**9.ReactJS-HOL**

**1. List the features of ES6**

* Arrow Functions
* Classes
* Template Literals
* Default Parameters
* Destructuring Assignment
* Rest and Spread Operators
* let and const keywords
* Promises
* Modules (import/export)
* Enhanced Object Literals
* Iterators and Generators
* for...of loop
* Map and Set data structures

**2. Explain JavaScript let**  
let is a block-scoped variable declaration introduced in ES6. Unlike var, which is function-scoped, let allows variables to be limited to the block in which they are declared. It helps prevent accidental overwriting of variables and supports the concept of temporal dead zone (TDZ), meaning the variable cannot be accessed before its declaration.

**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 (no initialization, TDZ) |
| Global Object Binding | Yes (if declared globally) | No |

**4. Explain JavaScript const**  
const is used to declare constants. Variables declared with const cannot be reassigned and must be initialized during declaration. It is block-scoped like let. However, const does not make objects immutable—object properties can still be modified.

**5. Explain ES6 class fundamentals**  
ES6 introduces a new syntax for creating objects and dealing with inheritance: class. It is essentially syntactic sugar over JavaScript's existing prototype-based inheritance.

class Person {

constructor(name) {

this.name = name;

}

greet() {

console.log(`Hello, my name is ${this.name}`);

}

}

**6. Explain ES6 class inheritance**  
Classes can inherit from other classes using the extends keyword. The super() function is used to call the constructor of the parent class.

class Employee extends Person {

constructor(name, id) {

super(name); // calls Person constructor

this.id = id;

}

showId() {

console.log(`ID: ${this.id}`);

}

}

**7. Define ES6 arrow functions**  
Arrow functions provide a shorter syntax for writing functions. They also do not have their own this, arguments, super, or new.target.

// Regular function

function add(a, b) {

return a + b;

}

// Arrow function

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

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

**Set**: A collection of unique values.

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

**Map**: A collection of key-value pairs where keys can be of any type.

const myMap = new Map();

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

myMap.set(1, 'One');

console.log(myMap.get('name')); // 'John'