SAT Javascript Test 7 Feb 2026.

1) Class & Constructor :

* class → blueprint / template to create objects

*Constructor → special method that runs automatically when object is created.

Example :

```
class Student {
    constructor (name, age) {
        this.name = name;
        this.age = age;
    }
}

let S1 = new Student("kumar", 21);

console.log(S1.name);
```

output :

kumar

2) Explain push, pop, shift, unshift, delete with example:

* These are array methods used to add or remove elements.

* push() → add elements at the end.
* pop() → removes last element
* unshift() → add element at the beginning
* Shift() → removes first element
* delete() → removes value but index stays empty

Example:

```
let arr = [10, 20, 30];

arr.push(40);
Console.log(arr);  Output : [10,20,30,40]

arr.pop();
Console.log(arr);  output [10,20,30]

arr.unshift(5);
console.log(arr);  output [5,10,20,30]

arr.shift();
Console.log(arr);  output [10,20,30]

delte arr[1];
Console.log(arr);  output [10, empty, 30]
```

3) Encapsulation :

* Encapsulation means hiding data and controlling data access using methods.

* user can't directly change the value - they must use functions :

Example :

```
class Bankaccount {
    constructor (balance) {
        this.balance = balance;
    }
    deposit (amount) {
        this.balance += amount;
    }
    withdraw (amount) {
        if (amount <= this.balance) {
            this.balance -= amount;
        } else {
            console.log ("Not enough balance");
        }
    }
    getBalance () {
        return this.balance;
    }
}
```

```
let acc = new BankAccount (1000);
acc.deposit(500);
acc.withdraw(200);
Console.log(acc.getBalance());
```

Output :

1300

A) Explain Inheritance in Javascript with example.

* Inheritance means one class can use properties and methods of another class.

Example :

```
class Car {
    constructor(brand) {
        this.brand = brand;
    }
    start() {
        console.log(this.brand + " car started");
    }
}

class ElectriCar extends Car {
    constructor(brand, battery) {
        super(brand);
        this.battery = battery;
    }
    charge() {
        console.log("Battery charging" + this.battery + "%");
    }
}
```

```
let obj = new ElectricCar("Tesla", 80);

obj.start();
obj.charge();
```

output:

Tesla car started

Battery charging: 80%

5) Explain overloading in Javascript with Example:.

* Overloading means using the same function name but doing different work based on inputs.

Example:.

```
function add(a,b,c){
    if(a!==undefined && b!==undefined && c!==undefined){
    console.log("Adding three numbers", a+b+c);
}
else if(a!==undefined && b!==undefined){
    console.log("Adding two numbers", a+b);
}
else{
    console.log("provide atleast two numbers");
}
}
add(10,20);
add(5,10,15);
add(7);
```

output:

Adding two numbers: 30

Adding three numbers: 30

Provide at least two numbers

6) Explain Overriding in Javascript with an example.

* child class re-defines a method that already exists in parent class.

Example:-

```
class Animal {
    Speak(){
    console.log("Animal makes a sound");
    }
}
class Dog extends Animal {
    Speak() {
        console.log("Dog Barks");
    }
}
let obj = new Dog();
Obj.Speak();
```

output :

Dog barks

7) Explain Static method in Javascript with an Example:-

* Static method means, Method belongs to the class itself, not to the object. So we call it using class name, not using object

Example:-

```
Class MathUtils {
    Static add (a,b) {
        console.log (a+b);
    }
}
MathUtils.add (5,3);
```

Output

8.

8) What is callback function Explain with van example:-

* Callback means a function that is passed as van arguement to another function, and it will run later after some work is finished.

Example:-

```
function greet (name, callback){
    console.log ("Hello" + name);
    call back();
}
function sayBye (){
    console.log ("Bye!");
}
```

```
greet ("selva", Say Bye);
```

output:

Hello Selva

Bye.

9) Promises in Javascript:

* promises means. It is used to handle future results in JS.

Promises has 3 states:

* Pending → work is still running
* Resolved → work completed successfully
* Rejected → work failed

Example:

```
let mypromise = new promise (function (resolve, reject) {
    let workDone = true;
    if (workDone){
        resolve ("Task completed successfully");
    }
    else{
        reject (" Task Failed");
    }
});
mypromise
    .then ( function (result){
        console.log (result);
    })
```

```
.catch (function (error){
    console.log (error);
});
```

Output:

Task completed successfully.

10) Explain Async / Await with an example:.

* Async | Await is used to handle promises in an easier way.

* Instead of using .then() and .catch(), we write code like normal step by step execution.

* async → makes a function return a Promise.

* await → waits until the promise gives result.

Example:

```
function getData(){
    return new promise (function (resolve){
        setTimeout (function (){
            resolve ("Data received");
        }, 2000);
});
```

```
async function showData (){
    console.log ("Loading ....");
    let result = await . getData();
    console. log (result);
}
showData();
```

Output:

Loading ....

after 2 seconds

Data received.