Assignment Questions 4

Q.1 Explain Hoisting in JavaScript?

Ans -

Hoisting in JavaScript is a behavior in which a function or a variable can be used before declaration.

For Example:-

```
console.log(name)  // undefined
var name;
```

But in case of **let** and **const** it behaves something different:

For Example:-

```
console.log(name)  // ReferenceError
let name;
console.log(name)  // ReferenceError
const name;
```

It also works with the function:

For Example:-

```
sayHello();
function sayHello(){
   console.log("Hello")
}
```

Output: Hello

Here we have used a function before its declaration.

If we declare an arrow function or a function expression then this will not work.

```
sayHello();
const sayHello = () => {
   console.log("Hello")
}
```

Output: Error

This code will throw an error.

Q.2 Explain Temporal Dead Zone?

Ans -

Temporal Dead Zone (TDZ) refers to a specific behavior related to variables declared with **let** and **const** keywords. The Temporal Dead Zone is a time span in which variables exist but are not yet initialized and cannot be accessed without causing a runtime error.

E.g-

```
console.log(x); // ReferenceError: x is not defined
let x = 10;
```

Q.3 Difference between var & let?

Ans -

Var -

- It has global scope
- It can be declared globally and accessed globally
- The variable declared with var can be re-declared and re-assignment.

For Example:-

```
var a=10;
a=15;
var a = 20;
console.log(a); // 20
```

- It is hoisted.

For Example:-

```
console.log(a); // undefined
var a = 10;
```

Let -

- It has block scope
- It can be declared globally but cannot access globally
- The variable declared with let can't be re-declared.

For Example:-

```
let a = 10;
let a = 20;  // ReferenceError
```

- It is not hoisted.

For Example:-

```
console.log(a); // ReferenceError
let a = 10;
```

Q.4 What are the major features introduced in ECMAScript 6?

Ans -

The following features were introduced in ECMAScript 6:-

- let and const keyword
- arrow function
- spread and rest operator
- Classes
- Promises
- for-of loop
- Template literals

Q.5 What is the difference between let and const ?

Ans -

Let -

- It can be updated.
- It is block scoped.
- It is not hoisted at the top.
- It can be declared without initialization.

Const -

- It cannot be updated.
- It is block scoped.
- It is not hoisted at the top.
- It needs to be declared and initialized at the same time.

Q.6 What are template literals in ES6 and how do you use them?

Ans -

The template literals are the Backticks (` `) in ES6.
We use it instead of double (“ “) or single quote (‘ ‘). template literals makes the code more readable.

E.g-

```
const name = “Laksh”;
const age = 19;
console.log(`My name is ${name}. I am ${age} years old.`)
// My name is Laksh. I am 19 years old.
```

Q.7 What's the difference between map & forEach?

Ans -

The difference are following:-

i) Returning Value: forEach() returns undefined but the map() returns a new array

E.g-

```
const arr = [1, 2, 3, 4, 5];
console.log(arr.forEach(el => el * el));  // undefined
console.log(arr.map(el => el * el));  //  [1, 4, 9, 16, 25]
```

ii) Ability to Chain other methods: map() function can chain with other methods like filter(), sort(), reduce() but forEach() doesn't have this functionality.

Q.8 How can you destructure objects and arrays in ES6?

Ans -

Destructuring Arrays -

To destructure an array, you use square brackets [ ] on the left side of the assignment, matching the structure of the array.

E.g -

```
const numbers = [1, 2, 3];
const [a, b, c] = numbers;
console.log(a); // 1
console.log(b); // 2
console.log(c); // 3
```

Destructuring Objects -

To destructure an object, you use curly braces { } on the left side of the assignment, matching the property names of the object.

E.g-

```
const person = {
  name: 'Ram',
  age: 30,
  city: 'Ranchi'
};
const { name, age, city } = person;
console.log(name); // Ram
console.log(age); // 30
console.log(city); // Ranchi
```

Q.9 How can you define default parameter values in ES6 functions?

Ans -

In ES6 (ECMAScript 2015) and later versions of JavaScript, wecan define default parameter values for function parameters. Default parameter values allow us to specify a default value that will be used if no argument or an undefined argument is passed to the function.

E.g-

```
function greet(name = 'Lakshman') {
  console.log(`Hello, ${name}!`);
}
greet();   // Hello, Lakshman!
greet('Ram');   // Hello, Ram!
```

Q.10 What is the purpose of the spread operator (. . .) in ES6?

Ans -

The spread operator spreads all the elements and combines them.

E.g-

```
const arr1 = [1, 2, 3];
const arr2 = [4, 5, 6];
const arr3 = [7, 8, 9];
const arr = [...arr1, ...arr2, ...arr3]; // [1, 2, 3, 4, 5, 6, 7, 8, 9]
```