

WEEK 7 –

9.REACT-JS-HOL –

1. Features of ES6 (ECMAScript 2015)

ES6 introduced major improvements to JavaScript. Key features include:

- let and const (block-scoped variables)
- Arrow functions (=>)
- Template literals
- Classes and inheritance
- Default and rest parameters
- Destructuring (arrays & objects)
- Modules (import and export)
- Promises (for asynchronous code)
- Map and Set data structures
- Spread (...) and rest (...) operators

2. let in JavaScript

- Declares block-scoped variables.
- Can be updated but not re-declared in the same block.
- Unlike var, it avoids hoisting-related bugs.

```
let x = 10; x = 20; // valid let x = 30; //
```

Error: Already declared

3. Difference Between var and let

Feature	var	let
Scope	Function-scoped	Block-scoped ({})
Hoisting	Hoisted and initialized	Hoisted but not initialized
Re-declaration Allowed		Not allowed in same block
Use in loops	Shared across iterations	Unique per iteration

4. const in JavaScript

- Also block-scoped like let.
- Must be initialized at the time of declaration.
- Cannot be reassigned, but mutable objects like arrays can still be modified.

```
const pi = 3.14;
// pi = 3.15; Error
```

```
const arr = [1, 2]; arr.push(3);
// Valid
```

5. ES6 Class Fundamentals

Introduces object-oriented class syntax.

javascript CopyEdit

```
class Person {
  constructor(name) {
    this.name = name;
  }

  greet() { return `Hello,
  ${this.name}`;
  }
}
```

```
const p = new Person("Harshi"); console.log(p.greet());
```

6. ES6 Class Inheritance

One class can inherit from another using extends.

```
class Animal {
  constructor(name) { this.name
= name;
  }
```

```
    speak() { return `${this.name} makes a  
sound`;  
  }  
}
```

```
class Dog extends Animal {  
  speak() { return `${this.name}  
barks`;  
  }  
}
```

```
const dog = new Dog("Bruno"); console.log(dog.speak()); //  
Bruno barks
```

7. Arrow Functions in ES6

- Compact function syntax.
- Lexically binds this.
- Cannot be used as constructors.

// Traditional function

```
add(a, b) { return a +  
b;  
}
```

// Arrow const add = (a, b)

=> a + b; Set() and Map()

Set

- Stores unique values of any type.
- No duplicate entries allowed. const numbers = new Set([1, 2, 2, 3]);

```
console.log(numbers); // Set { 1, 2, 3 }
```

Map

- Key-value pairs.
- Keys can be of any type (object, function, etc.).

```
const capitals = new Map(); capitals.set('India',  
'New Delhi'); capitals.set('Germany', 'Berlin');  
console.log(capitals.get('India')); // New Delhi
```

