Map

Map()	A JavaScript object that stores key-value pairs, where keys can be any type of data, and values can be any data type.
set(key, value)	Adds a new key-value pair to the Map. If the key already exists, it updates the value.
get(key)	Retrieves the value associated with the key from the Map. If the key doesn't exist, returns undefined.
has(key)	Checks if a key exists in the Map. Returns true or false.
delete-(key)	Removes a key-value pair from the Map.
clear()	Removes all key-value pairs from the Map.
forEac-h(call-back)	Iterates over each key-value pair in the Map and executes a provided function.

General

consol e.log()	Outputs a message to the console, commonly used for debugging.

General (cont)

let, const, var	Variable declaration keywords: let (block-scoped), const (block-scoped and immutable), var (function-scoped).
Promises	Represents a value that may be available in the future. It is used for handling asynch-ronous operations.
async/-await	Used to simplify asynchronous code. await waits for a promise to resolve, and async makes a function asynch-ronous.

Teorico

DOM (Document Object Model)	The structure that represents the HTML document as a tree of nodes, allowing dynamic manipulation.
AJAX (Async-hronous JavaScript and XML)	Technique for making asynch-ronous HTTP requests to fetch data without refreshing the page.

Teorico (cont)

JSON (JavaS-cript Object Notation)	Lightweight data-i-nterchange format that's easy for humans to read and write, and for machines to parse and generate.
RESTful APIs	Design principles for building web services that are stateless and use HTTP methods for CRUD operations (GET, POST, PUT, DELETE).
CSS Flexbox	Layout model that allows for responsive and flexible design, aligning elements in rows or columns.
CSS Grid	2D layout system for creating grid-based layouts with rows and columns.
CORS (Cross-Origin Resource Sharing)	A mechanism that allows restricted resources on a web page to be requested from another domain.

Express

express()	The function used to create an instance of an Express application.
app.liste-n(port)	Starts the Express server and listens on a specified port. For example: app.li-sten(3000) to listen on port 3000.
app.ge-t(route, callback)	Defines a route handler for GET requests. For example: app.get('/', (req, res) => res.se-nd('Hello World'))
app.po-st(route, callback)	Defines a route handler for POST requests. For example: app.po-st('/submit', (req, res) => res.se-nd('Data received'))
app.pu-t(route, callback)	Defines a route handler for PUT requests. Typically used for updating data.
app.de-lete(r-oute, callback)	Defines a route handler for DELETE requests. Usually used to remove resources.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Express (cont)

app.use(middleware)	Adds middleware to the Express app. Middleware functions execute during the lifecycle of a request. For example: app.use(express.json()) to parse JSON in request bodies.
res.send()	Sends a response to the client. Can send various types of data such as strings, objects, or arrays.
res.json()	Sends a JSON response. Converts an object or array to JSON format.

NodeJs

require()	Imports modules, JSON, or local files into your Node.js application.
fs module	Provides an API for interacting with the file system, allowing you to read, write, and modify files.

NodeJs (cont)

module.exports	Defines what a module exports so that other files can use it.
fs.readFile(path, encoding, callback)	Asynchronously reads the contents of a file. The callback receives two arguments: an error (if any) and the data.
fs.writeFile(path, data, encoding, callback)	Asynchronously writes data to a file, replacing the file if it already exists. The callback handles error (if any).
fs.appendFile(path, data, encoding, callback)	Asynchronously appends data to a file. If the file does not exist, it is created. The callback handles error (if any).
fs.copyFile(src, dest, callback)	Asynchronously copies a file from src to dest. The callback handles error (if any).

Events

addEventListener()	Attaches an event handler to an element for a specific event.
removeEventListener()	Removes an event handler from an element.
event.target	Returns the element that triggered the event.
event.preventDefault()	Cancels the default behavior of an event (e.g., prevents form submission).
event.stopPropagation()	Stops the event from propagating up or down the DOM tree.
DOMContentLoaded	Fires when the HTML has been completely loaded, but before stylesheets, images, and subframes finish loading.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com