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

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

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

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.addEventListener('mouseover', function() { ... })` - assigns a function to be executed when the mouse pointer is moved onto the element
- `element.addEventListener('mouseout', function() { ... })` - assigns a function to be executed when the mouse pointer is moved out of the element
- `element.addEventListener('change', function() { ... })` - assigns a function to be executed when the value of the element is changed
- `element.addEventListener('input', function() { ... })` - assigns a function to be executed when the value of the element is changed

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

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

Loops

- `for (let i = 0; i < array.length; i++) {console.log(array[i])}` - 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

Filtering

- `element.slice(start, end)` - extracts a section of an array and returns a new array
- `array[idx]` - accesses the element at the specified index in an array

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