# Repository of most useful commands and functions in JavaScript

## Setup

From your HTML file, you can include JavaScript code in the following ways:

- `<script> ... </script>` - include JavaScript code directly in the HTML file
- `<script src="script.js" defer></script>` - include an external JavaScript file in the HTML file with the `defer` attribute to load the script after the document has been parsed.
- (in the JavaScript file) `document.addEventListener('DOMContentLoaded', function() { ... })` - execute JavaScript code when the HTML document has been fully loaded and parsed

## HTML elements

### Accessing HTML elements

- `document.getElementById('id')` - returns the element with the specified id
- `document.getElementsByClassName('class')` - returns a collection of all elements with the specified class.
- `document.getElementsByTagName('tag')` - returns a collection of all elements with the specified tag
- `document.querySelector('selector')` - returns the first element that matches the specified selector
- `document.querySelectorAll('selector')` - returns a collection of all elements that match the specified selector
- `document.querySelector('.class')` - returns the first element with the specified class
- `document.querySelector('#id')` - returns the element with the specified id

### Adding HTML elements

- `document.createElement('div')` - creates a new HTML element
- `document.createElement('ul')` - creates a new unordered list
- `document.appendChild(element)` - adds a new child element to the end of the document
- `element.parentNode.deleteChild(element)` - removes an element from the document
- `element.parentNode.insertBefore(newElement, element)` - inserts a new element before an existing element

### Modifying HTML elements

- `document.body.appendChild(childElement)` - adds a child element to the end of the parent element
- `element.innerHTML = 'new content'` - changes the inner HTML of an element
- `element.textContent = 'new content'` - changes the text content of an element (ignores HTML tags)
- `element.setAttribute('attribute', 'value')` - sets the value of an attribute on the specified element

- `element.getAttribute('attribute')` - returns the value of the specified attribute on the specified element
- `element.className = 'class'` - sets the class attribute of an element
- `element.classList.add('class')` - adds a class to an element
- `element.style.backgroundColor = 'red'` - changes the background color of an element

### Event listeners and actions

- `element.onclick = () => { ... }` - assigns a function to be executed when the element is clicked
- `element.addEventListener('click', function() { ... })` - assigns a function to be executed when the element is clicked
- `element.removeEventListener('click', function) { ... }` - removes an event listener from the element
- `element.disabled = true` - disables an element
- `element.focus()` - sets focus on an element
- `function functionName(e) {e.target.style.color = 'red'}` - changes the color of the clicked element

List of events:

- `click` - when the element is clicked
- `mouseover` - when the mouse pointer is moved onto the element
- `mouseout` - when the mouse pointer is moved out of the element
- `change` - when the value of the element is changed
- `input` - when the value of the element is changed
- `keydown` - when a key is pressed down
- `focus` - when the element gets focus
- `blur` - when the element loses focus
- `dblclick` - when the element is double-clicked

## JavaScript functions

### Nested functions

- `function outerFunction() { function innerFunction() { ... } }` - a function can contain another more specific function
- `stopPropagation()` - stops the bubbling of an event to parent elements

### Popups and prompts

- `const content = prompt('message')` - displays a dialog box with a message and an input field for the user to enter text
- `alert('message')` - displays an alert box with a message

### Storage

- `localStorage.setItem('key', 'value')` - stores data in the web browser's local storage
- `localStorage.getItem('key')` - retrieves data from the web browser's local storage
- `localStorage.removeItem('key')` - removes data from the web browser's local storage

- `localStorage.clear()` - removes all data from the web browser's local storage

We can use the local storage to check conditions (see the Prompt and store user input example in the gallery).

### Date and time

- `new Date()` - creates a new date object with the current date and time
- `dateObject.getHours()` - returns the hour of the specified date
- `dateObject.getMinutes()` - returns the minutes of the specified date
- `dateObject.getSeconds()` - returns the seconds of the specified date
- `setInterval(function, milliseconds)` - calls a function at specified intervals (in milliseconds)

## JavaScript commands

### Text manipulation

- `string.toUpperCase()` - converts a string to uppercase
- `string.toLowerCase()` - converts a string to lowercase
- `string.trim()` - removes whitespace from both ends of a string
- `string.split(' ')` - splits a string into an array of substrings
- `string.replace('old', 'new')` - replaces a specified value with another value in a string
- `string.substring(start, end)` - extracts characters from a string and returns a new string
- `string.slice(start, end)` - extracts a section of a string and returns a new string
- `This is my name: ${name}` - template literals allow you to embed expressions in a string
- `stringarray.join(', ')` - joins all elements of an array into a string
- `string.includes('substring')` - checks if a string contains a specified substring
- `string.startsWith('substring')` - checks if a string starts with a specified substring
- `string.endsWith('substring')` - checks if a string ends with a specified substring
- `string.indexOf('substring')` - returns the position of the first occurrence of a specified substring in a string
- `string1 + ", " + string2` - concatenates two strings
- `string.length` - returns the length of a string

### Math operations

- `Math.round(number)` - rounds a number to the nearest integer
- `Math.floor(number)` - rounds a number down to the nearest integer
- `Math.random()` - returns a random number between 0 (inclusive) and 1 (exclusive)
- `Number('123')` - converts a string to a number
- `longFloat.toFixed(2)` - formats a number with a fixed number of decimal places
- `myNumber++;` - increments a number by 1

### Array operations

- `array.push(element)` - adds an element to the end of an array
- `array.pop()` - removes the last element from an array
- `array.shift()` - removes the first element from an array
- `array.unshift(element)` - adds an element to the beginning of an array
- `array.splice(index, count)` - removes elements from an array

- `array.map(element => element * 2)` - creates a new array with the results of calling a function on every element in the array
- `array.filter(element => element > 5)` - creates a new array with elements that pass a test

### Conditional

- `if (condition) return` - stops the execution of a function if a condition is met
- `if (condition) { ... } else { ... }` - executes different code depending on a condition
- `switch (expression) { case x: ... break; case y: ... break; default: ... }` - selects one of many code blocks to be executed
- `cond1 && cond2` - logical AND operator
- `cond1 || cond2` - logical OR operator
- `!condition` - logical NOT operator
- `condition ? value1 : value2` - conditional (ternary) operator

### Loops

- `for (let i = 0; i < array.length; i++) {console.log(array[i])}` - iterates through an array and logs each element to the console
- `for (const element of array) {console.log(element)}` - iterates through an array and logs each element to the console
- `array.forEach(element => {console.log(element)})` - iterates through an array and logs each element to the console
- `array.forEach((element, index) => {console.log(index, element)})` - iterates through an array and logs the index and element to the console
- `if (condition) {break}` - exits a loop if a condition is met
- `if (condition) {continue}` - skips the current iteration of a loop if a condition is met
- `while (condition) { ... }` - loops through a block of code while a specified condition is true
- `do { ... } while (condition)` - loops through a block of code once, and then repeats the loop while a specified condition is true

## Advanced JavaScript

### API requests

- `fetch('url')` - makes a request to the specified URL and returns a promise
- `fetch('url').then(response => response.json())` - converts the response to JSON format. You can add more `.then()` to handle the data.

### Action on page load

- `window.addEventListener('load', function() { ... })` - executes a function when the page is fully loaded

### Error handling

- `e.preventDefault()` - prevents the default action of an event from occurring
- `try { ... } catch (error) { ... }` - handles errors in a block of code