Assignment Questions 4

💡 **Q.1** Explain Hoisting in JavaScript

Ans.: Hoisting is a JavaScript concept that moves all variable and function declarations to the top of their scope before code execution. This means that you can use a variable or function before it is actually declared, as long as the declaration is within the same scope.  
For Example :

var run = “running”;

console.log(run);

function run() {

console.log(“Running”);

}

Here, the JavaScript interpreter hoists the declaration of ‘run’ to the top of the scope, before the console.log() statement. This means that the console.log() statement can see the declaration of run, even though it is not actually declared yet.

💡 **Q.2** Explain Temporal Dead Zone?

Ans.: A temporal dead zone (TDZ) is the area of a block where a variable is inaccessible until the moment the computer completely initializes it with a value. Suppose we attempt to access a variable before its complete initialization. In such a case, JavaScript will throw a ReferenceError. A block’s temporal dead zone starts at the beginning of the block’s local scope. It ends when the computer fully initializes your variable with a value. Once we initialize the variable, the TDZ doesn’t exist.

💡 **Q.3** Difference between var & let?

Ans.: The keywords let and var both declare new variables in JavaScript. The difference between let and var is in the scope of the variables they create:

Note : Variables declared by let are only available inside the block where they’re defined.

Note : Variables declared by var are available throughout the function in which they’re declared.

Following example describes the difference in best way :  
function varScoping() {

var x = 1;

if (true) {

var x = 2;

console.log(x); // will print 2

}

console.log(x); // will print 2

}

function letScoping() {

let x = 1;

if (true) {

let x = 2;

console.log(x); // will print 2

}

console.log(x); // will print 1

}

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

Ans.: ES6 comes with significant changes to the JavaScript language. It brought several new features like, let and const keyword, rest and spread operators, template literals, classes, modules and many other enhancements to make JavaScript programming easier and more fun.

let and const Keywords

Arrow Functions

Multi-line Strings

Default Parameters

Template Literals

Destructuring Assignment

Enhanced Object Literals

Promises

Classes

Modules

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

Ans.: As we know the both let and const variable uses block scope. Also once we declare the let variable, we can update it but we can’t re-declare into scope. Whereas, const cannot be updated or re-declared into the scope. The let variables can be declared without initializing it, but the const variable requires to be initialized when declared. When the let variable is accessed without initializing, it gives reference error.

💡 **Q.6** What is template literals in ES6 and how do you use them?

Ans.: Template literals are a new feature introduced in ECMAScript 6. It provides an easy way to create multiline strings and perform string interpolation. Template literals are the string literals and allow embedded expressions. Template literals can contain placeholders, which are indicated by the dollar sign and curly braces ($(expression}). Inside the backticks, if we want to use an expression, then we can place that expression in the ($(expression}).

Syntax for template literals is,

var str = `string value`;

💡 **Q.7** What’s difference between map & forEach?

Ans.: map and forEach both are methods used for operations on array in javascript. The map() method returns a new array, while the forEach() method does not return anything. This means that you can use the map() method to create a new array with the results of the iteration, while the forEach() method is only used to iterate through the array and perform some action on each element.

The map() method takes a callback function as an argument, while the forEach() method takes a function as an argument. The callback function in the map() method is used to transform each element of the array, while the function in the forEach() method is used to perform some action on each element of the array.

The map() method is chainable, while the forEach() method is not. This means that you can use the map() method to chain other methods together, while the forEach() method cannot be chained.

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

Ans.: Arrays or Objects can be destructured using the 2 operators as

1. Rest operator

In JavaScript, the rest syntax, also known as the rest parameter, allows you to represent an indefinite number of arguments as an array. It provides a convenient way to work with variadic functions or handle multiple function parameters without explicitly defining them.The rest parameter is denoted by three dots (...) followed by a parameter name. It must be the last parameter in a function declaration or function expression.

1. Spread operator

In JavaScript, the spread operator, denoted by three dots (...), is a powerful feature that allows you to expand iterable objects into multiple elements. It is primarily used in array literals, function calls, and object literals to provide a concise way of working with arrays and other iterable objects.

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

Ans.: Default parameter values can be defined as follows,  
function greet(time = “day”) {

console.log(`Good, ${time}!`);

}

greet(‘Morning!'); // Output: Good, Morning!

greet(); // Output: Good, day!

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

Ans.: Spread operator

In JavaScript, the spread operator, denoted by three dots (...), is a powerful feature that allows you to expand iterable objects into multiple elements. It is primarily used in array literals, function calls, and object literals to provide a concise way of working with arrays and other iterable objects. The spread operator is a powerful and versatile feature in JavaScript that simplifies various tasks involving arrays, strings, and objects. It provides a concise syntax for expanding and manipulating data, making your code more expressive and readable.