**JSON**

**-w3schools**

JSON is a **syntax** for storing and exchanging data.

The data that can be exchanged between a server and a browser can only be text/string. JSON, as it is text, helps achieve this. XML also serves similar purpose.

JSON and JavaScript objects are inter-convertible.

Any JS object can be converted to a JSON object.

In JSON keys in the objects must be strings whereas in JS they can be strings/identifier names/numbers.

In JSON, values must be one of these data types:

String, number, boolean, JSON object, array, null

In JS, values must be any of the above including:

A function, a date, undefined.

JSON – string values must be written in double quotes.

JS – string values can be in single or double quotes.

JSON.parse(text, **revivor**)

Revivor is a function that **checks each property** before returning the value.

Key and value can be sent as parameters to the revivor function.

JSON.stringify()

JSONP (JSON with Padding) is a **Method** for sending JSON data without worrying about cross-domain issues.

**JavaScript**

Arguments – actual value of the parameter of a function.

Ex: greet(‘world’);

Parameters – variable declared while function declaration

Ex: function greet (name) { }

A variable is a container to store data.

Types of variables:

String, number, Boolean, undefined (used to say that the variable is not defined), null – (used to assign no value or to say that the value is null)

Script tag placement:

Factors - Loading time, JS access to HTML

Browser downloads all the content to render it. When a JS file is encountered, it makes a http request to download the JS file and then executes it. No DOM content is downloaded while a JS file is downloaded and executed.

If the JS file is too large or there are multiple JS files and hence multiple HTTP requests, downloading and executing them can take longer and user experience can be bad as DOM isn’t loaded completely during that time.

So it is best to add script tags at the end of body tag.

A=1;b=”5”;

Arithmetic operations like \*, /, - on a and b will still work as JS will treat **b** as a number(as the string contains only a number) and perform the operations. A+b gives 15.

NaN – Not a Number

* A=5; b=”5” … A==b is true; A===b is false (strict equality) smillarly != vs !==

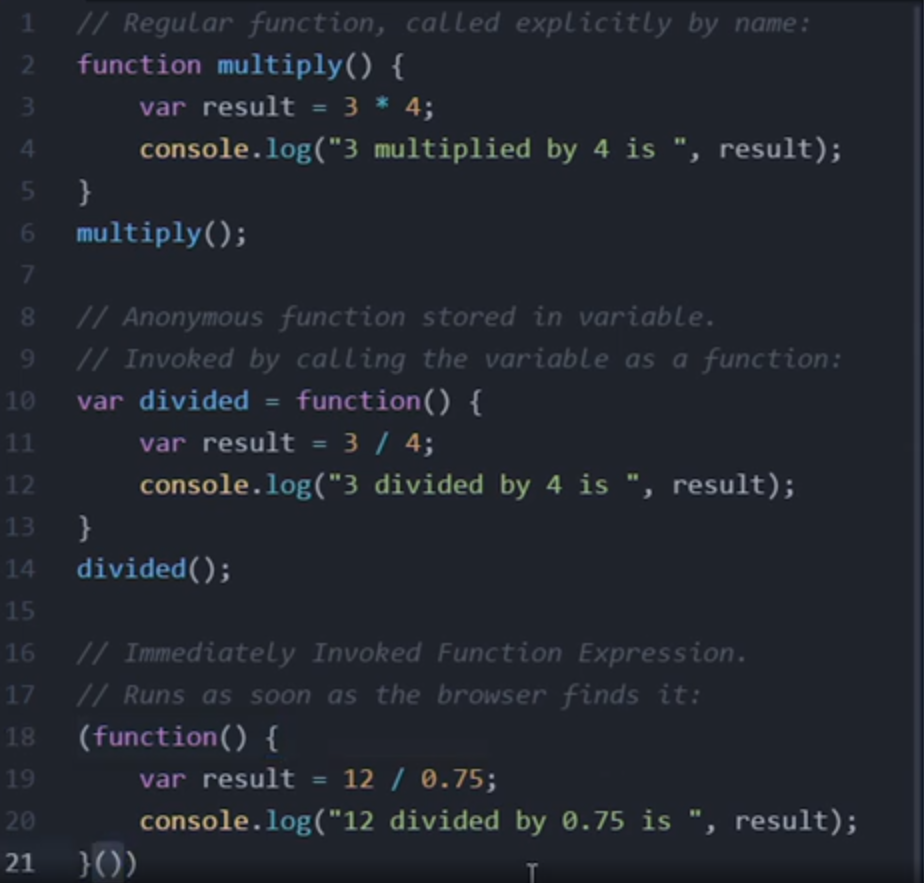
Var temp = new Array();

Array Methods:

* Shift(), unshift(“one”, “two”); pop(); push(); splice(positon, numberOfItemsToRemove); slice();

indexOf(“string”, index); join();

Place of declaration of function is immaterial in JS as the browser downloads the entire JS and then executes the code.



IIFE – Immediately Invoked Function Expresseions -> Browser executes them as soon as it encounters them.

How to create functions that can be used on any variable (ex: toUpperCase)?

Objects and Object constructors.

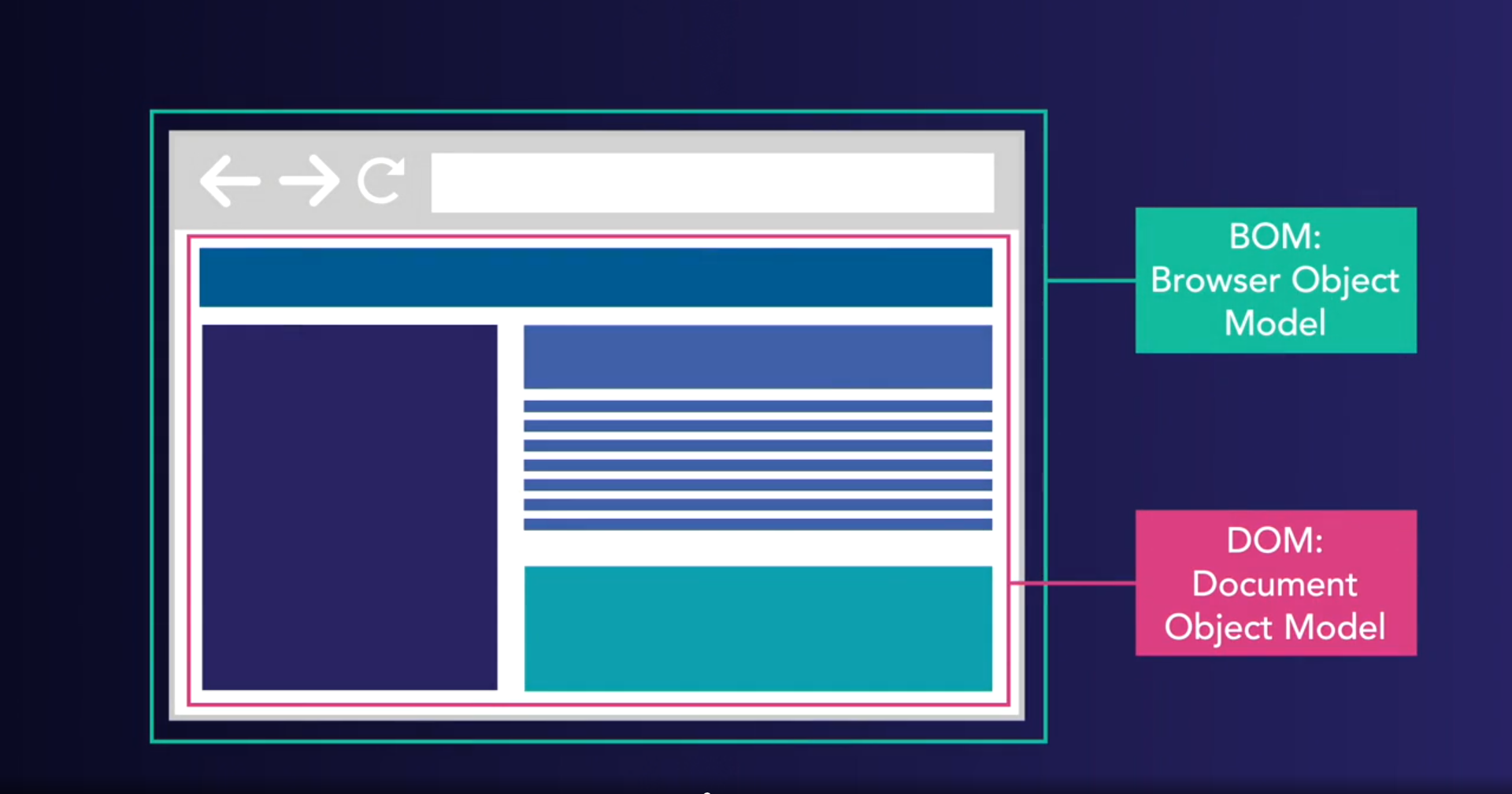
Closure:

A Closure is a combination of function and the lexical environment within which the function was declared.

BOM – Browser Object Model.

DOM – Document Object Model.

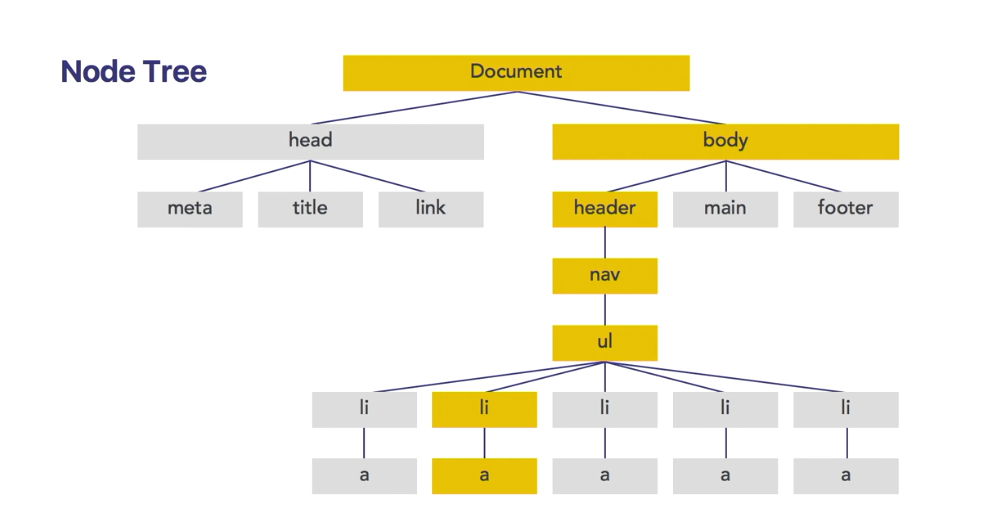
Browser itself is an object. Browser Window is an object.

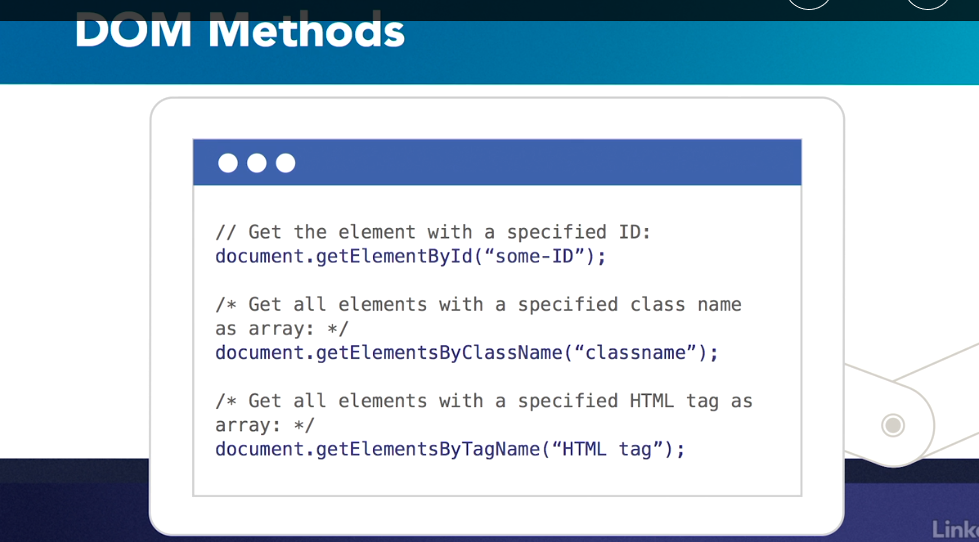


**DOM** is the Model of the Document that forms the current webpage. In other words, it is the way/standard (like class definition) of any Document object available on the Window object.

Each of the HTML element forms a DOM node. Browser treats each DOM node like any other object.

When a document is loaded in the browser, it is loaded into the Document property (object) on the window object present in Browser object (In short, it is loaded into Document Object of BOM) and a DOM is created for just this document instance. The browser now creates a node tree, modelling the relationships between different nodes.





In querySelector() . is used to indicate a css clss, # is used to indicate a css id.

