**Javascript sheet**

1.DOM-

Tree of nodes /element created by the browser.

Js can be used to read/write manipulate to the dom.

Oo representation.

2.Difference between htmlcollection and node list

An html collection is collection of document elements

A nodelist is a collection document nodes.

HTMLCollection items can be accessed by their name, id, or index number.

A nodelist can only be accessed by index.

getElementbyclasname() and getelementbytagname() method return a live htmcollection.

Queryselectorall() method returns a static nodelist.

Traversing the dom

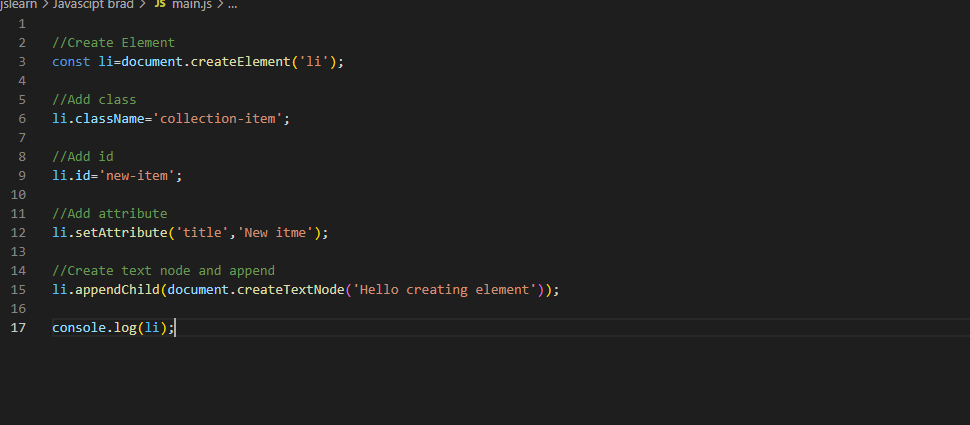
\* Getchildnode🡪returns a nodelist

\* Getchildrenn🡪returns a htmlcollection

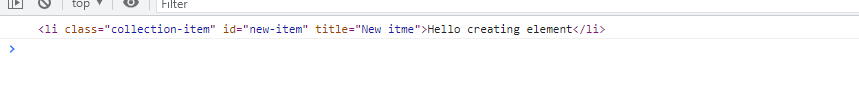
Create element in dom ->document.createElement(tagname)

QuerySelector return the first element within the document that matches specified selector.

QuerySelectorall return a static nodelist representing a list of docments element that match the specified group of selector.



Output---



Event delegation🡪

Instead of adding an event listener to each and every element, we can add event listener to parent element and call an event on a particular target using the .target property. In simple terms putting the event listener on parent .

EX- e.target.classList.contains(‘something’)

JSON🡪

Format of json is text only(string).

Json string -> JSON.parse() gives === Javascipt object

Javascipt object 🡪JSON.stringify() gives == json string

JSON makes it possible to store JavaScript objects as text.

3.OO IN JS

Prototypes are the mechanism by which JavaScript objects inherit features from one another.

ES5-Prototypes

ES6-Classes ,extends ,super

4.Asynchronous

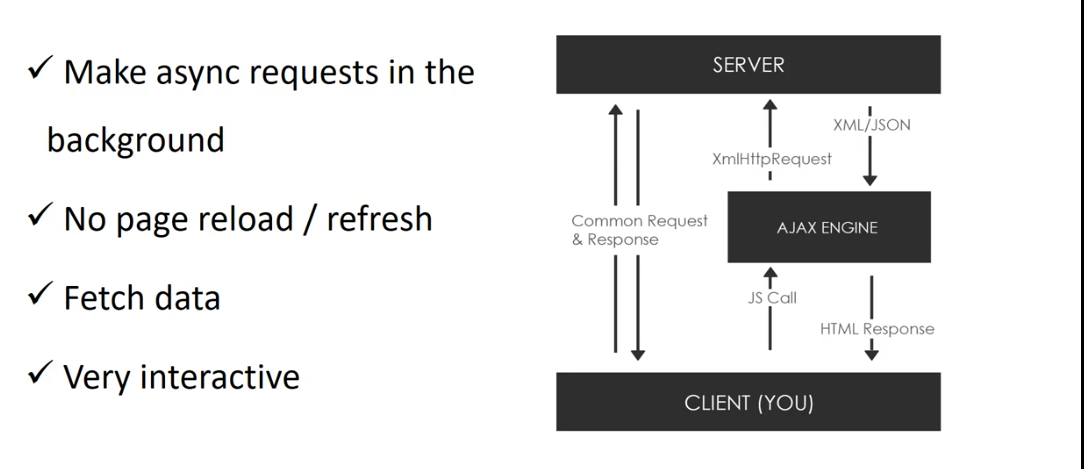
->Three ways to work with async code

\*Callbacks

\*Promises

\*Async/Await

-> What is Ajax? Asynchronous javascript and xml



->readyState Values

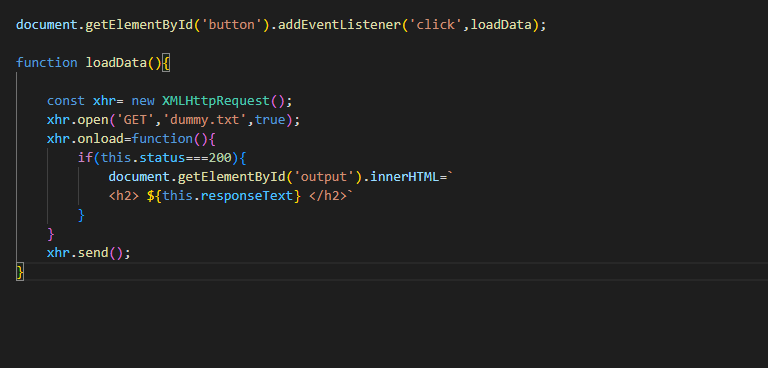
0: request not initialised

1: server connection established

2: request received

3: processing request

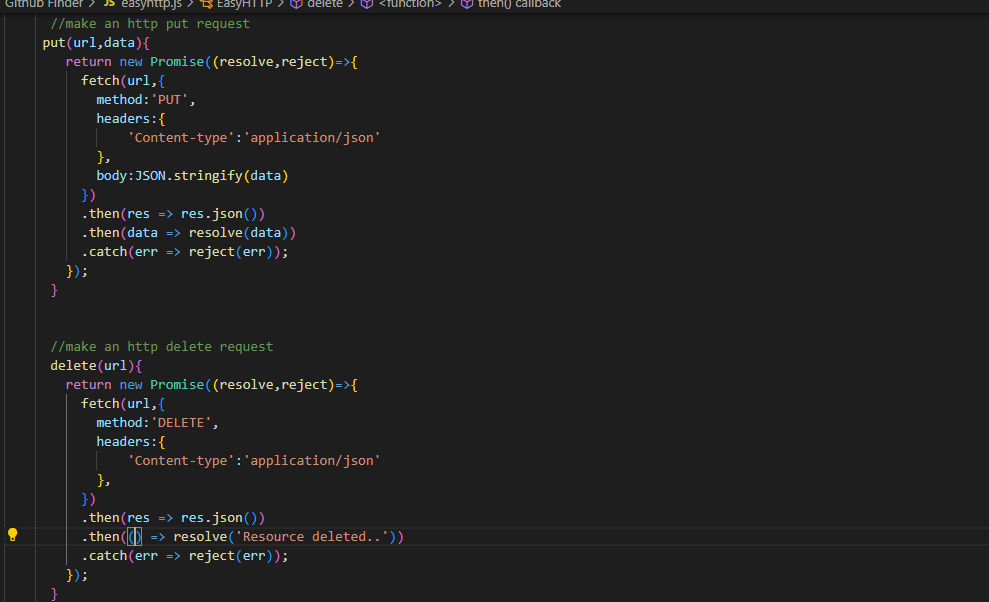
4: request finished and response is ready



Callback -----advancement-----🡪Promises -----advancement-----🡪Async await

Concept-Created custom http library using ES6 (Fetch with Promises)



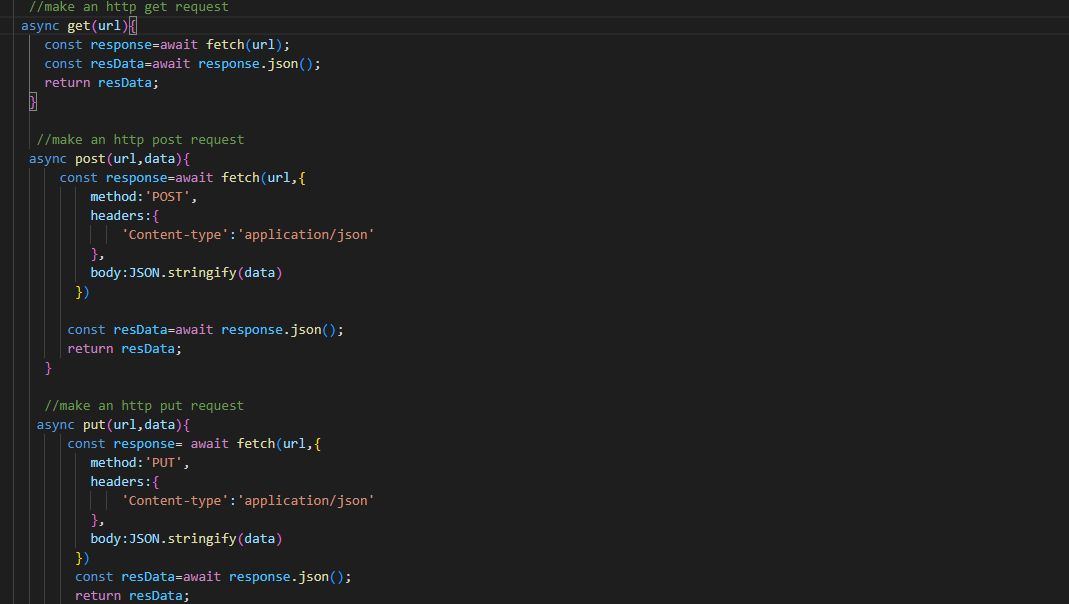


Called using fake rest api—



Custom http library with -async await

Await-Wait until the promise is resolved .



5.Error handling 🡪

Regular expression 🡪

Re is enclosed in / re/

Exec() -return result in an array or null , it matches with the given regular expression .

Test()- return true or false

For making case insensitive we use iflag in regular expression

Match()- return result array or null , opposite of exec

Search() – return index of the first match if not found return -1

//meta character symbols

Re = /^h/I // must start with h

Re = /d$/I // must end with the character given before money sign

Re = /h.llo/I //matches any one character

Re = /h\*llo/I // matches any character 0 or more times

Re=/gre?a?y/I //optional character before ?

re = /gre?a?y\?/I //escape character

//Brackets [ ] – character sets

Re = /gr[ae]y/I //must be an a or e

Re = /[^GF]ray/I // match anything except G or F

Re =/[A-Z]ray/ // match any uppercase letter

//Braces { } -quantifiers

Re = /hel{2}o/I // must occur exactly {m} amount of time

Re = /hel{2,4}o/I // must occur between {2,4} times the character l

Re = /hel{2,}o/I // must occur at least 2 times

//Paranthesis( )- grouping

Re = /([0-9]x){3}$/

// Shorthand character classes

Re =/\w/ //word character – alphanumeric or \_

Re =/\w+/ // += one or more

Re =/\W/ //non word character

Re =/\d+/ //match any digit 0 or more times

Re = /\D/ // match any non-digit

Re = /\s/ // match whitespaces

Re = /\S/ // match non-whitespaces char

//assertion

Re = /x(?=y)/ //match x only if followed by y

re = /x(?!=y)/ //match x only if not followed by y

6. Pattern in js

Module pattern –

