```
var num1 = null;
console.log('The type is:',typeof(num1));
```

- **bigint** is for integer numbers of arbitrary length
- **symbol** for unique identifiers.

Use **typeof()** to get data type of a variable

Basic Javascript

❑ Work with String

- You can write string with:

```
let single = 'single-quoted';
let double = "double-quoted";
let backticks = `backticks`;



function sum(a, b) {
    return a + b;
}

alert(`1 + 2 = ${sum(1, 2)}.`); // 1 + 2 = 3.
```

- Use **+** to concatenate strings

```
let s = "my" + "string";
alert(s); // mystring
```

- Use **length** property to get string length

```
alert( `My\n`.length ); // 3
```

- Use backslash \ to mask special character

```
alert( 'I\'m the Walrus!' ); // I'm the Walrus!

alert( "\u00A9" ); // ©
alert( "\u{20331}" ); // 𠌱, a rare Chinese hieroglyph (long unicode)
alert( "\u{1F60D}" ); // 😍, a smiling face symbol (another long unicode)

let str1 = "Hello\nWorld"; // two lines using a "newline symbol"
```

Basic Javascript

❑ Work with String

- To get a character at position pos

```
let str = `Hello`; // the first character
alert( str[0] ); // H
```

- Change the case: toLowerCase() and toUpperCase()

```
alert( 'Interface'.toUpperCase() ); // INTERFACE
alert( 'Interface'.toLowerCase() ); // interface
```

- Searching for substring with str.indexOf(substr, pos)

```
let str = 'Widget with id’;
alert( str.indexOf('Widget') ); // 0, because 'Widget' is found at the beginning
alert( str.indexOf('widget') ); // -1, not found, the search is case-sensitive
alert( str.indexOf("id") ); // 1, "id" is found at the position 1 (..idget with id)
```

- Get substring with str.slice(start [, end]) or str.substring(start [, end]) or str.substr(start [, length])   []: optional

```
let str = "stringify";
alert( str.slice(0, 5) ); // 'strin', the substring from 0 to 5 (not including 5)
alert( str.slice(0, 1) ); // 's', from 0 to 1, but not including 1, so only
character at 0
```

Basic Javascript

❑ Work with Number

The following math operations are supported:

- Addition +
- Subtraction -
- Multiplication *
- Division /
- Remainder %
- Exponentiation **

```
alert( 5 % 2 ); // 1, a remainder of 5 divided by 2
alert( 8 % 3 ); // 2, a remainder of 8 divided by 3

alert( 2 ** 2 ); // 4 (2 multiplied by itself 2 times)
alert( 2 ** 3 ); // 8 (2 * 2 * 2, 3 times)
alert( 2 ** 4 ); // 16 (2 * 2 * 2 * 2, 4 times)
```

Basic Javascript

❑ Work with Functions

Functions are the main “building blocks” of the program. They allow the code to be called many times without repetition.

```
function SetName(parameter, otherParam) {
    ...body...
}
```

```
function showMessage() {
    alert( 'Hello everyone!' );
}
```

- You can set default value to a parameter

```
function showMessage(from, text = "no text given") {
    alert( from + ": " + text );
}

showMessage("Ann"); // Ann: no text given
```

- A function can return a value back into the calling code as the result

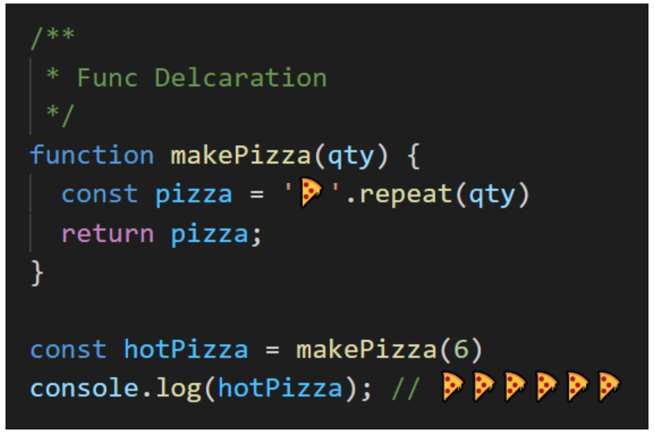
```
function sum(a, b) {
   return a + b;
}

let result = sum(1, 2);
alert( result ); // 3
```

Basic Javascript

❑ Work with Functions

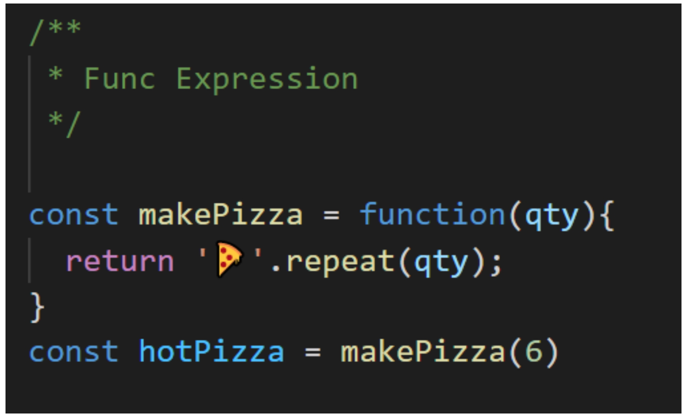
Different ways to define your functions (tips)

- Function expression



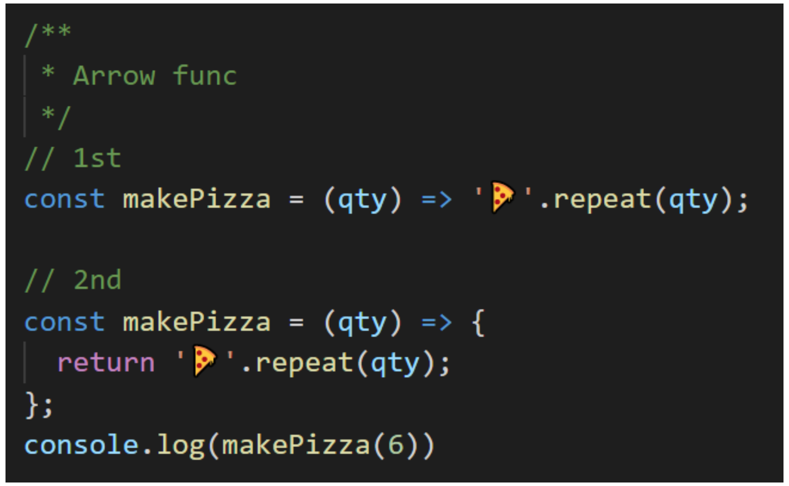
```
/**
 * Func Delcaration
 */
function makePizza(qty) {
  const pizza = '🍕'.repeat(qty)
  return pizza;
}

const hotPizza = makePizza(6)
console.log(hotPizza); // 🍕🍕🍕🍕🍕🍕
```

- Function expression



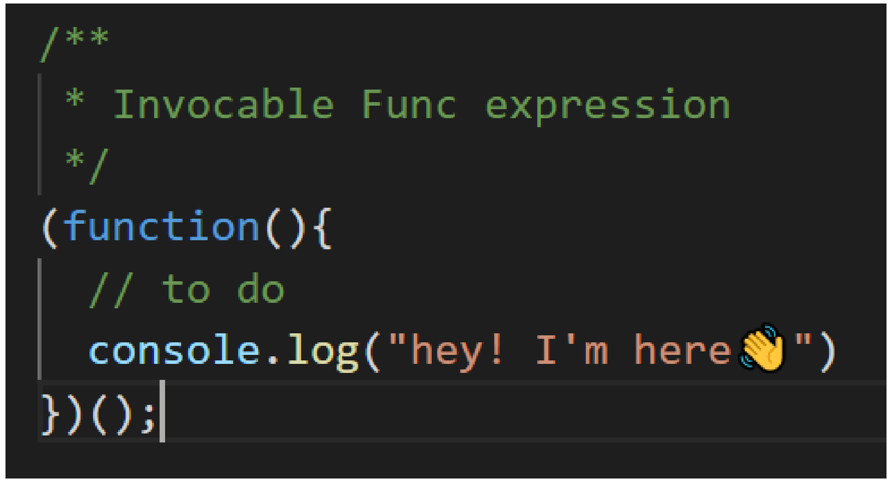
```
/**
 * Func Expression
 */

const makePizza = function(qty){
  return '🍕'.repeat(qty);
}
const hotPizza = makePizza(6)
```

- Arrow function



```
/**
 * Arrow func
 */
// 1st
const makePizza = (qty) => '🍕'.repeat(qty);

// 2nd
const makePizza = (qty) => {
  return '🍕'.repeat(qty);
};
console.log(makePizza(6))
```

- Invocable Function expression



```
/**
 * Invocable Func expression
 */
(function(){
  // to do
  console.log("hey! I'm here👋")
})();
```

Basic Javascript

❑ Control Statement

The **"If"** statement

```
let year = prompt('In which year was ECMAScript-2015 specification published?', '');
if (year == 2015) alert( 'You are right!' );
```

A **number 0**, an **empty string** "", **null**, **undefined**, and **NaN** all ➔ become **false**.
Because of that they are called "falsy" values.

- The "**If**/**else**/**else if**" statement

```
let year = prompt('In which year was the ECMA2015 specification published?', ’’);
if (year < 2015) {
   alert( 'Too early...' );
}
else if (year > 2015) { alert( 'Too late' ); }
else { alert( 'Exactly!' ); }
```

- The conditional operator "**?**"

```
let result = condition ? value1 : value2;
```

Basic Javascript

❑ Work with Array

- Declaring empty arrays

```
var arr = new Array(); // Assign arr an empty array object
var arr2 = []; // Assign an empty array to arr2
var arr3 = new Array(12) // Assign an array of 12 size to arr3
console.log(arr, arr2, arr3) // Print array

// Declaring arrays with elements
var arr1 = new Array(1, -2, "3"); // Create an array object and assign to arr1
var arr2 = [4, 5, "6", true]; // Create an array object and assign to arr2
console.log(arr1,arr2)

// Elements in an array
var first = arr1[0]; // access the element by index
arr1[0] = 5; // assign 5 to arr1 index 0
```

- Copying arrays: arr.slice(), [...arr] , Array.from(arr)

```
var arr1 = [1, 2, 3, 4]; // Assign an array to arr1
var arr2 = arr1.slice(); // Assign arr2 to copy of arr1
var arr3 = [...arr1]; // Assign arr3 to copy of arr1
var arr4 = Array.from(arr1); // Assign arr4 to copy of arr1
arr2[0] = 5; // assign 0 index of arr2 to 5
arr3[0] = 6; // assign 0 index of arr3 to 6
arr4[0] = 7; // assign 0 index of arr4 to 7
console.log(arr1, arr2, arr3, arr4); // Print all arrays
```

Basic Javascript

❑ Work with Array

- Loop with array

```
let arr = ["Apple", "Orange", "Pear"];
for (let i = 0; i < arr.length; i++) {
  alert( arr[i] );
}
```

- With a more elegant way

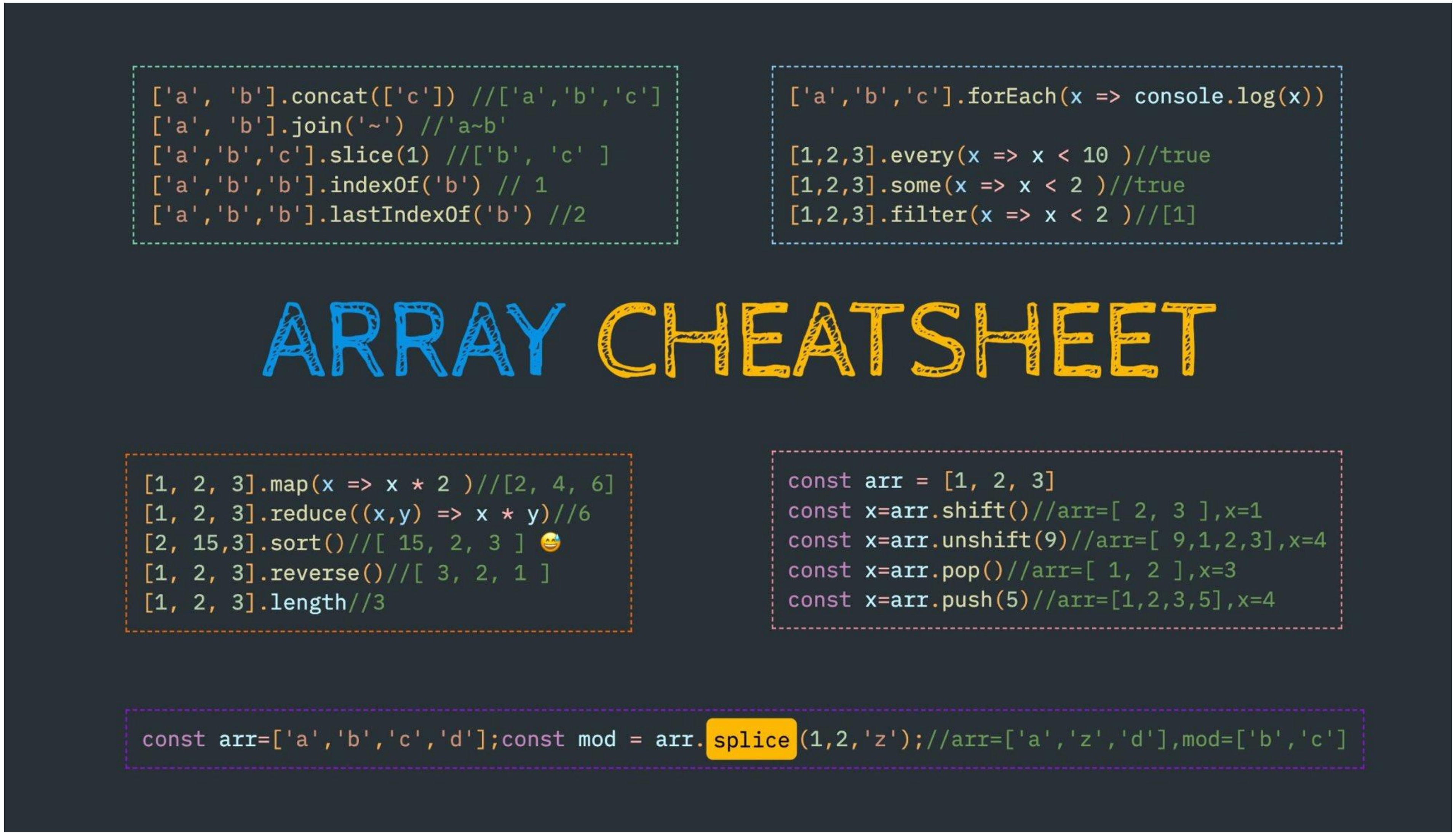
```
let fruits = ["Apple", "Orange", "Plum"];

// iterates over array elements
for (let fruit of fruits) {
   alert( fruit );
}
```

- You can also use:

```
let arr = ["Apple", "Orange", "Pear"];

for (let key in arr) {
   alert( arr[key] ); // Apple, Orange, Pear
}
```

Basic Javascript

❑ **More methods of Array**



```
['a', 'b'].concat(['c']) //['a','b','c']
['a', 'b'].join('~') //'a~b'
['a','b','c'].slice(1) //['b', 'c' ]
['a','b','b'].indexOf('b') // 1
['a','b','b'].lastIndexOf('b') //2
```

```
['a','b','c'].forEach(x => console.log(x))

[1,2,3].every(x => x < 10 )//true
[1,2,3].some(x => x < 2 )//true
[1,2,3].filter(x => x < 2 )//[1]
```

ARRAY CHEATSHEET

```
[1, 2, 3].map(x => x * 2 )//[2, 4, 6]
[1, 2, 3].reduce((x,y) => x * y)//6
[2, 15,3].sort()//[ 15, 2, 3 ] 😅
[1, 2, 3].reverse()//[ 3, 2, 1 ]
[1, 2, 3].length//3
```

```
const arr = [1, 2, 3]
const x=arr.shift()//arr=[ 2, 3 ],x=1
const x=arr.unshift(9)//arr=[ 9,1,2,3],x=4
const x=arr.pop()//arr=[ 1, 2 ],x=3
const x=arr.push(5)//arr=[1,2,3,5],x=4
```

```
const arr=['a','b','c','d'];const mod = arr.splice(1,2,'z');//arr=['a','z','d'],mod=['b','c']
```

Good luck 👌

References

1. https://snipcart.com/blog/why-javascript-benefits
2. https://www.reddit.com/r/learnjavascript/comments/kt4wd4/javascript_array_functions_cheat_sheet_as_asked/