WEEK 7

9.REACTJS-HOL

1. **List the features of ES6**

* let and const for block-scoped variables
* Arrow functions (=>)
* Classes and inheritance
* Template literals (`Hello ${name}`)
* Default parameters
* Destructuring (arrays & objects)
* Rest and Spread operators (...)
* Promises for async operations
* Map and Set collections
* Modules (import / export)
* Enhanced object literals
* for...of loop
* Symbol data type

2. **Explain JavaScript let**

* let is used to declare **block-scoped** variables.
* It avoids issues with var, which is **function-scoped**.
* Variables declared with let can be updated but **not re-declared** within the same scope.

3. **Identify the differences between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| Scope | Function-scoped | Block-scoped |
| Redeclaration | Allowed | Not allowed in the same scope |
| Hoisting | Hoisted (initialized as undefined) | Hoisted (not initialized) |
| Temporal Dead Zone | No | Yes (cannot access before declaration) |

4. **Explain JavaScript const**

* Used to declare block-scoped constants.
* The value cannot be reassigned.
* However, for objects/arrays, the contents can be modified.

CODE:

const x = 10;

const arr = [1, 2];

arr.push(3);

**5. Syntax for creating objects using class keyword.**

* Supports constructor methods, instance methods, and static methods.

CODE:

class Person {

constructor(name) {

this.name = name;

}

greet() {

return `Hello, ${this.name}`;

}}

6. **Explain ES6 class inheritance**

* Achieved using extends and super() keywords.
* Allows child classes to inherit properties and methods from parent classes.

CODE:

class Employee extends Person {

constructor(name, role) {

super(name);

this.role = role;

}

getRole() {

return this.role;

}}

7. **Define ES6 arrow functions**

* Shorter syntax for writing functions.
* **No own this, arguments, or super** (lexically bound).

const add = (a, b) => a + b;

8. **Identify set(), map()**

**Set**

* A collection of **unique** values.

const mySet = new Set([1, 2, 2, 3]);

**Map**

* A collection of key-value pairs with keys of **any type**.

const myMap = new Map();

myMap.set('name', 'Alice');

myMap.get('name');