Assignment Questions

1. Hoisting is a JavaScript Behavior where variable and function declarations are moved to the top of their containing scope during the compilation phase. This means that you can use variables and functions before they are actually declared in the code. However, only the declarations are hoisted, not the initializations or assignments.
2. The Temporal Dead Zone (TDZ) is a behavior in JavaScript that occurs when trying to access a variable before it has been initialized. When a variable is declared using ‘let’ or ‘const’, it is hoisted to the top of its block scope, but remains in an undefined state until the actual declaration statement is encountered. If you try to access the variable within the TDZ, a reference error is encountered.
3. The main difference between **var** and **let** in JavaScript is in their range of Scope. Variables declared through **var** are function scoped or globally accessible, which implies that they can be accessed throughout the entire function or global scope. Whereas in the variables declared via **let** are block-scoped, meaning that they are only accessible within the nearest enclosing block. Along with this there is another difference and that is **var** is hoisted whereas **let** is not.
4. ECMAScript 6 also commonly known as ES6 was introduced in 2015 and some of its ground breaking features include block scoped variables which are **let** and **var,** *arrow functions*, *classes, modules, templates, literals, destructuring assignments, spread operators and enhanced object literals.*
5. The major difference between **let** and **const** is in their reassignment behaviour. That simply means that any value that is assigned to a variable declared by **let** can be reassigned. But, in case of **const** if something is given some certain value during initialisation cannot be reassigned with some other value. But, It is also important to note that **const** variables are not immutable, which means that their properties can still be modified if they are object or arrays.
6. Template literals , also known as template strings *are a feature in ES6* that allows for more flexible string formatting. They are enclosed within backticks(`) instead of single or double quotes. **They also support multiline strings, interpolation of variables using “***${}***”** and the ability to include expressions or function calls directly within the string.

Example:

const name = ‘Anurag’;

const age = 21;

const greeting = `Hello, my name is ${name} and I am ${age} years old.`

console.log(greeting);

1. Both ‘map’and ‘forEach’ are array methods/functions in JavaScript. The main difference between them is that **map** creates a new array with the results of calling a provided function on every element, while **forEach** executes a provided function for each array element, but does not return a new array.

Example usage of **map:**

const numbers = [1,2,3,4];

const doubledNumbers = numbers.map(num=> num\*2);

console.log(doubledNumbers);

Example usage of **forEach:**

const numbers = [1,2,3,4];

numbers.forEach(num=> console.log(num));

1. Destructuring assignment is a feature of ES6 that allows you to extract values from objects into distinct variables.

Example usage with Objects:

const person = {name: ‘John’, age:25, city: ‘New York’};

const {name, age} = person;

console.log(name, age); // John 25

Example usage with arrays:

const numbers = [1,2,3];

const[first, second] = numbers;

console.log(first, second); // 1 2

1. ES6 introduced the ability to set default values for function parameters. If a parameter is not passed or is explicitly set as ‘**undefined**’**,**  the default value will be used.

Example:

Function greeting(name = ‘Anonymous’){

console.log(`Hello, ${name}!`);

}

greeting();

greeting(‘John’);

1. The purpose of the spread operator (…) is to expand elements of an array or properties of an object. It allows you to create copies or merge arrays/objects easily.

We can even say it works similar to concatenation but for arrays and objects in JavaScript.

Example usage with arrays:

const arr1 = [1,2,3];

const arr2 = […arr1,4,5,6];

console.log(arr2); // [1,2,3,4,5,6]

Example usage with objects:

const obj1 = {x: 1, y: 2};

const obj2 = {…obj1, z: 3};

console.log(obj2); // {x: 1, y: 2, z: 3}

The spread operator is versatile and can be used in various scenarios, such as function arguments, array manipulation, Object Cloning and much more.