---------- **20/8/24** ----------

TYPECASTING

Process of converting one datatype into another datatype.

Types of typecasting

1. Implicit
2. Explicit

parseInt() -> to convert string into number

JS OUTPUT METHODS

1. **console.log():** console → object, log → function

   -to print in the console window

2. **document.write() / document.writeln()**
   -to print the details in webpage

3. **window.alert();**
   -return type void

4. **window.confirm();**
   -returns boolean value

5. **window.prompt();**
   used to take user input
   returns string value

[3, 4, 5 -> popup method]

How internally code execution happens? / How JS Engine Works

- JavaScript Engine: it is execution unit of browser

- It is responsible for code execution in client side

Browser	JS Engine
Chrome	V8
Internet Explorer	Chakra
Firefox	Spider Monkey
Safari	JS Core

---------- **21/8/24** ----------

Variable Hoisting:

- Moving variable declaration at the top of the scope.

- variable hoisting happens for var, let and const variable.

- if we try to access var variable before the variable declaration it will return undefined

- if we try to access let and const variable before the declaration it will throws error i.e. uncaught reference error because of Temporal Dead Zone(TDZ).

Function Hoisting:

- Moving a function declaration to the top of the scope is called as function hoisting.

- function hoisting will happen only in named function.

Use Strict:

-it is used to follow the strict syntax in JavaScript code

Window:

- supermost object

- refers to the browser window

- If we want to access the methods / properties from window object without reference/name we can access

- if you declare any variable with var keyword it will get stored in the window object

Types of Functions:

1. **Anonymous function**

- Function with no function name
- we could not able to define the anonymous function alone
- Syntax: function() {

```
    //code
}
```

2. **Named function**

- the function with function name

3. **Function with Expression**

- storing the function as a data to a variable is called as function with expression
- Syntax: const fn =function() {

```
    //code
}
```

---------- **22/8/24** ----------

4. Arrow Function

- ES6: ECMA SCRIPT 6 →2015

- used to reduce the syntax

- Syntax: var a1 = () => {

```
        //code
        }
    ar();
```

(): parameters

=>: fat arrow

NAMED FUNCTION	ARROW FUNCTION

function demo (v) => {
 	
return v+10;
}
	const a = v => clg(v);
	
	const demo = v => v+10;

5. HOF: HIGHER ORDER FUNCTION

- function which takes another function as an argument.

6. CALLBACK FUNCTION

- function which we pass as an argument for HOF.

7. NESTED FUNCTION

- one function is present inside another function

CALL STACK:

- keep track on function calls.

- follow LIFO (Last In First Out) principle.

LEXICAL SCOPING OR SCOPE CHAINING:

- The ability of the function scope to access the variable till it reaches the global scope.

- If we try to access the variable within the innermost function first it will check in the local scope if the variable is not there then it will check in the parent function scope, till it reaches the global scope. This process is called as lexical scoping or scope chaining.

CLOSURE:

- it is a scope or memory allocation which gets created when we access the outer function variable inside the inner function. Then the closure will gets created for the outer function.

---------- **23/8/24** ----------

8. IMMEDIATE INVOKE FUNCTION EXPRESSION

- it is a technique to avoid the global pollution
- Syntax:

```
( function () {
        //code  →function declaration
}
) ( ); →function call
```

ARRAY

- non-primitive datatype
- we can store multiple values or data (any datatype)
- array length is not fixed
- it is the index data structure; index position starts from 0
- var a = [10, true, [100, 200], 2000, null, undefined] -> length:6

    Length defines: total number of values are present

- length = last index + 1

Ways to create array in JS

1. literal way
2. Using array constructor

    Let a1 = new Array(); -> constructor call

3. Using Array.of()

---------- **26/8/24** ----------

Class:

- Blueprint to create an object

- stored in script scope

- inside the class we can define properties and methods

NOTE: inside the class, if we are defining any variable/property we are not supposed to pass var, let and const keyword

- If we are going to pass any function then we are not supposed to use the function keyword

- Inside the class, inside the method if we're going to define any variable then it should be prefixed with var, let, const keyword.

Object: used to store value in key and value pair

Syntax:

```
class ClassName{
        constructor
        //statement
}
```

ClassName: identifier

Constructor:

- It Is a special type of method

- how to call constructor: new keyword

- no return statement

- inside class only one constructor is allowed

- data inside constructor belongs to local scope

- used to create the object

- there is no default constructor
- used to assign the value for key
- this refers to current creation object

Types of Methods:

1. Static Method:

- Static keyword and methods can be defined only inside the class
- if we want to access use class name

2. Non-Static Method:

Non-Static methods can be access by using object reference

---------- **28/8/24** ----------

OBJECT CREATION

1. using class and constructor

- If we have to define any properties or methods
- multiple object with same key

2. using constructor function

- If we don't want to pass any properties or methods
- reduces the syntax
- multiple object with same key

3. literal way

- when we need unique object
- Syntax: let Obj = {

    Mention key followed by colon value

    name: "abc",

    id: 80

  }

- fetch the data from object
  i. .notation → (obj.key)
  ii. Bracket notation→ obj[key]

4. Object constructor

- it uses pre-defined inbuilt class called object.
- inside that we have multiple methods.

STATIC METHODS:

1. freeze()

- if obj is in freeze state→ add, update, del is not possible
- 1 argument → object reference

2. seal()

- if obj is in seal state only update is possible
- 1 argument → object reference

3. isFrozen()

- will return Boolean value
- if obj is in freeze state it will return true
- 1 argument → object reference

4. isSealed()

- will return Boolean value
- if obj is in seal state it will return true
- 1 argument → object reference

NOTE: IF OBJ IS IN FREEZE STATE, THEN IT IS IN SEAL STATE

IF OBJ IS IN SEAL STATE, THEN IT IS NOT IN FREEZE STATE

5. keys()

- it will take obj reference as input
- it will return the output in the form of array of keys
- it will return all the keys which is present in the object reference

6. value()

- it will return the output in the form of array of value

- it will return all the keys which is present in the object reference

7. entries()

- it will return the output in the form of two-dimensional array

8. Object.assign(target, source) –>

- it is used to merge one or multiple objects and the merged value will get stored in the targeted object

- it will take two or more multiple arguments

1. Target object
2. Rest of the arguments are sourced

- and the merged value will not be stored in the sourced object

---------- **29/8/24** ----------

Looping:

- If we want to execute the same set of code again and again based on condition.

Types Of Loops:

1. **for loop**

Syntax:

```
for( initialize, condition, inc/dec) {
}
```

2. **while loop**

Syntax:

```
initialize
while(condition) {
inc/dec
}
```

3. **do while loop**

Syntax:

```
initialize
do{
inc/dec
}
while( condition);
```

4. for in loop

- it is used to iterate index of array and string.

- NOTE: OBJECTS ARE NOT ITERABLE

Syntax:

```
for(let i in arr_reference) {
        clg(i);
}
```

5. for of loop

- it is used to iterate values of array a string

Syntax:

```
for(let v of arr_reference) {
        clg(v);
}
```

SYMBOL (PRIMITIVE DATATYPE)

- Symbols are often used to add unique property keys to an object that won't collide with keys any other code might add to the object.

- it is used to pass the unique key inside the object.

---------- **2/9/24** ----------

ARRAY STATIC METHODS:

1. **Array.from()**

- used to convert array like object into array

2. **Array.of()**

- used to create new array

3. **Array.isArray()**

- used to check whether it is array or not
- it will return the Boolean value
- pass array reference as argument

ARRAY NON-STATIC METHODS:

1. **push()**

- used to append new elements at the end of the array
- will take multiple arguments
- it will affect the original array, it won't create a new array for output
- returns the new length of the array

2. **unshift()**

- used to append new elements at the beginning of the array
- it will affect the original array, it won't create a new array for output

3. **pop()**

- used to remove the array element at the end of the array

- only one array elements will be removed

- it will return the deleted array element

- it will affect the original array

4. shift()

- this method also behaves exactly like pop method but it will remove the array element at the beginning of the array

- it will return the deleted array element

- it will affect the original array

5. slice(start index, <u>end index</u>)

- used to extract the part of the array

- It will return the output in new array

- It will not affect the original array

- we can also pass -v value: it will extract from last index

6. splice() (IMPORTANT)

- used to delete and add the array element at the same time

- it will take three arguments

  - start index
  - delete count
  - array elements you want to add

- affects the original array

- returns the deleted element in the form of array

7. IndexOf()

- it will take one argument -> search element

- based on value it will return the position of the value

- it will start searching from 0 to last position (left to right)

- second argument: fromindex → from which index we have to start searching

8. lastIndexOf()

- similar to IndexOf()

- it will start searching from last position to 0 (right to left)

9. concat()

- used to merge two or more arrays

10. flat(depth) (IMPORTANT)

- used to convert multidimensional array into single dimensional array based on the depth you provided.

- it will not affect original array

- depth: default value is 1

11. includes()

- it will return Boolean value

- it will return true if element is present inside the array

- 1st argument → search element

- 2nd argument → fromindex

12. fill()

- used to replace all the array elements and the value will be what we are passing as a first argument to the fill method

- 3 argument → 1st is mandatory

  - value
  - start
  - end

- it will affect the original array

13. reverse()

- used to reverse the array

---------- **3/9/24** ----------

ARRAY HOF: (INBUILT) – NON-STATIC

1. find()

- one argument → function

- function will take 3 args

  - value of array
  - index of array
  - original array

- it will iterate the array and return the first satisfied array element value based on the passed condition

- if not satisfied it will return undefined

- pass condition inside function, within return statement

- SYNTAX:

arr = [……..]

Let fn = (v, I, arr) => {

    return condition;

}

2. findIndex()

- similar to find() method

- it will iterate the array and return the first satisfied array element index based on the passed condition

- if no array element satisfies the condition it will return -1

- return type is number

3. filter()

- it will iterate the array till the last index and return all the satisfied array element based on the passed condition in the form of array

- it will create new array for output

4. map() (IMPORTANT)

- used to perform the same operation for every array element.

- it will create new array for output

5. reduce()

- it will iterate the array and reduce all the array element value to single value

- 2 argument → function, accumulator value → optional

- function will take 2 parameters

  - accumulator value
  - current value

- it will iterate from 0 to last index

6. reduceRight()

- similar to reduce()

- it will iterate from last to 0 index

7. sort() (IMPORTANT)

- used to sort the array based on the ASCII value

- applicable for only homogenous array

- it will affect the original array

- ASCII

  - 0-9 → 48-57

- a-z → 97-122
- A-Z→65-90

- by default, sort() will sort the array in ascending order based on the ASCII value

- if we are going to sort 1 number array then we have to pass 1 function as an argument and that function will take 2 parameters and it will return number

- this compare function will take values from the array and make a comparison between two values and it returns a number

- If compare function returns positive value then b sorted before a

- If compare function returns negative value then a sorted before b

- If zero then it will not make any changes

8. some()

- I will return the Boolean value and it will return true if any one array element is satisfied with the condition.

- if all the array element is not satisfied with the condition then it will return false

9. every()

- it will return true if every array element is satisfying the condition and any one element also not satisfy with the condition it will return false

10.forEach()

- used to iterate the array

- the return type of forEach() is void.

Q. what is the difference between map() and forEach()

ANS: map() will return the new array and the return type of forEach() is void

---------- **3/9/24** ----------

STRING METHODS: NON-STATIC METHODS

1. **slice():**

- used to extract the part of the string.
- last index will not be considered.
- similar to array method
- it won't affect the original string

2. **substring():**

- same to slice
- it won't accept the negative value

3. **substr():**

- same to slice, substring
- deprecated
- 2nd arg is length

4. **includes():**

- similar to array method

5. **IndexOf():**

- similar to array method

6. **lastIndexOf():**

- similar to array method

7. **toUpperCase():**
8. **toLowerCase():**

9. **charAt():**

- 1 arg → index position

- used to return the char based on the index passed

10.charCodeAt():

- used to return the ASCII value of the particular char

11.length():

- it is the property not method.

- will return length of string object.

---------- **9/5/24** ----------

12. replace()

- used to replace the part of the string with a new value
- by default, it will replace the first matching value/string
- 1 arg→ search value, 2 arg → replace value
- it won't affect the original array

13. replaceAll()

- used to replace the part of the string with a new value
- by default, it will replace all the matching string/value

14. trim()

- used to remove whitespaces from both the end of a string
- it won't affect the original array

15. trimstart()

- remove whitespaces from start

16. trimend()

- remove whitespaces from end

17. split() (IMPORTANT): string method

- used to convert string into array
- 1 arg-> splitter (string datatype)
- if we pass space it will split into words
- if we pass empty string it will split into char

18. join(): array method

- used to convert array into string

DATE CLASS: NON-STATIC METHODS

- date is the inbuilt class in JS and this class is having one constructor so in order to create the date object we have to invoke the date constructor

- this date object will return the current date and time along with the time zone which is fetched from the system

- in date class we have non-static methods to set the date and time and to get the date and time

- used to set and get the date, time

1. getFullYear(): will return current year
2. getMonth(): will start from 0
3. getDate(): will return current date
4. getDay(): will return current day
5. getHours(): will return current hours
6. getMinutes(): will return current minutes
7. getSeconds(): will return current seconds

1. setFullYear(): used to set the year for the date obj
2. setMonth():
3. setDate():
4. setDay():
5. setHours():
6. setMinutes():
7. setSeconds():

MATH CLASS:

- inbuilt class in JavaScript

- here we don't have constructor

- all the methods are static methods, hence use class name to call the methods

- used to deal only with the numbers

1. round()
2. ceil() nearest largest value
3. floor() nearest smallest value
4. trunc() remove decimal
5. random() return random number

   Math.random() * (highest value – lowest value) + lowest value

---------- **6/9/24** ----------

REST PARAMETER

- used to store the rest of the argument value in the form of array
- rest parameter must be last parameter in the parameter list.
- …c (prefixed with 3 dots)

SPREAD OPERATOR

- Used to spread the values of datatype which is iterable (array and string)
- …str (prefixed with 3 dots)

NOTE: if 3 dots mentioned in function it is rest parameter else it will be spread operator

- both used to copy of the array/object

Destructuring

- **Array destructuring**
- **Object destructuring**

- it is process of passing unique identifier to the array elements or object elements.
- it will follow the order

Object destructuring

- identifier and key should be same
- it will not follow the order/sequence

= operator will compare the value or data in case of primitive datatype and it will non-primitive compares object address.

Why primitive datatypes are immutable and non-primitive datatypes are mutable? (IMPORTANT)

1. when we try to change the primitive dt values in the heap area the value will get changed in the different object address
2. if we try to change the non-primitive datatype values the value will get changed in the same object address
3. const keyword will prevent the values which get changed in the different object address that's why if the variable is declared with const, in case of primitive datatype it throws error and in case of non-primitive datatype the value is getting changed

---------- **12/9/24** ----------

DOM: Document Object Model

- used to manipulate the HTML elements using JS

- we can add HTML element, remove HTML element, update the content of HTML elements

- Set the attributes for the HTML elements

- we can give the CSS for the HTML elements using JS

- we can perform the events, interactive animations, etc.

- fetch the HTML Elements

- it is a tree like structure

- one webpage will have only one DOM tree

- whenever webpage is getting loaded it will create the DOM tree

Document:

- The study of document object is called DOM

- Document is the inbuilt object which is a child of window object

DOM Tree:

METHODS:

1. Document.getElementById(" ");

- 1 arg → Id value in form of string

- Based on id value it will html element in the form of object

- will target only one html element based on the id passed

- will target the first satisfied id

- id not present it will return null

2. document.getElementsByClassName();

- 1 arg → class Name

- will return in the form of HTML Collection →object → all the values will be stored in the form of index data structure (we can iterate)

- use Array.from() to convert arraylike object into array to iterate the HTML Collection

- If class is not present it will create empty HTML Collection of length 0 if present it will return HTML Collection with elements

Properties: Diff between InnerText and innerHTML

InnerText – used to pass the text content for the HTML element

InnerHTML – used to pass the text content and also, we can pass the child elements

The value should be string for both the properties

3. document.getElementByTagName(" ")

- 1 arg → tagname in the form of string

- it will return the HTML collection

4. document.querySelector() (Important)

- used to target HTML element with the help of id, class and tagname

- it will return only first html element which has the class value, id value, tagname in html document

- it will target only first element having same class/id/tagname even if multiple element has same class/id/tagname

- should pass #for id, .for class name

5. document.querySelectorAll()

- used to target HTML element with the help of id, class and tagname

- it will return all html element which has the class value, id value, tagname in html document

NODELIST

- it is an object but every data will be stored in the form of index data structure.

- It is iterable.

- to convert nodelist into array we can use Array.from()

NOTE:

The diff between html collection and node list is in html collection we can store only html element and in node list we can store html elements, text elements and other comments also.

---------- **13/9/24** ----------

Style Property:

- it is used to give the CSS for the html elements in JS

Syntax: element.Style.css_prop = “css_value”

createElement()

- used to create the new html elements based on the argument passed

- 1 arg → element_name

- it will create the html element and return it

Diff between append() and appendChild() (important)

append() is used to make multiple html elements as the child of another html element

- will take multiple arg i.e. children

Syntax: parent_element.append(child)

appendChild() is used to make only one html element as the child of another html element.

- 1 arg → child element

Syntax: appendChild(child)

ATTRIBUTE METHODS IN DOM:

1. setAttribute()

- used to set the attribute name and attribute value to html elements

- 2 arg → attribute name, attribute value

- if the attribute is having only attribute name then second attribute should be empty string

2. getAttribute()

- used to fetch the attribute value based on the argument passed

- 1 arg → attribute name

- will return the attribute value in the form of string

3. removeAttribute()

- used to remove the attribute name and attribute value based on the attribute passed

- 1 arg → attribute name

4. hasAttribute()

- will return true if the attribute is present else it will return false

- 1arg → attribute name

---------- **16/9/24** ----------

CLASSLIST PROPERTY

- it is property having the inbuilt methods which is used to add the className, remove the className and check whether the className is present or not

- **Syntax:**

1. element.classList.add():

- used to add the multiple className for the html elements

- multiple args → every args will be converted to values

- if className is already present then it will not add the className

- return type is void

2. element.classList.remove()

- used to remove the className for the html elements

- return type is void

- multiple args → every args will be converted to values

- it will remove all the className which passed as an argument

3. element.classList.contains()

- 1 arg → className

- return type is Boolean

- will return true if className is present else it will return false

4. element.classList.toggle()

- it will add the className if the className is not present and it will return true

- it will remove the className if the className is already present in the html element and it will return false

- 1arg → className

DOM EVENTS

- events are the actions which is performed on the html elements

- Example: click, mouseover, mouseout, submit, input, change, keyup

How to handle the event in the webpage

- we can handle in 2 ways

1. Using event handler property like onclick, onsubmit, oninput, etc.

- these event handler properties will take one function as value and that function is called as event handler function and it will execute when the event is triggered

- event handler function will take 1 parameter→ event object, which gives the information about the particular event

2. addEventListner()

- this method is used to handle the event

- it will take 3 args →

- Event type like click, submit, mouseover, etc.
- Event handler function
- Boolean value (optional)

---------- **18/9/24** ----------

Click event → Pointer Event

Mouse event → Mouse Event

Keyboard event → Keyboard Event

(each event type will have different event type)

Difference between keyup, keydown and keypress

Keyup event will get triggered when the user releases the key and it will accept all the keys in the keyboard

Keydown event will get triggered if we start pressing the key and it will accept all the keys in the keyboard

Keypress event will get triggered if we start pressing the key and it will accept only the characters

NOTE: both keydown and keypress will get triggered continuously if the user pressing the key continuously.

Submit Event:

- it will be applicable only for form tag

- it will get triggered when the user submits the form

- it will return the event object as submit event only

e.preventDefault()

-this method prevents from the default behavior of the html elements

Value property

- the value property is used to fetch the values from the input tag and textarea tag

---------- **19/9/24** ----------

Form Data:

- form data is used to fetch all the data from the form

- it is the inbuilt class in JS

- to fetch data, first we have to create FormDataObject

- Syntax:

let data = new FormData();

- this formdata constructor will take one argument → form reference

- to convert formdata object into JS object we can use obj.fromEntries() and it will take 1arg→ formdata reference

CHECKED PROPERTY:

- this property will return true if the option is selected otherwise it will return false

- this property will be used in checkbox, radio button, dropdowns in the form to fetch the value

getAll():

- it is a non-static method and it is used to fetch multiple data from checkbox and it will take 1 arg →name attribute value

---------- **20/9/24** ----------

EVENT PROPOGATION:

Event propagation is a concept that describes how an event propagates through a DOM tree to achieve its target and what happens after that

- there are 3 phases in event propagation

1. **Capturing phase**

- the event is in capturing phase if the 3rd arg is true in addEventListner method

- it will execute before the target phase

2. Target phase

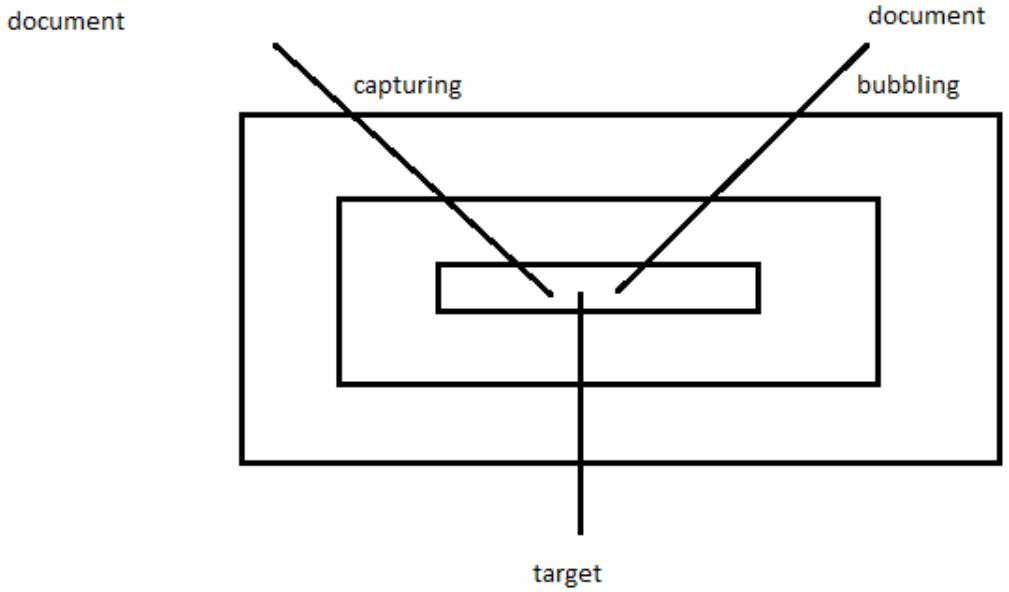
- the event is in target phase when you perform the action especially for that html element

- the target phase will execute if the 3rd arg if true or false in addEventListner method

3. Bubbling phase

- the event is in bubbling phase if the 3rd arg is false in addEventListner method or if we are not passing any argument also

- bubbling phase will execute after target phase



event.eventPhase

- this property will return 1 if the event is in capturing phase

- this property will return 2 if the event is in target phase

- this property will return 3 if the event is in Bubbling phase

e.stopPropagation()

- it will stop the propagation once the event achieve its target and it will not propagate through the next elements in DOM tree

e.stopImmediatePropagation():

- it also behaves exactly like stop propagation but also it will prevent from executing the other events for the same elements

call(), apply(), bind()

- all the methods are used to set the value for this keyword inside the function (named function)

Set the value for this, and also it will execute the fn

Fn_name.call(Object_ref, …)

Fn_name.apply(Object_ref, [ ])

Set the value and returns bound function

Fn_name.bind(Object_ref)

Call and apply method will set the value for this keyword and also execute the function but bind method will only set the value for his keyword and it will return bound function

- if you want to execute the function you should call the bound function

- the difference between call and apply function is call method will take multiple args and apply method will take only two args and 2nd arg is argument array

---------- **23/9/24** ----------

This keyword:

- globally → outside function and outside block→ window object

- inside function (except arrow function) →window object (not using use strict)

- inside function → undefined (using use strict)

- inside arrow function → parent scope this keyword value

- inside constructor → current creating object

- inside the object and inside function (except arrow function) → current object

- inside the addEventListener() (if second arg is anonymous function) → targeted element

LOCAL STORAGE AND SESSION STORAGE:

- Storage option in browser

- Browser specification and JS used to access it

- this properties both are present inside the window object

- will return storage object

- size: 2mb to 10mb data

- local storage will save data permanently and session storage will save data for a particular session

---------- **24/9/24** ----------

HOW TO MAKE COPY OF AN OBJECT?

SHALLOW COPY (IMPORTANT)

- if you make the copy of an object and if we make any changes in the original object then it should affect the copied object this type of copying is called shallow copying.

- we can achieve shallow copy of an object in 4 ways:

1. By copying obj ref
2. Using Object assign()
3. Spread operator
4. For in loop

DEEP COPY (IMPORTANT)

- if we make any changes in the original object it will not affect the copied object this type of copying an object is called as deep copy

- we can achieve deep copy of an object in two ways

1. Using StructuralClone()
2. Using JSON.Stringify() and JSON.parse()

---------- **25/9/24** ----------

PROMISE

- it is used to perform the asynchronous operation in JS

- promise have 3 states

1. Resolve/fulfilled
2. Rejected
3. Pending

- we can create the promise object by invoking the promise constructor

- It will take 1 arg → callback function that call function will take 2 parameters

→1. Resolve function

→ 2. Rejected function

- within this callback function we have to define the logic for resolved and rejected state

- if you call the resolved function then the promise is in resolve state and if you call the rejected function then the promise is in rejected state

- if you are not calling both resolved and rejected function then the promise is in pending state

- this resolve n reject function will take one argument that will be the promise result

HOW TO HANDLE THE PROMISE

- we can handle the promise using 3 promise instance methods

1. then()
2. catch()
3. finally()

- all the three functions will take 1 callback function as an argument and all the 3 function will return the same promise

- Then() callback function will execute if the promise is resolved and it will take 1 parameter i.e. promise result

- Catch() callback function will execute if the promise is rejected and it will take 1 parameter i.e. promise result

- if the promise is in pending state all the 3 methods will not get executed

---------- **27/9/24** ----------

window.fetch()

- it is used to fetch the data from the API and JSON file

- fetch method will take one argument i.e. API url or JSON file path

- fetch method will return promise so we should the promise and the promise result will be response object

.JSON()

- this method is used to convert the response object into data while converting it will convert json object into JavaScript object

- this method also will return promise and the actual data will be present in the promise result

Async function

- async function is used to handle the asynchronous operation in JavaScript

- it will return promise with resolved state if the function executes completely and it will return the promise with rejected state if there is any error

- we can handle the async function in 2 ways

1. using try catch block

2. using then catch block

Await operator

- it should be prefixed with the promise and it will wait for the promise to resolve or reject then the rest of the code will get execute before that it will execute all the synchronous code

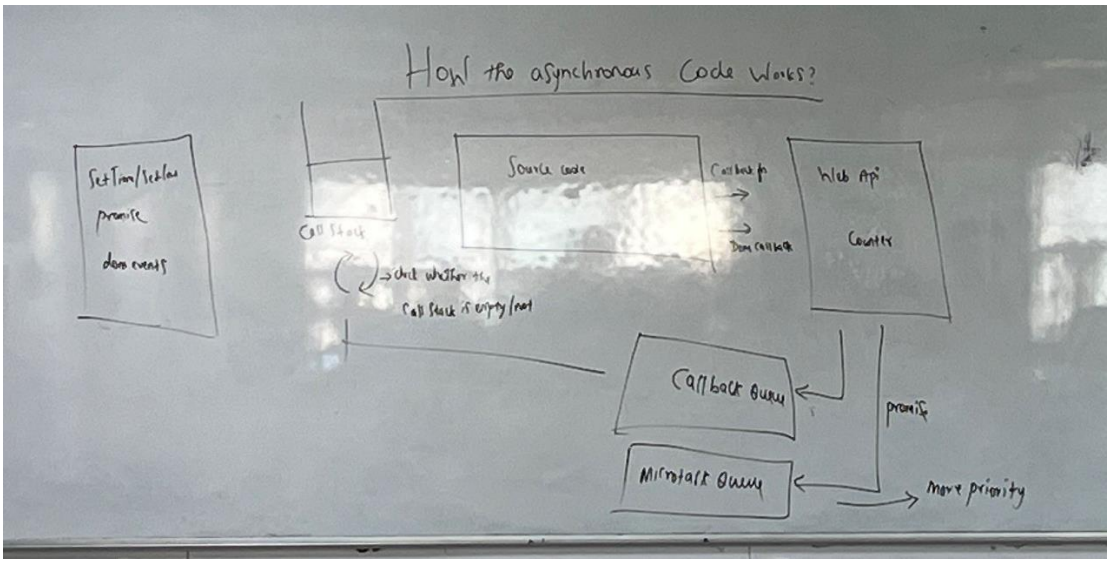
---------- **30/9/24** ----------

Modules:

1. CommonJs →server
2. ES6 →browser

---------- **01/10/24** ----------

HOW THE ASYNCHRONOUS CODE WORKS



Web API:

- web API are the collections of browsers provided interfaces that allow JS code to interact with the browser environment and perform the task

- example of web API's in JS engine or DOM, fetch API, Timer API and local storage and session storage

Callback Queue

- it is a queue of task that are executed after the current task.

- the callback queue is handled by the JS engine after it has executed the task in the microtask queue

microtask queue

- microtask queue is like a callback queue but it has more priority

- all the callback functions coming through promises will go inside the microtask queue

- example: fetch method

Event Loop

- the event loop is continuous process that continuously checks the state of the callback queue while the call stack is empty

- if the call stack is empty the event loop takes the first task (callback) from the callback queue or microtask queue and push into the call stack

- then the callback function is getting executed in the order it was added to the queue

Modules in JS

- modules are used to send the data from one JS file to another JS file

- there are 2 types of modules

1. Common JS

- it will work only in the server side

- it will not work in the browser side

- in common JS module to export the data we should use the property module.exports and the value should be in the form of object

- to import the data, we should use require method and it will take 1 arg → JS file path

2. ES6 Module

- it will work only in the browser side or client side and it will not work in the server side

- to perform ES6 module the JS file extension should be .mjs and in the script tag we should pass attribute “type = module”

- in ES6 module we have 2 types

1. default export

- import statement should be same in both the cases

- import identifier from “App.mjs” → default export

- in default export we can export only one data and in named export we can export multiple data

2. named export

import { identifiers } from “App.mjs”

Prototype:

- every object is having 1 property i.e. prototype which means the parent object

- we can set one object as a parent of another object by using the property “__proto__”

Prototype Inheritance:

- accessing the properties and methods from the parent object using child object reference

Set Object:

- set object is used to store the unique values or data and duplicate values are not allowed in set object

- to create the set object, we should invoke the set constructor

- it will take 1 arg → array, and within this array we can pass the values which has to be stored inside the set object

- w can also add the values using add() and delete the values using delete() and clear all the values using clear()