Java script road map

Basic

Topics to learn

- What is Java Script & Its features
- Variables in JS
- Data types
- Operators
- Conditional &Control statements
- Loops
- Data structures
- Scope
- Objects
- Functions
- Form Validations
- DOM Manipulation
- Events
- BOM Elements

Topics to learn

Advanced

- Prototyping
- Proto type Inheritance
- Call backs
- IIFE
- Promises
- Ajax
- Async Await
- Closures
- OPPs Concepts
- JSON

Variables in Java script

Variables in js

What is variable

How to declare variable

Rules to name a variable

Variables in js

How to create variable :

- var
- let
- const

Variables in js

Rules for variable names:

- Variable names contain only letters(a-z,A-Z),digits (0-9),_, $
- Should start with letters(a-z,A-Z) or _
- Variable names are case sensitive
- Camel Case is a convention to name variables

Data types in Java script

Data types in Js

primitive

- Number
- String
- Boolean

Special

- Undefined
- Null

Composite

- Array
- Object

Operators in Java script

Operators in Js

- Arithmetic Operators
- Comparison Operators
- Logical Operators
- Assignment Operators
- Type Operators
- Bitwise Operators
- Ternary Operators

Operators in Js

Arithmetic Operators

- Add 2 + 4 → 6
- Subtract 2 - 4 → -2
- Multiply 2 * 4 → 8
- Divide 2 / 4 → 0.5
- Modulus 5 % 2 → 1
- Negate -(5) → -5

Operators in Js

Comparision Operators

- a == b True if a and b are the same
- a != b True if a and b are not the same
- a > b True if a is greater than b
- a >= b True if a is greater than or equal to b
- a < b True if a is less than b
- a <= b True if a is less than or equal to b

Operators in Js

Comparison Operators

Ex:

```
if ( x != o )
        result = y / x;
else
        result = “not defined”;
```

Logical Operators

Operators in Js

a && b	AND	True if both are true
a || b	OR	True of either or both are true
!a	NOT	True if a is false

Operators in Js

Assignment Operators

=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
>>>=	x >>>= y	x = x >>> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y
**=	x **= y	x = x ** y

Bitwise Operators

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

Ternary Operator in Java script

Ternary Operator in Js

Syntax

```
condition ? expression_1 : expression_2;
```

Ex:

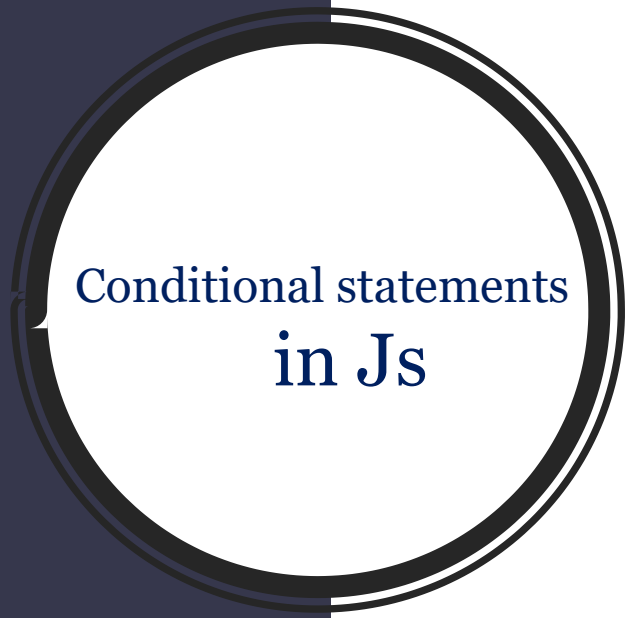
```
big  = a>b ? a : b;
```

Conditional Statements in Java script

Types of control statements: Conditional statements

Loops/iterative statements

Types of conditional statements:



1. if statement
2. if-else statement
3. if-else if - else statement
4. Switch statement

Conditional statements
in Js

Syntax of if statement

```
if(condition){

        statement 1
        statement 2
        .
        .
        .
        statement n


}
```

Conditional statements
in Js

Syntax of if –else statement

```
if(condition){

        statement 1
        statement 2
        .
        .
        .
        statement n

}

else{
        statement 1
        statement 2
        .
        .
        .
        statement n

}
```

Conditional statements
in Js

Syntax of if-else-if statement :

```
if(condition 1 is true) {
    // code is executed

}

else if (condition 2 is true) {

    // code is executed
}

else {
        // code is executed
}
```

Conditional statements
in Js

Syntax of switch statement

```
switch (expression) {

case choice1:
        run this code
        break;

 case choice2:
        run this code instead
        break;

 default:
        no case matches run this code

}
```

Loops in Java script

Loops
in Js

Why loops(iterative statements) are needed:

- Time will be saved
- Code length will be reduced
- Readability increases

Loops
in Js

Types of loops :

- for loop
- while loop
- do-while -loop
- for-in-loop
- for-of-loop

Loops
in Js

Syntax of for loop :

```
for(initialization; condition; increment) {
        // Code to be executed
}
```

Loops
in Js

Syntax of -while loop :

```
do {
    // Code to be executed
}
while(condition);
```

Loops
in Js

Syntax of do-while loop :

```
while(condition) {

    // Code to be executed

}
```

for in vs for of loops in Java script

Functions in Java script

Functions
in JS

Topics:

- What is function
- Why functions
- How to create functions
- How to call function
- What is function expresion

Functions in JS

What is Function:

A block of code written to do a particular task

What is Block:

Anything that enclosed between { }

Ex: {

statement 1 ;

statement 2 ;

statement n ;

}

Functions in JS

How to create function/Function definition:

```
function functionName(parameter1,parameter2 ...parameter n) {
        body
        return value;
}
```

return & parameters are optional

```
Ex:  function  sum(num1,num2) {
        var result = num1+num2
        return  result;
}
```

Functions
in JS

How to call functions:

functionName(parameters)

Ex: sum(7,8)

Functions
in JS

What is function expression:

Assigning function to a variable

Ex:

```
add =  function  sum(num1,num2) {

        var result = num1+num2

        return  result;

}
```

add(7,8) => 15

sum(7,8) => Error

Types of Functions in Java script

Functions
in JS

Types:

- Named functions
- Anonymous function
- Immediately Invoked function expression(IIFE)
- Arrow function

Functions in JS

Named function:

```
function functionName(parameter1,parameter2 ...parameter n) {
        body
        return value;
}
```

Ex:

```
function sum(num1,num2) {
        var result = num1+num2
        return result;
}
```

Functions in JS

Anonymous function:

```
function (parameter1,parameter2 ...parameter n) {

        body

        return value;

}
```

```
Ex:  function  (num1,num2) {

        var result = num1+num2

        return  result;

}
```

we cannot write anonymous function directly

we have to assign it to a variable

Anonymous function:

Functions
in JS

```
Sum =   function  (num1,num2) {

            var result = num1+num2

             return  result;

}
```

Functions in JS

IIFE:

When we want to execute a function immediately where they Created,IIFE used.

Syntax: (function definition) ( );

```
(

    Ex:  function  product (num1,num2) {

            var result = num1*num2

            return  result;

    }


) ( 6,8 );
```

Functions in JS

Arrow function:

```
var  any name = (parameter1,parameter2 ...parameter n) =>{
        body
        return value;
}
```

Ex:

```
var product =  (num1,num2) =>  {
        var result = num1*num2
        return  result;
}
```

Calling: product(5,8)

Functions in JS

Arrow function forms :

1. If only one statement in function body

```
var product =  (num1,num2) =>  num1*num2
```

2. If only one parameter

```
var cube =  num1 =>  num1*num1*num1
```

```
var cube =  (num1 )=>  num1*num1*num1
```

```
var cube =  (num1 )=> { return  num1*num1*num1 }
```

Calling: cube(5)

Functions
in JS

Arrow function forms :

3. If no argument

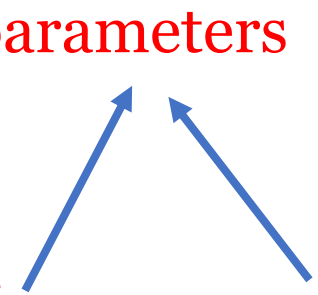
```
Var  greet = ( ) => console.log(“Good Morning");

Var  greet = _ => console.log(“Good Morning");
```

Calling: greet( )

Default and Rest parameters in Java script

Function parameters in JS

Function definition:




```
function  product(num1,num2) {

        var result = num1*num2

        return  result;

}
```

Function call: product(8,4) ->32

product(8) -> ?

Default parameters:

```
function  product(num1,num2=1) {

        var result = num1*num2

        return  result;
}
```

Sum(8) -> 8

Sum(8,7) -> 56

Sum(8,7,9) -> 56

Functions
in JS

Rest parameters:

We can make function to accept unspecified number of parameters

Rest parameter

```
function  product(num1=1,num2=1,  ...arr) {

        var result = num1*num2

        return  result;
}
```

***Use ... before rest parameter ***

Functions
in JS

Rest parameters:

```
function  product(num1=1,num2=1,  ...arr) {

        var result = num1*num2

        for (num of arr){

                result = result*num

        }

        return  result;
}
```

Functions
in JS

forEach in Java script

forEach
in JS

For every element present inside array the specified function will be executed.

Syntax :

```
arr.forEach (function funName(currentItem,index,arr)) {

        //code here

});
```

currentItem :Required

Index : optional

arr : optional

forEach
in JS

For every element present inside array the specified function will be executed.

Syntax :

```
arr.forEach (functionName);
```

```
arr.forEach (test);
```

```
function test(item, index, arr) {
 code
}
```

Objects in Java script

Object:

- object is collection of elements in the form of properties and methods.
- Property is a key-value pair.

Objects
in JS

Ex:

```
Var user1 = {
        name:'xyz',
        email: xyz@gmail.com,
        mobile: 9898987898 ,
        login : function (){
          console.log("login success")
        }
    };
```

Ways to create object :

Objects
in JS

1.Object Literal

2.Using a new keyword with Object constructor

3.Using a new keyword with a constructor function

4.Object.create() method

5.Classes

1. Using Object Literal :

Create object in JS

```
let movie1 = { }                    -> Create an empty Object


let movie = {                       ->Object with properties (key: value pairs)
                name: “RRR"
                release : 2021
                director : “Rajamouli"
}
```

How to access object values :

We can access values by using keys in 2 ways

1. obj["key"] -> quotes are mandatory

   Ex: movie[“name"] ->valid

   movie[name] ->Invalid

2. obj.key

   Ex: movie.name

Objects
in JS

How to add new properties to object:

We can add properties by using keys in 2 ways

1. obj["key"] = value

   Ex: movie[“budget"] = “400 crores”

Create object in JS

2. obj.key = value

   Ex: movie.budget =“400 crores”

How to update values of object :

We can update values by using keys in 2 ways

1. obj["key"] = value

   Ex: movie[“budget"] = “500 crores”

Create object in JS

2. obj.key = value

   Ex: movie.budget =“500 crores”

2. Using new Operator Object Constructer:

Create object in JS

```
let movie1= new  Object();

movie1 .name = “RRR”;

movie1 .director = “Rajamouli”;
```

3. Using new Operator with Constructer function:

Create object in JS

Step1: create constructor function :

```
function user(name, age,place) {
        this.name = name;
        this.age = age;
        this.place = place

    }
```

Step2: create object with constructor function call :

```
let movie1= new  movie(“abc", 25,”Hyderabad”);
```

4. Object.create method:

```
let movie2 = Object.create(movie);

let movie3  = Object.create(movie, {
                    name: {
                    value: “RRR
                    },
                    Music: {
                    value: “Keeravani”
                    }
})
```

Create object in JS

Console Methods in Java script

Console Methods in JS

- console.log()
- console.info()
- console.warn()
- console.error()
- console.table()
- console.clear()
- console.assert()
- console.count()
- console.countreset()
- console.time()
- console. Timelog()
- console.timeend()
- console.group()
- console.groupend()

DOM in Java script

DOM
in JS

What is DOM

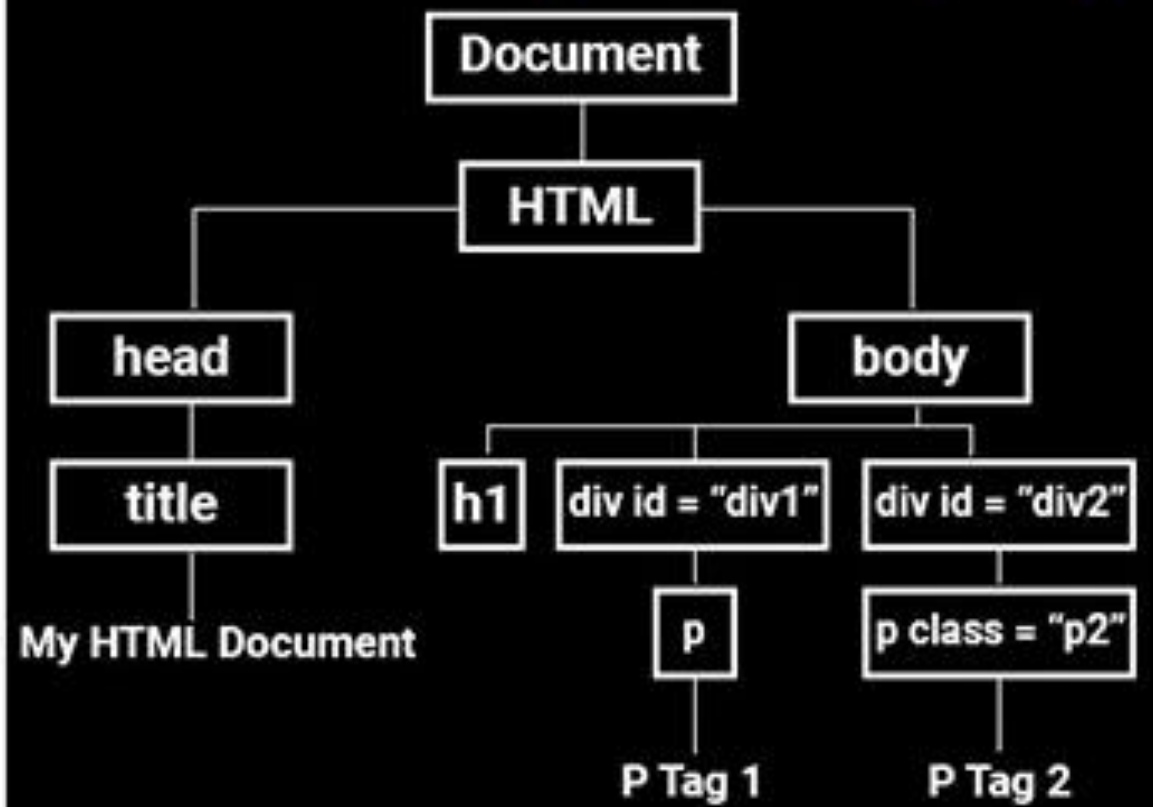
- DOM : Document Object Model
- By using HTML DOM JavaScript can access, change or remove any elements of HTML document and also can create new elements at any position.
- When a webpage loaded, browser create DOM of webpage

HTML Document

index.html

```
1   <html>
2       <head>
3           <title>My HTML Document</title>
4       </head>
5
6       <body>
7           <h1>Heading</h1>
8           <div id="div1">
9               <p>P Tag 1</p>
10          </div>
11          <div id="div2">
12              <p class="p2">P Tag 2</p>
13          </div>
14      </body>
15  </html>
```

Document Object Model (DOM)



Document Object in JS

Document Object has

- Properties
- Methods

Using this document object properties& methods we can

- Select HTML elements
- Modify HTML elements
- Remove/delete HTML elements
- Create HTML elements
- Add/Remove /Change styles to HTML elements

Methods to select HTML Elements :

1.document.getElementById(“idname”)

Returns element with the specified id

2. document.getElementsByClassName(“classname”)

Returns list of all elements belongs to the specified class

3. document.getElementsByTagName(“tagname”)

Returns list of all elements with the specified tag

4. document.querySelector(“.class/#id/tagname”)

Returns the first object matching CSS style selector

5. document.querySelectorAll(“.class/#id/tagname”))

Returns all objects Matches the CSS Style Selector

DOM properties to select HTML Elements :

- document.body
- document.head
- document.title
- document.anchors
- document.forms
- document.images
- document.scripts

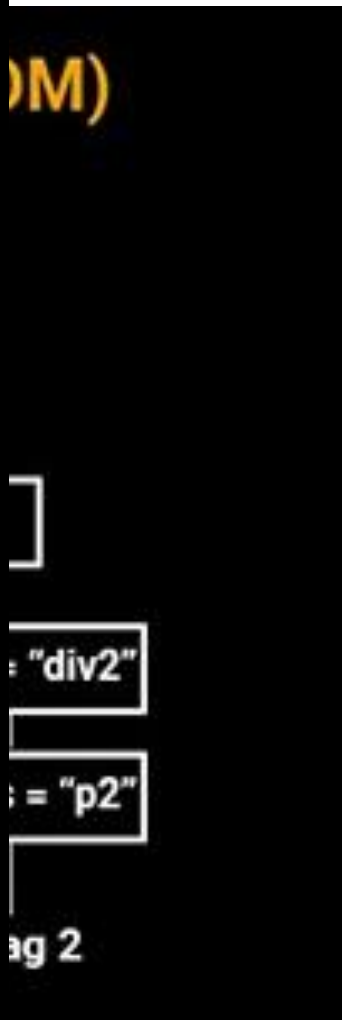
DOM Manipulation in Java script

HTML Document

index.html

```
1  <html>
2      <head>
3          <title>My HTML Document</title>
4      </head>
5
6      <body>
7          <h1>Heading</h1>
8          <div id="div1">
9              <p>P Tag 1</p>
10         </div>
11         <div id="div2">
12             <p class="p2">P Tag 2</p>
13         </div>
14     </body>
15 </html>
```

```
<html >

<head>
  <title>My HTML Document</title>
</head>

<body>

  <h1>Heading</h1>
  <div id="div1">
    <p>P Tag 1</p>
  </div>
  <div id="div1">
    <p class="p2">P Tag 2</p>
  </div>
</body>

</html>
```



Topics :

DOM Manipulation in JS

- How to create HTML elements
- How to set content to element
- How to append a new element
- How to insert an element before another element
- How to remove an element
- How to remove child element
- How to replace an element

Add/change styles in Java script

CSS Styling
in JS

Topics :

- How to change style of element
- How to get attribute value of an element
- How to set/change attribute of an element

Modal Windows in Java script

Modal windows
Java script

3 modal windows :

- Alert – to show message to user
- Prompt- To take some input from user
- Confirm – to get confirmation from user

Events & Event handling in Java script

Events
& Event handling in
Java script

Topics :

- What is event & types of events in JS
- What is event handler
- Ways to handle events in JS

Events &
Event handling in
Java script

Event

- Event is nothing but an action
- In webpage also what ever user do everything is an event

Who generates events:

- user –> keypress,scroll,focus,....
- System ->load,error,abort..

Types of user generated events:

- Browser specific events ->scrollup/down resize browser....
- DOM /web page specific events -> click,hover,focus ....

Event handler is function we write to run when an event happens

Commonly used events in JavaScript:

Events &
Event handling in
Java script

Mouse Events

- click
- double click
- mouseover
- mouseout
- mouse move

Keyboard Events

- Key down
- Key up
- Key press

Focus events

- focus
- blur
- focusin
- focusout

Form Events

- submit
- reset
- change

Events &
Event handling in
Java script

Ways to handle events:

- Inline event handlers(using event attributes in HTML)

  Ex: <button onclick="JavaScript code">

- using event properties in JavaScript

  Ex: let btn =document.getElementById('b1')

  button.onclick = function name/anonymous function

- using addEventListener() method in JavaScript

  Ex: let btn =document.getElementById('b1')

  button.addEventListener(eventname,function name/anonymous fun)

Higher order functions in Java script

Higher order functions in Java script

- Higher order function is a function that receives another function as argument or return another function or both

Example of receiving another function as argument:

```
function first(fun){

        fun()

      }

 function second(){
          console.log("This is second function")
 }

 first(second)
```

Higher order functions in Java script

Example of returning a function:

```
function first(){

        return second

      }

 function second(){
          console.log("This is second function")
}

 let res = first()

  res()
```

This keyword in Java script

This
in Js

- Value of this keyword outside a function /object
- Value of this keyword inside a regular function
- Value of this keyword inside a method
- Value of this keyword inside event handler
- Value of this keyword inside a function in strict mode

Math Object in Java script

Math Object

Methods of Math Object

- Sign()
- abs()
- sqrt()
- pow()
- max()
- min()
- round()
- ceil()
- floor()
- trunc()
- random()
- exp()
- log()
- log2()
- log10()

String Methods in Java script

Window Object in Java script

Window Object

properties of Window Object

- document
- innerWidth
- outerWidth
- innerHeigth
- outerHeight
- location
- closed
- Name
- navigator

Open() & close() browser window with sJava script

Window Object

Methods of Window Object

- Open()
- Close()
- moveBy()
- moveTo()
- alert()
- Prompt()
- confirm()
- reSizeTo()
- reSizeBy()
- print()

open() Method syntax:

Window Object

Window.open(URL , name/target , specifications)

- _blank
- _self
- _parent
- _top

- width
- height
- left
- Top
- Resizable
- scrollable

moveTo() & moveBy()
resize()
browser window
with Java script

Window Object

- moveTo()
- moveBy()
- reSizeTo()
- reSizeBy()

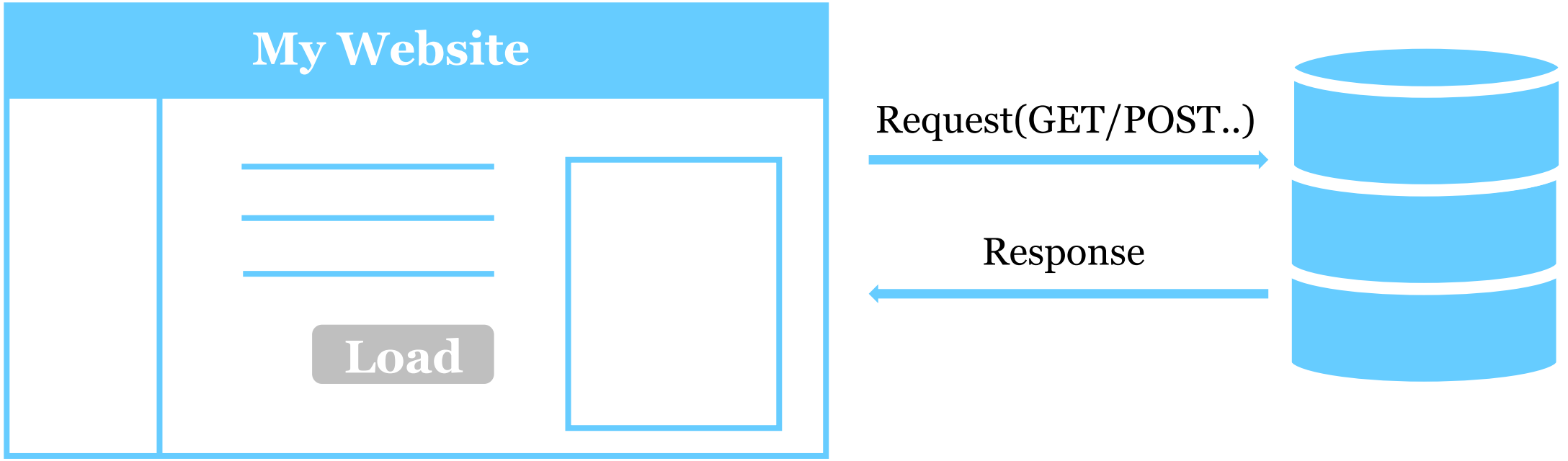
Asynchronous Java script

AJAX

AJAX

- AJAX – Asynchronous JavaScript And XML
- AJAX helps in fetching data asynchronously from a remote web server
- Data loaded by AJAX call is done asynchronously with out page refresh
- Web server will send response which contains data that we have requested
- Data can be of any format like JSON,XML …
- Initially servers used to send data in XML format

AJAX



AJAX

- In JavaScript we have to use XMLHttpRequest object to make AJAX call to exchange data from webserver.
- AJAX request can be send to sever in with 3 steps

1. Create XMLHttpRequest object

Ex: let xhr = new XMLHttpRequest()

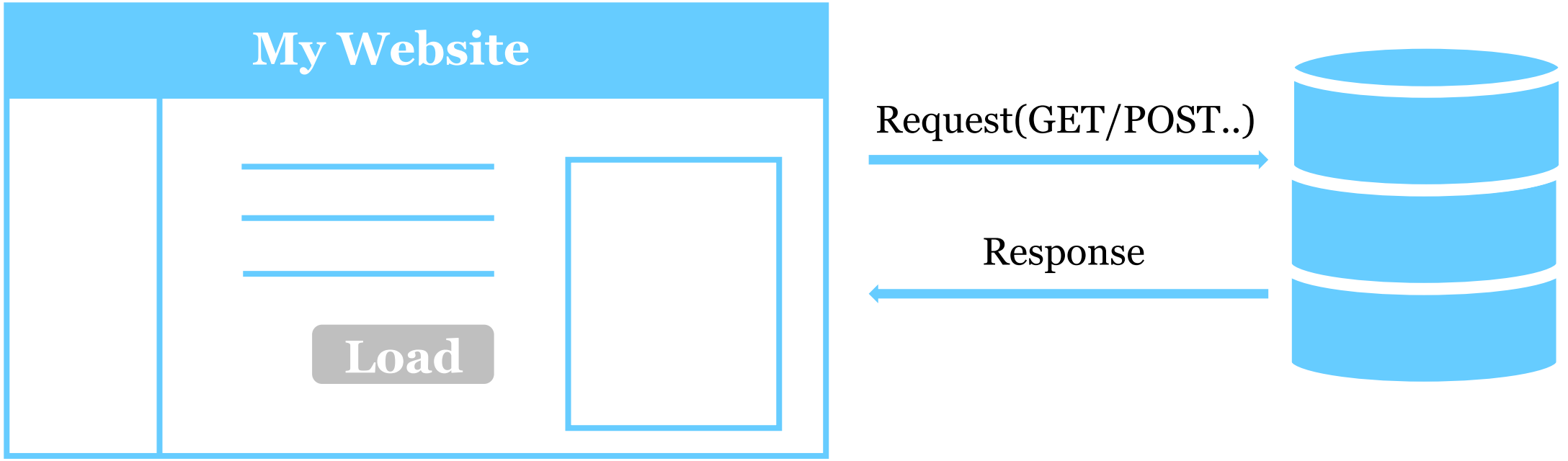
2. Create request with that object open() method

Syntax : open(method,url,async,username,password)

3. Send the request using send() method

AJAX

AJAX



Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
open(method, url, async, user, psw)	Specifies the request method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information

Properties

Property	Description
onload	when the request is complete and the response is fully downloaded.
onreadystatechange	Defines a function to be called when the readyState property changes
onprogress	triggers periodically while the response is being downloaded, reports how much has been downloaded.
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	Returns the status-text ("OK" or "Not Found")

What is Call Back Hell

Promises In JavaScript

Promise

- Promise is an object

```
let p = new promise()
```

- Promise object takes a callback function

```
let p = new promise(  function(){})
```

- This object returns some data either success data or error information.

```
let p = new promise(  function(resolve,reject){

})
```

Promise

- States of promise

  Pending

  fulfilled

  rejected

Fetch API In JavaScript

Fetch API

- fetch() is a method of browser window Object which helps to make AJAX call

  Syntax : fetch (url,[options])

  options: Optional parameters like method, headers…

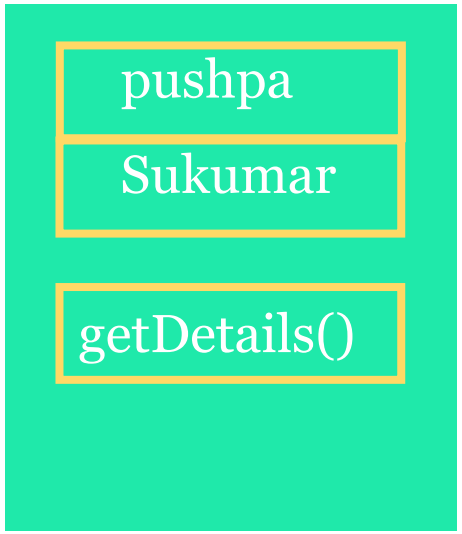
- fetch() method returns a promise
- So to use the data from server we have to use then() method of promise return by fetch() method

```
let promise  = fetch("https://restcountries.com/v3.1/name/india")

promise.then(function(response){
    console.log(response)

})
```

Prototype & __proto__ In JavaScript

Prototype

```
function movie(name,director){
        this.moviename=name
        this.director = director
        this.getGetails = function(){
            console.log("Director of ",this.moviename,":",this.director)
        }
    }
```




```
movie1 = new movie("Pushpa","Sukumar")
```

```
movie2 = new movie("RRR","Rajamouli")
```

Regular Expressions

- Regular expression is a sequence of characters, helps to create a search pattern

Ways to create regular expressions:

1. RegExp literal notation

   Syntax : /pattern/[flags]

   Ex: /hello/

   /^[0-9]{10}$/

2. Using RegExp Constructor function

   Syntax : new RegExp('pattern',[flags])

   Ex: new RegExp("hello")

   new RegExp("^[0-9]{10}$")

Regular Expressions In JavaScript

Regular Expressions

- Regular expression is a sequence of characters, helps to create a search pattern

Ways to create regular expressions:

1. RegExp literal notation

   Syntax : /pattern/[flags]

   Ex: /hello/

   /^[0-9]{10}$/

2. Using RegExp Constructor function

   Syntax : new RegExp('pattern',[flags])

   Ex: new RegExp("hello")

   new RegExp("^[0-9]{10}$")

Brackets:

Expression	Description	Example
[……..]	Any one character between the brackets.	[a89] => a or 8 or 9
[^……]	Any one character not between the brackets	[a89] => everything except a , 8 and 9
[0-9]	Any number from 0 -9	0 /1/2 ..../9
[a –z]	Any lower case alphabet	a/ b/ c/ ……/z
[A - Z]	Any Upper case alphabet	A/B/……/Z
[a-Z]	Any lower case or uppercase .	a/ b/ c/ ……/z , A/B/……/Z

Quantifiers: For specifying the number of occurrences of a character

Quantifier	Description	Example
*	Match zero or more times.	Bm* => "B", "Bm", "Bmm", "Bmmm", and so on...
+	Match one or more times.	j+ => “j", “jj", “jjj“ ....
?	Match zero or one time.	bS? => “b" or “bs", but not “bss” ....
{ n }	Match exactly n times.	Jab{5} . => “Jabbbbb"
{ n ,}	Match at least n times.	mb{3, } => “mbbb", “mbbbb“ .....
{ n , m }	Match from n to m times.	br{3,5}a => “brrrr“ or “brrrrr“ or “brrrrrr“

Split() & join()
In JavaScript

Split(): It divides a string into an array of substrings based on separator

```
let movies = "RRR BhemlaNayak Pushpa Sarainodu";

console.log(movies.split(' '))

Output: ['RRR', 'BhemlaNayak', 'Pushpa', 'Sarainodu']
```

Syntax: `string.split(separator, limit)`

```
console.log(movies.split(' ', 2))

Output: ['RRR', 'BhemlaNayak']
```

Join(): It concatenate all elements of an array & returns new string separated by separator

```
let movies = ['RRR', 'BhemlaNayak', 'Pushpa', 'Sarainodu'];

console.log(movies.join('-'))
```

Output: "RRR-BhemlaNayak-Pushpa-Sarainodu"

Syntax: `array.join(separator)`

```
console.log(movies.join())
```

Output: "RRR,BhemlaNayak,Pushpa,Sarainodu"

Multi dimensional Arrays In JavaScript

Multi dimensional array

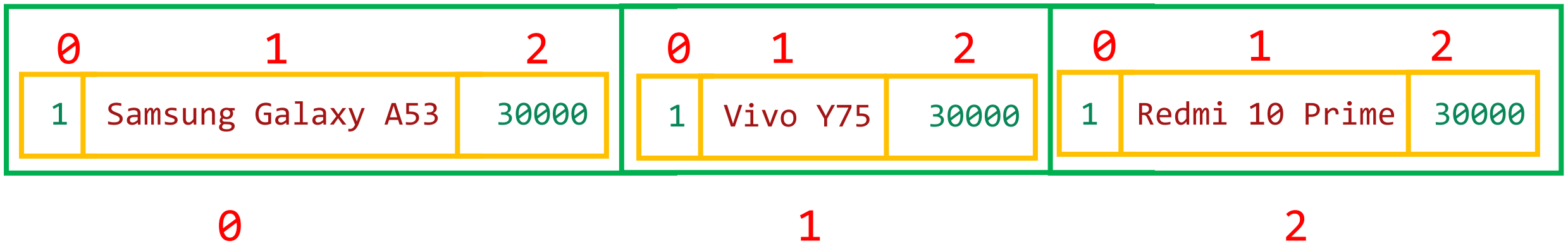
```
let product = [1,"Samsung Galaxy A53",30000]
```

Multi dimensional array

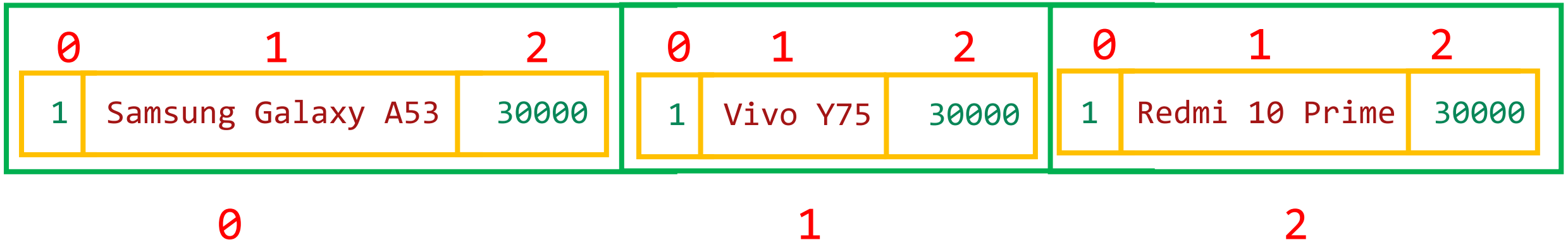
```
let product = [1,"Samsung Galaxy A53",30000]
```


```
let products = [
                  [1,"Samsung Galaxy A53",30000],
                  [2,"Vivo Y75",20000],
                  [1,"Redmi 10 prime",13000],
                ]
```

```
let products = [
                  [1,"Samsung Galaxy A53",30000],
                  [2,"Vivo Y75",20000],
                  [1,"Redmi 10 prime",13000],
                ]
```



```
let products = [
                  [1,"Samsung Galaxy A53",30000],
                  [2,"Vivo Y75",20000],
                  [1,"Redmi 10 prime",13000],
                ]
```




```
products[0] =[1,"Samsung Galaxy A53",30000]

products[0][0] = 1
products[0][1] = Samsung Galaxy A53
products[0][2] = 30000
```