

## 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

## Links:

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

## Section 3: HTML: The Essentials

---

Essentials:

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

HTML

HTML Sceleton:

```
1 ! - create a HTML Sceleton
2
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <title> My page </title>
7 </head>
8 <body>
9   <!-- Content Goes Here -->
10 </body>
11 </html>
```

HTML

Lists:

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

HTML

Nested List:

```
1 <ul>
2   <li>Bantam
3     <ul>
4       <li>Silkie</li>
5       <li>Polish</li>
6     </ul>
7   </li>
8 </ul>
```

HTML

Image:

```
1  - image
```

HTML

```
2  - image with height and width (but  
better with CSS)  
3  - adding alt to the image (for the  
screenreader)
```

## Section 4: HTML: Next Steps & Semantics

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

HTML

HTML Entities:

- 1 `&lt;` - <
- 2 `&gt;` - >
- 3 `&copy;` - ©
- 4 ...

HTML

Semantik:

- 1 `<main>...</main>` - main content of the page
- 2 `<nav>...</nav>` - navigation links
- 3 `<section>...</section>` - section of the page / of the content
- 4 `<article>...</article>` - article (can be used on its own & independent)
- 5 `<aside>...</aside>` - side content (f.e. ads)
- 6 `<header>...</header>` - header of the page
- 7 `<footer>...</footer>` - footer of the page
- 8 `<time>...</time>` - time (f.e. date)
- 9 `<figure>...</figure>` - figure (f.e. image with caption)

HTML

Emmet:

- 1 `>` - child: `main>section>h1 =>`
- 2 `<main>`
- 3 `<section>`
- 4 `<h1>...</h1>`
- 5 `</section>`
- 6 `</main>`
- 7 `+` - sibling: `h1+h2+h2 =>`
- 8 `<h1>...</h1>`
- 9 `<h2>...</h2>`
- 10 `<h2>...</h2>`
- 11 `()` - grouping
- 12 `*` - multiplication: `h1*3 =>`
- 13 `<h1>...</h1>`
- 14 `<h1>...</h1>`

HTML

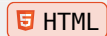
```
15     <h1>...</h1>
16 $ - item numbering: nav>ul>li.item$*3 =>
17     <nav>
18         <ul>
19             <li class="item1">...</li>
20             <li class="item2">...</li>
21             <li class="item3">...</li>
22         </ul>
23     </nav>
24 {} - text: a{Click me!} =>
25     <a href="...">Click me!</a>
```

## Section 5: HTML: Forms & Tables

---

### HTML Tables:

```
1 <table>...</table> - table
2 <td>...</td> - table data (cell)
3 <th>...</th> - table header (bold)
4 <tr>...</tr> - table row
5 <thead>...</thead> - table header (row)
6 <tbody>...</tbody> - table body (row)
7 <tfoot>...</tfoot> - table footer (row)
8 <th rowspan="2">...</th> - table header (row span)
9 <th colspan="2">...</th> - table header (column span)
```



### HTML: Forms

```
1 <form action="...">...</form> - form with an action / link ("/" + to
  where the form goes; f.e. /home)
2 <input type="..."> - input
3 <input type="..." placeholder="..."> - placeholder
4
5 Text input form with placeholder "name":
6 <form action="...">
7   <input type="text" placeholder="name">
8 </form>
9
10 <label for="...">...</label> - label for input
11
12 Using label (for = unique id of the input):
13 <label for="name">Name</label>
14 <input type="text" id="name" placeholder="name">
15
16 <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)
17
18 <button type="submit">...</button> - button (goes to the action of the form)
19 <button type="reset">...</button> - reset button (resets the form)
20 <button type="button">...</button> - button (does not go to the action of the
  form)
21
22 <input type="checkbox" name="..."> - checkbox with name
23 <input type="radio" name="..."> - radio button with name
24 <input type="..." name="..." value="..."> - value of the input (f.e. for
  checkbox or radio button)
25 <select>...</select> - select box
26 <option>...</option> - option in the select box
27
```



```
28 Drop down:
29 <select name="..." id="...">
30     <option value="1">...</option>
31     <option value="2">...</option>
32 </select>
33
34 <input type="range" id="..." min="..." max="..." step="..." value="..."
    name="..."> - range input (slider); value is the default(start) value
35 <input type="color" id="..." name="..."> - color input
36 ...
37
38 <textarea>...</textarea> - text area (multiline input)
39 <textarea id="..." rows="..." cols="...">...</textarea> - text area with rows
    and columns
```

Built in validations:

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



## Section 6: CSS: The Very Basics

---

Styles:

```
1  Style element: HTML
2  <h1 style="color: red;">...</h1> - inline style (not recommended)
3
4  External style (using CSS):
5  <head>
6      <link rel="stylesheet" href="style.css"> - link to the CSS file (in the head
        of the HTML file; works automatically)
7  </head>
8  <body>
9      <h1>...</h1>
10 </body>
```

External style: link to CSS (href="style.css")



```
1  style.css: CSS
2  h1 {
3      color: red;
4  }
```

Color changing in CSS:

```
1  ... {color: red;} CSS
2  /* CSS color*/
3  ... {background-color: red;}
4  /* CSS background color*/
5  ... {background: red;}
6  /* CSS background (color, image, position, repeat, size)*/
7  ... {color: rgb(255, 0, 0);}
8  /* CSS color (RGB)*/
9  ... {color: #FF0000;}
10 /* CSS color (hexadecimal)*/
```

Text properties:

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



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

## Section 7: The World of CSS Selectors

```
1  * {color: red;}
2  /*CSS universal selector (selects all elements)*/
3  ... , ... {color: red;}
4  /*CSS group selector (selects all elements with the same style)*/
5  #id {color: red;}
6  /*CSS id selector (selects the element with the id)*/
7  .class {color: red;}
8  /*CSS class selector (selects all elements with the class; fe. <span
   class="class"> ... </span>)/
9  ... ... {color: red;}
10 /*CSS descendant selector (selects all elements inside the element, fe. all
   paragraphs p inside the div)*/
11 ... + ... {color: red;}
12 /*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)*/
13 ... > ... {color: red;}
14 /*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)*/
15 ... [ ... = "..."] {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") */
17 ... [ ... *= "..."] {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:

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

Pseudo elements:

```
1  ... ::after {color: red;}
2  /* CSS after selector (selects the element after the element) */
3  ... ::before {color: red;}
4  /* CSS before selector (selects the element before the element) */
5  ... ::first-letter {color: red;}
```

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

The CSS Cascade:

```
1  h1 {color:red}
2  h1 {color:blue}
3
4  => h1 is blue
```

 CSS

CSS Specificity:

```
1  ID > Class > Element
2
3  section p {color:teal;} /* => 0 0 2 */
4  #submit {color: olive;} /* => 1 0 0 */
5  /* 1 0 0 is far more specific than 0 0 2 */
6  nav a.active {color: orange;} /* => 0 1 2 */
7
8  Also: Inline styles > ID
9
10 ... {color: red !important;} /* !important is the most important
11 => !important > Inline styles > ID > Class > Element */
```

 CSS

CSS Inheritance:

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

 CSS

## Section 8: The CSS Box Model

---

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

 CSS

Border:

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

 CSS

Padding:

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

 CSS

Margins:

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

 CSS

Display properties:

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

 CSS

CSS Units:

```
1 /* Font size:
2 1em = the font-size of the parent
3 2em = 2 times the font-size of the parent
4
5 1rem = the font-size of the root element (html)
6 2rem = 2 times the font-size of the root element (html)
7
8 Other properties:
9 1em = computed from font-size of the element itself */
```

 CSS

## Section 9: Other Assorted Useful CSS Properties

Alpha:

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

CSS

Positions:

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

CSS

Transitions:

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

CSS

Transforming:

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

CSS

Background:

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

CSS

Google Fonts:

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

CSS

Import Google font using link in HTML



```
1 <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

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

 CSS

### Responsive CSS

```
1  @media (min-width: 600px) { /* media query (min-width, max-width, width)
   */
2      ... {color: red;} /* CSS code for the media query */
3  }
4  @media (min-width: 600px) and max-width: 800) {
5      ... {color: red;}
6  }
7  @media (orientation: landscape) { /* media query (orientation) */
8      ... {color: red;}
9  }
```

 CSS

## Section 11: Pricing Panel Project

---

```
1 ...: last-child {color: red;} /* CSS last child selector (selects the last  
   child of the element) */  
2 ...: 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/>

```
1 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/css/bootstrap.min.css" integrity="sha384-Sg0Ja3DmI69IuZQ2PVdRZhwQ+dy64/BUTbMJw1MZ8t5HZApChRkUc4W0kG879m7" crossorigin="anonymous"> - link to Bootstrap
```

HTML

```
1 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/bootstrap.bundle.min.js" integrity="sha384-k6d4wzSIapyDyvlkpU366/PK5hCdSbCRGRCMv+epl0QJWyd1fbcAu90CUj5zNLiq" crossorigin="anonymous"></script>
```

JavaScript

```
2 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM80Dewa9r" crossorigin="anonymous"></script>
```

```
3 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.5/dist/js/bootstrap.min.js" integrity="sha384-VQqxDN0EQCkWoxt/0vsQvZswzTHUV0ImccYmSyhJTp7kGtPed0Qcx8rK9h9YEg+" crossorigin="anonymous"></script>
```

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

HTML

Close button (Bootstrap):

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

HTML

Grid system:

```
1 <div class="container">
2   <div class="row">
3     <div class="col-6">...</div> - .col-6 (max is 12 = full grid)
4     <div class="col-6">...</div> - .col-6 (max is 12 = full grid)
5   </div>
6 or
```

HTML

```

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:

```

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:

```

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

```

Forms:

```

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:

```

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:

```

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

```

```

3     <button class="navbar-toggler" type="button" data-toggle="collapse" data-
      target="#navbarNavAltMarkup" aria-controls="navbarNavAltMarkup">
4         <span class="navbar-toggler-icon"></span>
5     </button>
6     <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
7         <div class="navbar-nav">
8             <a class="nav-item nav-link active" href="#">Home</a>
9             <a class="nav-item nav-link" href="#">Features</a>
10            <a class="nav-item nav-link disabled" href="#" tabindex="-1" aria-
              disabled="true">Disabled</a>
11        </div>
12    </div>
13 </nav>

```

Icons:

```

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

```

 HTML

Other Utilities:

```

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

```

 HTML

Padding:

```

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

```

 HTML

## Section 14: JavaScript Basics!

---

Browser -> Inspect -> Console = JavaScript

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

JS JavaScript

## Section 15: JavaScript Strings and More

---

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

JS JavaScript

## Section 16: JavaScript Decision Making

---

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

JS JavaScript

Using JavaScript

```
1 <body>
2   <!-- Code -->
3   <script src="script.js"></script> <!-- link to JavaScript file -->
4 </body>
```

HTML

Truthy and falsy values:

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

JS JavaScript

## Section 17: JavaScript Arrays

---

```
1  let <array> = [1, 2, "a"] // array declaration; arrays are dynamic
2  <array>[n] = "b" // add element to array
3  <array>.push("b") // add element to array
4  <array>.pop() // remove last element from array
5  <array>.shift() // remove first element from array
6  <array>.unshift("b") // add element to the beginning of the array
7  <array>.concat([1, 2]) // concatenate arrays
8  <array>.includes("a") // check if array includes element
9  <array>.indexOf("a") // get index of element
10 <array>.reverse // reverse array
11 <array>.slice(0, 2) // slice array from index 0 to index 2
12 <array>.slice(1) // slice array from index 1 to end
13 <array>.splice(0, 2) // remove elements from index 0 to index 2
14 [1,2,3] === [1,2,3] // false (arrays are reference types)
15
16 num = [1,2,3]
17 numCopy = num
18 num === numCopy // true (both are the same reference)
19
20 const <array> can be changed, but not reassigned
21 f.e.
22 const <array> = [1,2,3]
23 <array>[0] = 4 // ok
24 <array> = [4,5,6] // error
```

JS JavaScript

## Section 18: JavaScript Object Literals

---

```
1  const person = {
2      firstName: 'Mick',
3      lastName: 'Jagger',
4      age: 18
5  }
6  person['firstName'] // get value of firstName
7  person.firstName // get value of firstName
8  person['first' + 'Name'] // get value of firstName
9  person.firstName = 'John' // set value of firstName
10
11 // Nested structure:
12 const students = {
13     student1: {
14         firstName: 'Mick',
15         lastName: 'Jagger',
16         age: 18
17     },
18     student2: {
19         firstName: 'John',
20         lastName: 'Doe',
21         age: 20
22     }
23 }
```

JS JavaScript

## Section 19: Repeating Stuff With Loops

---

```
1  for (let i = 0; animals.length; i++) {  
2      console.log(animals[i])  
3  }  
4  
5  let <variable> = parseInt(prompt("...")) // parseInt (convert string to number)  
6  
7  // For... of loop:  
8  for (let a of <array>) {  
9      console.log(a)  
10 }  
11  
12 Object.keys(<object>) // get keys of object  
13 Object.values(<object>) // get values of object  
14 Object.entries(<object>) // get entries of object  
15  
16 for (let [key, value] of Object.entries(<object>)) {  
17     console.log(key, value)  
18 }  
19  
20 for (let value of Object.values(<object>)) {  
21     console.log(value)  
22 }
```

JS JavaScript

## Section 20: Introduction Functions

---

```
1 function <name>(<parameters>) {  
2     // code  
3 }
```

JS JavaScript

## Section 21: Leveling Up Functions

---

```
1 // Function Expressions  
2 const <name> = function(<parameters>) {  
3     // code  
4 }  
5  
6 // Higher Order Functions = function as parameter  
7 const <name> = function(<parameters>, <callback>) {  
8     // code  
9     <callback>(<parameters>)  
10 }  
11  
12 // Returning Functions  
13 const <name> = function(<parameters>) {  
14     return function(<parameters>) {  
15         // code  
16     }  
17 }  
18  
19 // Methods  
20 const <object> = {  
21     <name>: function(<parameters>) {  
22         // code  
23     }  
24 }  
25  
26 // try/catch  
27 try {  
28     // code  
29 } catch (error) {  
30     console.log(error)  
31 }
```

JS JavaScript



## Section 22: Callbacks & Array Methods

---

```
1  // For-each loop
2  const <array> = [1, 2, 3, 4, 5]
3  <array>.forEach(function(<element>) {
4      console.log(<element>)
5  })
6
7  // Map
8  <array>.map(function(<element>) {
9      return <element> * 2
10 })
11
12 // Arrow function
13 const <name> = (<parameters>) => {
14     // code (does not need return if only one line)
15     // f.e. const add = (a, b) => a + b
16 }
17
18 // Map + Arrow function
19 const <name> = <array>.map(<element> => <element> * 2)
20
21 // setTimeout & setInterval
22 setTimeout(function() {
23     // code
24 }, 1000) // setTimeout (after 1 second)
25
26 setInterval(function() {
27     // code
28 }, 1000) // setInterval (every 1 second)
29
30 const id = setInterval(function() {
31     // code
32 }, 1000) // setInterval (every 1 second)
33 clearInterval(id) // clear interval
34
35 // Filter method (create a new array with elements that pass the test)
36 <array>.filter(function(<element>) {
37     return <element> > 2
38 })
39
40 // Filter + Map
41 <array>.filter(function(<element>) {
42     return <element> > 2
43 }).map(function(<element>) {
44     return <element> * 2
```

JS JavaScript

```
45 })
46
47 // Some (similar to every, but returns true if at least one element is true/
   passes the test)
48 <array>.some(function(<element>) {
49     return <element> > 2
50 })
51
52 // Every (tests whether all elements in the array pass the test implemented by
   the provided function, returns a Boolean value)
53 <array>.every(function(<element>) {
54     return <element> > 2
55 })
56
57 // Reduce method
58 <array>.reduce(function(<accumulator>, <element>) {
59     return <accumulator> + <element>
60 }, 0) // reduce (0 is the initial value of the accumulator)
61
62 // Arrow functions & this
63 const <name> = function() {
64     <property> = <value>
65     <method> = function() {
66         console.log(this.<property>)
67     }
68 }
```

## Section 23: Newer JavaScript Features

---

```
1 // Default params (when param is undefined)
2 function <name>(<param1> = 1, <param2> = 2) {
3     // code
4 }
```

JS JavaScript

### Spread

```
1 // Spread in function calls
2 const <array> = [1, 2, 3]
3 Math.max(...<array>) // Math.max(1, 2, 3)
4 Math.max(<array>) // NaN
5 console.log(...'Hello') // H e l l o
6
7 // Spread in array literals
8 const <array1> = [1, 2, 3]
9 const <array2> = [4, 5, 6]
10 const <array3> = [...<array1>, ...<array2>] // [1, 2, 3, 4, 5, 6]
11 const <array4> = [1, 2, 3, ...<array1>] // [1, 2, 3, 1, 2, 3]
12
13 // Spread in objects
14 const <object1> = {a: 1, b: 2}
15 const <object2> = {c: 3, d: 4}
16 const <object3> = {...<object1>, ...<object2>} // {a: 1, b: 2, c: 3, d: 4}
17
18 // Rest params (when param is an array)
19 function <name>(...<params>) {
20     // code
21 }
22 function <name>(a, b, ...<params>) {
23     console.log(a);
24     console.log(b);
25     console.log(<params>);
26 }
27 <name>(1, 2, 3, 4, 5) // 1, 2, [3, 4, 5]
```

JS JavaScript

### Destructuring

```
1 // Destructuring arrays
2 const <array> = [1, 2, 3]
3 const [<a>, <b>, <c>] = <array> // a = 1, b = 2, c = 3
4
5 // Destructuring objects
6 const <object> = {a: 1, b: 2}
```

JS JavaScript

```
7  const {<a>, <b>} = <object> // a = 1, b = 2
8  const {<a>: <newName>} = <object> // a = 1, newName = 1
9
10 // Destructuring params
11 function <name>({<a>, <b>}) {
12     console.log(<a>, <b>)
13 }
14 const <object> = {a: 1, b: 2}
```

## Section 24: DOM

---

### Document

```
1 document // document (HTML document)
2 document.createElement("tag") // create element
3 console.dir(document) // show document in console (JavaScript object)
4 document.getElementById("id") // get element by id
5 document.getElementsByTagName("tag") // get element by tag name
6 document.getElementsByClassName("class") // get element by class name =>
  HTMLCollection
```

JS JavaScript

### QuerySelector

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

JS JavaScript

### ClassList

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

JS JavaScript

### Parent, Child, Sibling

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

JS JavaScript

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

## Section 25: DOM Events

---

1 NOT RECOMMENDED

HTML

2 `<button onclick="function()">...</button>` - onclick (HTML)

```
1  const btn = document.querySelector("#<id>") // get button
2  btn.onclick = function() { // onclick
3      // code
4  }
5  document.querySelector("#<id>").onclick = () => { // onclick
6      // code
7  }
8  <button>.addEventListener("click", function() { // add event listener
9      (recommended: because of override in inline style)
10     // code
11 })
12 <button>.addEventListener("<event>", <function>, {once: true}) // add event
13 listener (once: true - remove after first click)
```

This in addEventListener

```
1  for (let a in <array>) {
2      a.addEventListener("click", <nameOfFunction>)
3  }
4  function <nameOfFunction> {
5      this.style.color = "red" // this = element that was clicked => a in <array>
6  }
```

Keys

```
1  document.querySelector("<element>").addEventListener("keydown",
2  function(event) { // keydown event
3      console.log(event.key) // get key that was pressed (or event.code)
4  })
5  // or
6  window.addEventListener("keydown", function(event) { // keydown event
7      console.log(event.key) // get key that was pressed (or event.code)
8  })
```

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

```
4     console.log(event.target.elements[0].value) // get value of the first input
      in the form
5   })
6
7   // Input and change events
8   <element>.addEventListener("input", function(event) { // input event
9       console.log(event.target.value) // get value of the input
10  })
11
12  // Event Bubbling
13  <element>.addEventListener("click", function(event) {
14      // code
15      event.stopPropagation() // stop event bubbling (stop event from going to
      parent elements)
16  })
17
18  // Event Delegation
19  <element>.addEventListener("click", function(event) {
20      event.target // get element that was clicked
21      // f.e. event.target.nodeName == 'LI' && e.target.remove();
22  })
```