**1. List the Features of ES6**

ES6 (also known as ECMAScript 2015) introduced many new features to JavaScript, including:

* let and const (block-scoped variables)
* Arrow functions (=>)
* Classes and Inheritance
* Template literals (`Hello ${name}`)
* Default parameters
* Destructuring (arrays & objects)
* Spread (...) and Rest (...) operators
* Modules (import / export)
* Promises (asynchronous programming)
* Enhanced object literals
* Set and Map data structures
* for...of loop
* Symbols
* Generators and Iterators

**2. Explain JavaScript let**

The let keyword is used to declare **block-scoped** variables in JavaScript.

**Example:**

let x = 10;

if (true) {

let x = 20;

console.log(x); // 20

}

console.log(x); // 10

**Key Points:**

* Declares a variable limited to the block scope {}.
* Cannot be re-declared in the same scope.
* Can be updated (unlike const).

**3. Identify the Differences Between var and let**

| **Feature** | **var** | **let** |
| --- | --- | --- |
| **Scope** | Function-scoped | Block-scoped |
| **Hoisting** | Hoisted & initialized as undefined | Hoisted but **not** initialized |
| **Re-declaration** | Allowed | Not allowed in the same scope |
| **Global object** | Adds to window object (in browsers) | Does **not** add to window |

**4. Explain JavaScript const**

The const keyword declares **block-scoped constants**, meaning the variable **cannot be reassigned** after declaration.

**Example:**

const pi = 3.14;

pi = 3.15; // Error

**Key Points:**

* Must be initialized at declaration.
* Cannot be reassigned.
* For objects or arrays, the **reference** can't change, but **contents** can.

const person = { name: "John" };

person.name = "Jane"; // OK

person = {}; // Error

**5. Explain ES6 Class Fundamentals**

ES6 introduced a cleaner, more readable syntax for creating objects using **classes**.

**Basic Syntax:**

class Person {

constructor(name) {

this.name = name;

}

greet() {

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

}

}

const p = new Person("Alice");

p.greet(); // Hello, Alice

**Features:**

* Uses constructor() to initialize
* Supports methods directly inside the class
* this refers to the instance

**6. Explain ES6 Class Inheritance**

ES6 allows **class inheritance** using the extends and super() keywords.

class Animal {

constructor(name) {

this.name = name;

}

speak() {

console.log(`${this.name} makes a noise`);

}

}

class Dog extends Animal {

speak() {

console.log(`${this.name} barks`);

}

}

const dog = new Dog("Rex");

dog.speak(); // Rex barks

**Key Points:**

* extends is used to inherit from another class.
* super() is used to call the parent class constructor.

**7. Define ES6 Arrow Functions**

Arrow functions are a shorter syntax for writing function expressions.

**Syntax:**

// Traditional

function add(a, b) {

return a + b;

}

// Arrow function

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

**Key Features:**

* Shorter syntax
* **No binding of this** – they inherit this from their parent scope
* Cannot be used as constructors

**8. Identify Set() and Map()**

**Set**

* A collection of **unique** values (no duplicates).
* Can store any data type.

**Example:**

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

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

**Map**

* A collection of **key-value** pairs.
* Keys can be any type (objects, numbers, strings).

**Example:**

const myMap = new Map();

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

myMap.set(1, 'One');

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