**------------------------------------ 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

1. **document.write() / document.writeln()**

-to print the details in webpage

1. **window.alert();**

-return type void

1. **window.confirm();**

-returns boolean value

1. **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

}

1. **Named function**

- the function with function name

1. **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 ------------------------------------**

1. **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; |

1. **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

1. .notation 🡪 ( obj.key )
2. 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) {

}

1. **while loop**

Syntax:

initialize

while(condition) {

inc/dec

}

1. **do while loop**

Syntax:

initialize

do{

inc/dec

}

while( condition);

1. **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);

}

1. **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