React

1. Features of ES6 (ECMAScript 2015)

ECMAScript 6 (commonly known as ES6) introduced a wide range of features aimed at improving the language's functionality, readability, and performance.

Key features include:

1. **let and const** – Block-scoped variable declarations.
2. **Arrow Functions (=>)** – Concise syntax for writing functions with lexical this.
3. **Classes** – Syntax for defining object-oriented classes.
4. **Template Literals** – Use of backticks (`\) to embed expressions in strings.
5. **Destructuring Assignment** – Extract values from arrays or properties from objects into distinct variables.
6. **Default Parameters** – Set default values for function parameters.
7. **Rest and Spread Operators (...)** – Gather or spread array elements or object properties.
8. **Promises** – Native support for asynchronous programming.
9. **Modules (import/export)** – Allow code to be modularized across files.
10. **Enhanced Object Literals** – Shorthand for defining object properties and methods.
11. **Symbol Type** – New primitive data type for unique identifiers.
12. **Iterators and Generators** – Define custom iteration behavior for objects.

---

2. JavaScript let

The let keyword is used to declare variables that are **block-scoped**.

Key points:

- Scope: Limited to the block ({}) in which it's defined.

- Cannot be redeclared in the same scope.
- Can be updated, but not redeclared.
- Does not attach to the window object (in browsers).

Example:

```javascript
let x = 10;
if (true) {
  let x = 20;
  console.log(x); // 20 (inside block)
}
console.log(x); // 10 (outside block)
```

---

3. Differences Between var and let

Feature	`var`	`let`
Scope	Function-scoped	Block-scoped
Hoisting	Hoisted and initialized as `undefined`	Hoisted but not initialized
Redeclaration	Allowed in the same scope	Not allowed in the same scope
Global Object	Becomes a property of `window`	Does not attach to `window`

Example:

```javascript
var a = 1;
var a = 2; // Valid

let b = 1;
// let b = 2; // SyntaxError: Identifier 'b' has already been declared
```

---

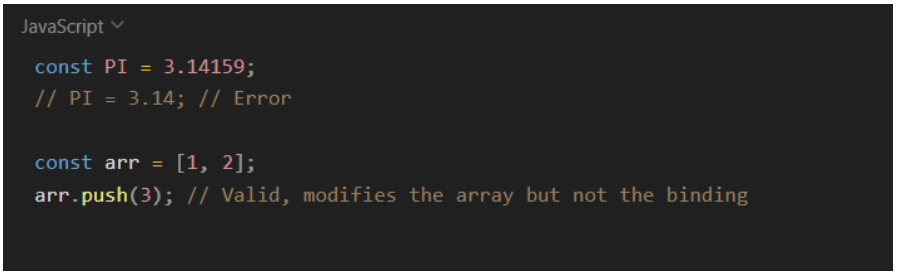
4. JavaScript const

The const keyword is used to declare **block-scoped, read-only constants**.

Key points:

- Must be **initialized at the time of declaration**.
- Cannot be **reassigned**.
- The **value itself is not immutable**—objects or arrays declared with const can be modified.

Example:



---

5. ES6 Class Fundamentals

ES6 introduces a simpler, cleaner syntax to create classes and handle inheritance.

Basic class syntax:

Key points:

1. Use class and constructor.
2. Methods are defined without function keyword.

3. Class declarations are not hoisted.
4. Classes can be instantiated using new.

---

6. ES6 Class Inheritance

ES6 supports inheritance through the extends and super keywords.

Example:

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

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

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

Key points:

- **extends** sets up prototype chain.
- super() calls the constructor or method of the parent class.
- Child classes inherit both properties and methods.

---

7. ES6 Arrow Functions

Arrow functions provide a concise syntax for writing functions.

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

Key characteristics:

1. Shorter syntax.
2. Do **not bind their own this**—they inherit from the enclosing scope.
3. Cannot be used as constructors.
4. Cannot use arguments object.

Examples:

```javascript
const square = x => x * x;
const greet = () => 'Hello';
```

---

8. Set and Map in ES6

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

Key characteristics:

1. Duplicate values are automatically removed.
2. Maintains insertion order.
3. Provides methods like .add(), .has(), .delete(), and .clear().

Example:

```javascript
const numbers = new Set([1, 2, 3, 3]);
console.log(numbers); // Set {1, 2, 3}
```

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

Key characteristics:

1. Maintains insertion order.
2. Keys are not limited to strings (unlike objects).
3. Provides .set(), .get(), .has(), .delete(), .clear().

Example: