Commands:

`Shift + Opt + F` - format the Document in VS Code
`Shift + Opt + ↑ / ↓` - duplicate line in up or down direction
`fn/option` - write text in multiple places
`Opt + Command + J` - open Console (javascript) in Browser
`Inspect + Sources` - open Stack

Links:

- https://getbootstrap.com/docs/5.3/getting-started/introduction/ - Bootstrap

Section 3: HTML: The Essentials

Essentials:

```HTML
<b>...</b> - bold text
<p>...</p> - paragraph
<h1-h6>...</h1-h6> - headings
<a href="link.com or .html (for the file)"><!-- Name of a link --></a> - hyperlink
<!--...--> - comment
```

HTML Sceleton:

```HTML
! - create a HTML Sceleton

<!DOCTYPE html>
<html>
<head>
    <title> My page </title>
</head>
<body>
    <!-- Content Goes Here -->
</body>
</html>
```

Lists:

```HTML
<ol>...</ol> - Ordered List (Numbered List)
<ul>...</ul> - Unordered List (Bulletpoints)
<li>...</li> - List (Child of ul or ol)
```

Nested List:

```HTML
<ul>
    <li>Bantam
        <ul>
            <li>Silkie</li>
            <li>Polish</li>
        </ul>
    </li>
</ul>
```

Image:

```HTML
<img src="image source"> - image
```

```
<img src="image source" width="" height=""> - image with height and width (but better with CSS)
<img src="image source" alt="image name"> - adding alt to the image (for the screenreader)
```

Section 4: HTML: Next Steps & Semantics

```HTML
<div>...</div> - container for elements (is a block element)
<span>...</span> - wrap an element, so it can be changed in CSS (mostly for a piece of text)
<hr> - divider
<br> - line break
<sup>...</sup> - create a superscript (f.e. 1 in 1/2)
<sub>...</sub> - create a subsrcipt (f.e. 2 in 1/2)
```

HTML Entities:

```HTML
&lt - <
&gt - >
&copy - ©
...
```

Semantik:

```HTML
<main>...</main> - main content of the page
<nav>...</nav> - navigation links
<section>...</section> - section of the page / of the content
<article>...</article> - article (can be used on its own & independent)
<aside>...</aside> - side content (f.e. ads)
<header>...</header> - header of the page
<footer>...</footer> - footer of the page
<time>...</time> - time (f.e. date)
<figure>...</figure> - figure (f.e. image with caption)
```

Emmet:

```HTML
  > - child: main>section>h1 =>
      <main>
          <section>
              <h1>...</h1>
          </section>
      </main>
  + - sibling: h1+h2+h2 =>
      <h1>...</h1>
      <h2>...</h2>
      <h2>...</h2>
() - grouping
* - multiplication: h1*3 =>
      <h1>...</h1>
      <h1>...</h1>
```

```
    <h1>...</h1>
$ - item numbering: nav>ul>li.item$*3 =>
    <nav>
        <ul>
            <li class="item1">...</li>
            <li class="item2">...</li>
            <li class="item3">...</li>
        </ul>
    </nav>
{} - text: a{Click me!} =>
    <a href="...">Click me!</a>
```

Section 5: HTML: Forms & Tables

HTML Tables:

```HTML
<table>...</table> - table
<td>...</td> - table data (cell)
<th>...</th> - table header (bold)
<tr>...</tr> - table row
<thead>...</thead> - table header (row)
<tbody>...</tbody> - table body (row)
<tfoot>...</tfoot> - table footer (row)
<th rowspan="2">...</th> - table header (row span)
<th colspan="2">...</th> - table header (column span)
```

HTML: Forms

```HTML
<form action="...">...</form> - form with an action / link ("/" + to where the form goes; f.e. /home)
<input type="..."> - input
<input type="..." placeholder="..."> - placeholder

Text input form with placeholder "name":
<form action="...">
    <input type="text" placeholder="name">
</form>

<label for="...">...</label> - label for input

Using label (for = unique id of the input):
<label for="name">Name</label>
<input type="text" id="name" placeholder="name">

<input type="text" name="name"> - input with name (for the server; f.e. after submitting the form, name=John or https://www.google.com/search?q=John)

<button type="submit">...</button> - button (goes to the action of the form)
<button type="reset">...</button> - reset button (resets the form)
<button type="button">...</button> - button (does not go to the action of the form)

<input type="checkbox" name="..."> - checkbox with name
<input type="radio" name="..."> - radio button with name
<input type="..." name="..." value="..."> - value of the input (f.e. for checkbox or radio button)
<select>...</select> - select box
<option>...</option> - option in the select box

```

```
Drop down:
<select name="..." id="...">
    <option value="1">...</option>
    <option value="2">...</option>
</select>

<input type="range" id="..." min="..." max="..." step="..." value="..." name="..."> - range input (slider); value is the default(start) value
<input type="color" id="..." name="..."> - color input
...

<textarea>...</textarea> - text area (multiline input)
<textarea id="..." rows="..." cols="...">...</textarea> - text area with rows and columns
```

Built in validations:

```
<input type="text" minlength="..."> - minimum length of the input
<input type="text" maxlength="..."> - maximum length of the input
<input type="text" required> - required input (must be filled out | must be not empty)
```

Section 6: CSS: The Very Basics

Styles:

```HTML
Style element:
<h1 style="color: red;">...</h1> - inline style (not recommended)

External style (using CSS):
<head>
    <link rel="stylesheet" href="style.css"> - link to the CSS file (in the head of the HTML file; works automatically)
</head>
<body>
    <h1>...</h1>
</body>
```

External style: link to CSS (href:“style.css”)

↓

```CSS
style.css:
h1 {
    color: red;
}
```

Color changing in CSS:

```CSS
... {color: red;}
/* CSS color*/
... {background-color: red;}
/* CSS background color*/
... {background: red;}
/* CSS background (color, image, position, repeat, size)*/
... {color: rgb(255, 0, 0);}
/* CSS color (RGB)*/
... {color: #FF0000;}
/* CSS color (hexadecimal)*/
```

Text properties:

```CSS
... {text-align: center;}
/* CSS text alignment (can be left, right, center, justify)*/
... {width: 100px;}
/* CSS width*/
... {height: 100px;}
/* CSS height*/
... {font-weight: bold;}
/* CSS font weight (can be normal, bold, bolder, lighter)*/
```

```
... {font-size: 20px;}
/* CSS font size*/
... {text-decoration: underline;}
/* CSS text decoration (can be none, underline, overline, linethrough)*/
... {letter-spacing: 2px;}
/* CSS letter spacing*/
... {font-family: Arial, sans-serif;}
/* CSS font family (can be serif, sans serif, monospace, cursive, fantasy) */
```

Section 7: The World of CSS Selectors

```css
* {color: red;}
/*CSS universal selector (selects all elements)*/
... , ... {color: red;}
/*CSS group selector (selects all elements with the same style)*/
#id {color: red;}
/*CSS id selector (selects the element with the id)*/
.class {color: red;}
/*CSS class selector (selects all elements with the class; fe. <span class="class"> ... </span>)*/
... ... {color: red;}
/*CSS descendant selector (selects all elements inside the element, fe. all paragraphs p inside the div)*/
... + ... {color: red;}
/*CSS adjacent sibling selector (selects the element that is next to the element; fe. h1 and p are siblings/are next to each other => h1 + p)*/
... > ... {color: red;}
/*CSS child selector (selects the element that is a child of the element; fe. h1 is child of the div => div > h1 => h1 will be changed)*/
... [ ... = "..."] {color: red;}
/* CSS attribute selector (selects the element with the attribute; fe. input with type text => input[type="text"] or section[class="class"] => section with class "class") */
... [ ... *= "..."] {color: red;}
/* CSS attribute selector (selects the element with the attribute that contains the value; fe. input with type text => input[type*="text"] or section[class*="class"] => section with class "class") */
```

Pseudo classes:

```css
... :hover {color: red;}
/* CSS hover selector (selects the element when the mouse is over it) */
... :checked {color: red;}
/* CSS checked selector (selects the element when it is checked; f.e. checkbox or radio button) */
... :nth-of-type(2n) {color: red;}
/* CSS nth type selector (selects the element that is the nth child of the element; f.e. every second child => 2n) */
```

Pseudo elements:

```css
... ::after {color: red;}
/* CSS after selector (selects the element after the element) */
... ::before {color: red;}
/* CSS before selector (selects the element before the element) */
... ::first-letter {color: red;}
```

```
/* CSS first letter selector (selects the first letter of the element) */
... ::first-line {color: red;}
/* CSS first line selector (selects the first line of the element) */
... ::selection {color: red;}
/* CSS selection selector (selects the selected text of the element) */
```

The CSS Cascade:

```
h1 {color:red}
h1 {color:blue}

=> h1 is blue
```

CSS Specificity:

```
ID > Class > Element

section p {color:teal;} /* => 0 0 2 */
#submit {color: olive;} /* => 1 0 0 */
/* 1 0 0 is far more specific than 0 0 2 */
nav a.active {color: orange;} /* => 0 1 2 */

Also: Inline styles > ID

... {color: red !important;} /* !important is the most important
=> !important > Inline styles > ID > Class > Element */
```

CSS Inheritance:

```
/* certain things don't inherit at default: buttons, input, etc. ... */
button {color: inherit;} /* inherit color from the parent element */
```

Section 8: The CSS Box Model

```css
... {width: 100px;} /* width of the element */
... {height: 100px;} /* height of the element */
```

Border:

```css
... {border: 1px solid red;} /* border (width, style, color) */
... {border_width: 1px;} /* border width */
... {border_style: solid;} /* border style (solid, dotted, dashed, double, groove, ridge, inset, outset) */
... {border_color: red;} /* border color */
... {box-sizing: border-box;} /* box sizing (content-box, border-box) */
... {border-radius: 10px;} /* border radius (round corners) */
```

Padding:

```css
... {padding: 10px;} /* padding (top, right, bottom, left) */
... {padding: 10px 20px;} /* padding (top/bottom, right/left) */
```

Margins:

```css
... {margin: 10px;} /* margin (top, right, bottom, left) */
... {margin: 10px 20px;} /* margin (top/bottom, right/left) */
```

Display properties:

```css
... {display: block;} /* display block (takes the whole width) */
... {display: inline;} /* display inline (takes the width of the content) */
... {display: inline-block;} /* display inline block (takes the width of the content, but can have width and height) */
... {display: flex;} /* display flex (flexbox) */
... {display: grid;} /* display grid (grid) */
```

CSS Units:

```css
/* Font size:
1em = the font-size of the parent
2em = 2 times the font-size of the parent

1rem = the font-size of the root element (html)
2rem = 2 times the font-size of the root element (html)

Other properties:
1em = computed from font-size of the element itself */
```

Section 9: Other Assorted Useful CSS Properties

Alpha:

```css
... {color: rgb(0, 0, 0, alpha)} /* color (RGB with alpha) */
... {opacity: 0.5;} /* opacity (0 = transparent, 1 = opaque) */
```

Positions:

```css
... {position: static;} /* static (default) */
... {position: relative;} /* relative (relative to the element itself) */
... {position: absolute;} /* absolute (relative to the parent element) */
... {position: fixed;} /* fixed (relative to the viewport(containbox)) */
... {position: sticky;} /* sticky (relative to the viewport, but only when scrolling) */
```

Transitions:

```css
... {transition: 1s} /* transition (time) */
... {transition: property name duration timingFunction delay} /* transition (property name, duration, timing function, delay) */
... {transition-timing-function: ease;} /* transition timing function (ease, linear, ease-in, ease-out, ease-in-out) */
```

Transforming:

```css
... {transform: rotate(45deg);} /* transform (rotate) */
... {transform-origin: top left;} /* transform origin (where the element is transformed from) */
... {transform: scale(1.5);} /* transform (scale) */
... {transform: translate(10px, 20px);} /* transform (translate X and Y) */
... {transform: skew(10deg);} /* transform (skew) */
```

Background:

```css
... {background-image: url(image.png);} /* background image */
... {background-size: cover;} /* background size (cover, contain) */
... {background-position: center;} /* background position (top, bottom, left, right, center) */
... {background-repeat: no-repeat;} /* background repeat (no-repeat, repeat, repeat-x, repeat-y) */
or
... {background: url(image.png) no-repeat center / cover;} /* background (image, repeat, position, size) */
```

Google Fonts:

```css
font-family: 'Roboto', sans-serif; /* font family (Google Fonts) */
```

Import Google font using link in HTML

↓

```HTML
<link href="https://fonts.googleapis.com/css2?family=Roboto:wght@400&
display=swap" rel="stylesheet"> /* link to Google Fonts */
```

Section 10: Responsive CSS & flexbox

Flexbox

```CSS
... {display: flex} /* display flex (flexbox) */
... {flex-direction: row} /* flex direction (row, column, row-reverse) */
... {justify-content: center} /* justify content (center, flex-start, flex-end, space-between, space-around, space-evenly)*/
... {flex-wrap: wrap} /* flex wrap (wrap, nowrap, wrap-reverse) */
... {align-items: center} /* align items (center, flex-start, flex-end, stretch) */
... {align-content: center} /* align content (center, flex-start, flex-end, stretch) */
... {align-self: center} /* align self (center, flex-start, flex-end, stretch); is just for one element */
... {flex-basis: 100px} /* flex basis (width of the element); depending on the main axis: height or width */
... {flex-grow: 1} /* flex grow (how much the element grows); 1 - takes all available space, 2 - twice as much space as 1, ... */
... {flex-shrink: 1} /* flex shrink (how much the element shrinks) */
... {flex: 1} /* flex (flex-grow, flex-shrink, flex-basis) */
... {flex: 1 0 100px} /* flex (flex-grow, flex-shrink, flex-basis) */
```

Responsive CSS

```CSS
@media (min-width: 600px) { /* media query (min-width, max-width, width) */
    ... {color: red;} /* CSS code for the media query */
}
@media (min-width: 600px) and max-width: 800) {
    ... {color: red;}
}
@media (orientation: landscape) { /* media query (orientation) */
    ... {color: red;}
}
```

Section 11: Pricing Panel Project

```CSS
...: last-child {color: red;} /* CSS last child selector (selects the last child of the element) */
...: first-child {color: red;} /* CSS first child selector (selects the first child of the element) */
```

Section 12: CSS Frameworks: Bootstrap

https://getbootstrap.com/docs/5.3/getting-started/introduction/

```HTML
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/css/bootstrap.min.css" integrity="sha384-SgOJa3DmI69IUzQ2PVdRZhwQ+dy64/BUtbMJw1MZ8t5HZApcHrRKUc4W0kG879m7" crossorigin="anonymous"> - link to Bootstrap
```

```JavaScript
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/bootstrap.bundle.min.js" integrity="sha384-k6d4wzSIapyDyv1kpU366/PK5hCdSbCRGRCMv+eplOQJWyd1fbcAu9OCUj5zNLiq" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/bootstrap.min.js" integrity="sha384-VQqxDN0EQCkWoxt/0vsQvZswzTHUVOImccYmSyhJTp7kGtPed0Qcx8rK9h9YEgx+" crossorigin="anonymous"></script>
```

```HTML
<div class="container">...</div> - container (Bootstrap)
<div class="container-fluid">...</div> - container fluid (Bootstrap)
<button type="button" class="btn btn-outline-info btn-lg"></button> - button (Bootstrap)
<h1 class="display-1">...</h1> - heading (Bootstrap)
<blockquote class="blockquote text-right">...</blockquote> - blockquote (Bootstrap)
<button class="btn btn-warning">...</button> - button (Bootstrap) - button.btn.btn-warning
```

Close button (Bootstrap):

```HTML
<button aria-label="Close" class="close" data-dismiss="alert">
    <span aria-hidden="true">&times;</span>
</button>
```

Grid system:

```HTML
 <div class="container">
     <div class="row">
         <div class="col-6">...</div> - .col-6 (max is 12 = full grid)
         <div class="col-6">...</div> - .col-6 (max is 12 = full grid)
     </div>
 or
```

```
7       <div class="row">
8           <div class="col">...</div> - .col (even space)
9           <div class="col">...</div> - .col (even space)
10      </div>
11  </div>
```

Responsive grid system:

```HTML
1  <div class="col-md-6">...</div> - .col-md-6 (stacks below medium = md)
2  <div class="col-xl-6">...</div> - .col-xl-6 (stacks below extra large = xl)
```

Align items:

```HTML
1  <div class="row align-items-start">...</div> - (align items start)
2  <div class="row justify-comtent-center">...</div> - (justify content center)
```

Forms:

```HTML
1   <form action="...">
2       <div class="form-group">
3           <label for="email">Email</label>
4           <input type="email" class="form-control" id="name" placeholder="Email">
5       </div>
6
7       <div class="form-group">
8           <label for="password">Password</label>
9           <select name="state" id="state" class="form-control">
10              <option value="1">...</option>
11              <option value="2">...</option>
12          </select>
13      </div>
14  </form>
```

Custom checkbox:

```HTML
1  <div class="class-control custom-checkbox">
2      <input type="checkbox" class="custom-control-input" id="customCheck1">
3      <label class="custom-control-label" for="customCheck1">Custom checkbox</
       label>
4  </div>
```

Navbar:

```HTML
1   <nav class="navbar navbar-expand-lg navbar-light bg-light">
2       <a class="navbar-brand" href="#">Navbar</a>
```

```
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
        <div class="navbar-nav">
            <a class="nav-item nav-link active" href="#">Home</a>
            <a class="nav-item nav-link" href="#">Features</a>
            <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-disabled="true">Disabled</a>
        </div>
    </div>
</nav>
```

Icons:

```
<button class="...">...<svg ...>...</svg></button> - button with icon (Bootstrap)
```

HTML

Other Utilities:

```
<span class="border">...</span> - border
<div class="shadow">...</div> - shadow
```

HTML

Padding:

```
<button class="btn p-0">...</button> - padding (from 0 - 5)
```

HTML

Section 14: JavaScript Basics!

`Browser -> Inspect -> Console = JavaScript`

```JavaScript
typeof <variable> // type of the variable
NaN // not a number
let <variable> = <value>; // variable declaration
const <variable> = <value>; // constant declaration
var <variable> = <value>; // old variable declaration (not recommended)
```

Section 15: JavaScript Strings and More

```JavaScript
<string>.trim() // remove whitespace from the beginning and end of the string
<string>.slice(<index>) // slice the string from beginning to index
<string>.slice(<startIndex>, <endIndex>) // slice the string from index to index
<string>.replace(<oldString>, <newString>) // replace the old string with the new string
"<string> ${variable}" // string interpolation
let <variable> = null // null
let <variable> = undefined // undefined
```

Section 16: JavaScript Decision Making

```JavaScript
console.log("...") // print to console
console.error("...") // print error to console
alert("...") // alert
prompt("...") // prompt, like alert, but with input f.e. let var = prompt("...")
```

Using JavaScript

```HTML
<body>
    <!-- Code -->
    <script src="script.js"></script> <!-- link to JavaScript file -->
</body>
```

Truthy and falsy values:

```JavaScript
// falsy values: 0, "", null, undefined, NaN
// everything else is truthy
```

Section 17: JavaScript Arrays

```JavaScript
let <array> = [1, 2, "a"] // array declaration; arrays are dynamic
<array>[n] = "b" // add element to array
<array>.push("b") // add element to array
<array>.pop() // remove last element from array
<array>.shift() // remove first element from array
<array>.unshift("b") // add element to the beginning of the array
<array>.concat([1, 2]) // concatenate arrays
<array>.includes("a") // check if array includes element
<array>.indexOf("a") // get index of element
<array>.reverse // reverse array
<array>.slice(0, 2) // slice array from index 0 to index 2
<array>.slice(1) // slice array from index 1 to end
<array>.splice(0, 2) // remove elements from index 0 to index 2
[1,2,3] === [1,2,3] // false (arrays are reference types)

num = [1,2,3]
numCopy = num
num === numsCopy // true (both are the same reference)

const <array> can be changed, but not reassigned
f.e.
const <array> = [1,2,3]
<array>[0] = 4 // ok
<array> = [4,5,6] // error
```

Section 18: JavaScript Object Literals

```JavaScript
const person = {
    firstName: 'Mick',
    lastName: 'Jagger',
    age: 18
}
person['firstName'] // get value of firstName
person.firstName // get value of firstName
person['first' + 'Name'] // get value of firstName
person.firstName = 'John' // set value of firstName

// Nested structure:
const students = {
    student1: {
        firstName: 'Mick',
        lastName: 'Jagger',
        age: 18
    },
    student2: {
        firstName: 'John',
        lastName: 'Doe',
        age: 20
    }
}
```

Section 19: Repeating Stuff With Loops

```javascript
for (let i = 0; animals.length; i++) {
    console.log(animals[i])
}

let <variable> = parseInt(prompt("...")) // parseInt (convert string to number)

// For... of loop:
for (let a of <array>) {
    console.log(a)
}

Object.keys(<object>) // get keys of object
Object.values(<object>) // get values of object
Object.entries(<object>) // get entries of object

for (let [key, value] of Object.entries(<object>)) {
    console.log(key, value)
}

for (let value of Object.values(<object>)) {
    console.log(value)
}
```

Section 20: Introduction Functions

```JavaScript
function <name>(<parameters>) {
    // code
}
```

Section 21: Leveling Up Functions

```JavaScript
// Function Expressions
const <name> = function(<parameters>) {
    // code
}

// Higher Order Functions = function as parameter
const <name> = function(<parameters>, <callback>) {
    // code
    <callback>(<parameters>)
}

// Returning Functions
const <name> = function(<parameters>) {
    return function(<parameters>) {
        // code
    }
}

// Methods
const <object> = {
    <name>: function(<parameters>) {
        // code
    }
}

// try/catch
try {
    // code
} catch (error) {
    console.log(error)
}
```

Section 22: Callbacks & Array Methods

JavaScript

```
// For-each loop
const <array> = [1, 2, 3, 4, 5]
<array>.forEach(function(<element>) {
    console.log(<element>)
})

// Map
<array>.map(function(<element>) {
    return <element> * 2
})

// Arrow function
const <name> = (<parameters>) => {
    // code (does not need return if only one line)
    // f.e. const add = (a, b) => a + b
}

// Map + Arrow function
const <name> = <array>.map(<element> => <element> * 2)

// setTimeout & setInterval
setTimeout(function() {
    // code
}, 1000) // setTimeout (after 1 second)

setInterval(function() {
    // code
}, 1000) // setInterval (every 1 second)

const id = setInterval(function() {
    // code
}, 1000) // setInterval (every 1 second)
clearInterval(id) // clear interval

// Filter method (create a new array with elements that pass the test)
<array>.filter(function(<element>) {
    return <element> > 2
})

// Filter + Map
<array>.filter(function(<element>) {
    return <element> > 2
}).map(function(<element>) {
    return <element> * 2
```

```
})

// Some (similar to every, but returns true if at least one element is true/ passes the test)
<array>.some(function(<element>) {
    return <element> > 2
})

// Every (tests whether all elements in the array pass the test implemented by the provided function, returns a Boolean value)
<array>.every(function(<element>) {
    return <element> > 2
})

// Reduce method
<array>.reduce(function(<accumulator>, <element>) {
    return <accumulator> + <element>
}, 0) // reduce (0 is the initial value of the accumulator)

// Arrow functions & this
const <name> = function() {
    <property> = <value>
    <method> = function() {
        console.log(this.<property>)
    }
}
```

Section 23: Newer JavaScript Features

```JavaScript
// Deafault params (when param is undefined)
function <name>(<param1> = 1, <param2> = 2) {
    // code
}
```

Spread

```JavaScript
// Spread in function calls
const <array> = [1, 2, 3]
Math.max(...<array>) // Math.max(1, 2, 3)
Math.max(<array>) // NaN
console.log(...'Hello') // H e l l o

// Spread in array literals
const <array1> = [1, 2, 3]
const <array2> = [4, 5, 6]
const <array3> = [...<array1>, ...<array2>] // [1, 2, 3, 4, 5, 6]
const <array4> = [1, 2, 3, ...<array1>] // [1, 2, 3, 1, 2, 3]

// Spread in objects
const <object1> = {a: 1, b: 2}
const <object2> = {c: 3, d: 4}
const <object3> = {...<object1>, ...<object2>} // {a: 1, b: 2, c: 3, d: 4}

// Rest params (when param is an array)
function <name>(...<params>) {
    // code
}
funcion <name>(a, b, ...<params>) {
    console.log(a);
    console.log(b);
    console.log(<params>);
}
<name>(1, 2, 3, 4, 5) // 1, 2, [3, 4, 5]
```

Destructuring

```JavaScript
// Destructuring arrays
const <array> = [1, 2, 3]
const [<a>, <b>, <c>] = <array> // a = 1, b = 2, c = 3

// Destructuring objects
const <object> = {a: 1, b: 2}
```

```
const {<a>, <b>} = <object> // a = 1, b = 2
const {<a>: <newName>} = <object> // a = 1, newName = 1

// Destructuring params
function <name>({<a>, <b>}) {
    console.log(<a>, <b>)
}
const <object> = {a: 1, b: 2}
```

Section 24: DOM

Document

```JavaScript
document // document (HTML document)
document.createElement("tag") // create element
console.dir(document) // show document in console (JavaScript object)
document.getElementById("id") // get element by id
document.getElementByTagName("tag") // get element by tag name
document.getElementsByClassName("class") // get element by class name => HTMLCollection
```

QuerySelector

```JavaScript
document.querySelector("selector") // get just one element by selector (CSS selector) (classes, id, tags, :nth-child, attributes, etc.)
document.querySelectorAll("selector") // get element/elements by selector (CSS selector) => NodeList
document.querySelector("selector").innerText // get inner text of the element
document.querySelector("selector").textContent // get text content of the element
document.querySelector("selector").innerHTML // get inner HTML of the element (f.e. text with tags)
document.querySelector("#selector").src // get src of the element (f.e. image)
document.querySelector("selector").setAttribute("attribute", "value") // set attribute of the element
<element>.style.<property> = "value" // set style of the element = inline style in HTML (f.e. <h1 style="color: red">)
window.getComputedStyle(<element>).<property> // get computed style of the element
```

ClassList

```JavaScript
<element>.classList // get class list of the element
<element>.classList.add("class") // add class to the element
<element>.classList.remove("class") // remove class from the element
<element>.classList.contains("class") // check if element has class -> Boolean
<element>.classList.toggle("class") // toggle class (add if not present, remove if present)
<element>.getAttribute("attribute") // get attribute of the element
```

Parent, Child, Sibling

```JavaScript
  <element>.parentElement // get parent element
  <element>.children // get children of the element (not an array => HTMLCollection)
  <element>.nextElementSibling // get next sibling of the element => Node
  <element>.previousElementSibling // get previous sibling of the element => Node
```

```
document.body.appendChild(<element>) // append child to the end of the body
document.body.removeChild(<element>) // remove child from the body
<element>.append(<element>) // append child to the element as the last child
<element>.prepend(<element>) // append child to the element as the first child
<element>.insertAdjacentElement(<where>, <element>) // insert element before the element (beforebegin, afterbegin, beforeend, afterend)
<element>.remove() // remove element
```

Section 25: DOM Events

```HTML
NOT RECOMMENDED
<button onclick="function()">...</button> - onclick (HTML)
```

```JavaScript
const btn = document.querySelector("#<id>") // get button
btn.onclick = function() { // onclick
    // code
}
document.querySelector("#<id>").onclick = () => { // onclick
    // code
}
<button>.addEventListener("click", function() { // add event listener (recommended: because of override in inline style)
    // code
})
<button>.addEventListener("<event>", <function>, {once: true}) // add event listener (once: true - remove after first click)
```

This in addEventListener

```JavaScript
for (let a in <array>) {
    a.addEventListener("click", <nameOfFunction>)
}
function <nameOfFunction> {
    this.style.color = "red" // this = element that was clicked => a in <array>
}
```

Keys

```JavaScript
document.querySelector("<element>").addEventListener("keydown", function(event) { // keydown event
    console.log(event.key) // get key that was pressed (or event.code)
})
// or
window.addEventListener("keydown", function(event) { // keydown event
    console.log(event.key) // get key that was pressed (or event.code)
})
```

```JavaScript
// Form Events and PreventDefaults
document.querySelector("#id").addEventListener("submit", function(event) { // submit event
    event.preventDefault() // prevent default action (f.e. reload page)
```

```
    console.log(event.target.elements[0].value) // get value of the first input in the form
})

// Input and change events
<element>.addEventListener("input", function(event) { // input event
    console.log(event.target.value) // get value of the input
})

// Event Bubbling
<element>.addEventListener("click", function(event) {
    // code
    event.stopPropagation() // stop event bubbling (stop event from going to parent elements)
})

// Event Delegation
<element>.addEventListener("click", function(event) {
    event.target // get element that was clicked
    // f.e. event.target.nodeName == 'LI' && e.target.remove();
})
```

Section 27: Async JavaScript

JavaScript is single threaded

Callbacks

```JavaScript
<first_function>(<parameter>, () => {
    <second_function>(<parameter>, () => {
        // if it works, run this:
    }, () => {
        // if it doesn't work, run this:
    })
})
```

Enter promises

```JavaScript
const <promise> = <function> (<parameter>);
<promise>
    .then(() => {
    // if it works, run this:
    })
    .catch(() => {
    // if it doesn't work, run this:
    })

const <promise> = new Promise((resolve, reject) => {
    // code
    if (<condition>) {
        resolve(<value>)
    } else {
        reject(<error>)
    }
})
```

Async functions

```JavaScript
// async functions always return a promise
async function <name> (<parameter>) {
    // code
}

// f.e.
const login = async (username, password) => {
    if (!username || !password) {
        throw new Error("Username and password are required")
```

```
    }
    if (password === "12345") return "Welcome!"
    throw new Error("Wrong password")
}

login ("Mick", "12345")
    .then((msg) => {
        console.log("Logged in")
        console.log(msg)
    })
    .catch((error) => {
        console.log(error)
    })

// Logged in
// Welcome!
```

Await keyboard

```
async function <name> (<parameter>) {
    await <promise> // wait for the promise to resolve (insted of .then)
    // code
}
```

JavaScript

Error handling in async functions

```
async function <name> (<parameter>) {
    try {
        await <promise> // wait for the promise to resolve (insted of .then)
        // code
    } catch (error) {
        console.log(error)
    }
}
```

JavaScript

Section 28: AJAX and API's

https://hoppscotch.io/ - API testing tool

JSON:

```javascript
JSON.parse(<data>) // parse JSON data
JSON.stringify(<data>) // stringify data to JSON
```

Making XHRs:

```javascript
const req = new XMLHttpRequest(); // create new XMLHttpRequest

req.onload = fucntion () {
    console.log("Loaded");
    const data = JSON.parse(this.responseText); // parse JSON data
    consoloe.log(data.<property>); // get property from data
};

req.onerror = function () {
    console.log("Error");
    conslo.log(this);
};

req.open("GET", "https://api.example.com/data"); // open request (GET, POST, PUT, DELETE)
req.send(); // send request
```

Using Fetch API:

```javascript
fetch("https://api.example.com/data");
    .then(res => {
        console.log(res);
        return res.json();
    })
    .then(data => {
        console.log(data);
        return fetch("https://api.example.com/data2");
    })
    .then(res => {
        console.log(res);
        return res.json();
    })
    .catch((e) => {
        console.log(e);
```

```
    });

// or using async function

const getData = async () => {
    try {
        const res = await fetch("https://api.example.com/data");
        const data = await res.json();
        console.log(data);

        const res2 = await fetch("https://api.example.com/data2");
        const data2 = await res2.json();
        console.log(data2);
    } catch (e) {
        console.log(e);
    }
}
getData();
```

Axios:

```
<script src="https://cdn.jsdelivr.net/npm/axios/dist/axios.min.js"></
script>
```
HTML