1. **List the features of ES6**

Features of ES6 (ECMAScript 2015)

ES6 introduced numerous features to JavaScript, enhancing its capabilities and developer experience. Key features include:

* let and const: Block-scoped variable declarations.
* **Arrow Functions**: Concise syntax for writing functions.
* **Template Literals**: Improved string interpolation and multi-line strings using backticks (` `).
* **Destructuring Assignment**: Conveniently extract values from arrays or objects.
* **Classes**: Syntactic sugar for creating constructor functions and managing inheritance.
* **Modules**: Standardized way to organize and reuse code across files.
* **Promises**: A mechanism for handling asynchronous operations.
* for...of loop: Iterates over iterable objects like arrays, strings, and Maps/Sets.
* **Default Parameters**: Allows setting default values for function parameters.
* **Rest and Spread Operators**: ... operator for collecting arguments into an array (rest) or expanding iterables (spread).
* **Map and Set**: New data structures for key-value pairs and unique values, respectively.

1. **JavaScript let**

The let keyword, introduced in ES6, is used for declaring variables. Unlike var, let creates block-scoped variables, meaning they are only accessible within the block (code enclosed within curly braces {}) in which they are defined.

Characteristics of let

* Block Scope: Variables declared with let are limited to the block they are defined in, preventing access from outside that block.
* No Hoisting to Scope Start: While let variables are hoisted, they are not initialized, so accessing them before declaration results in a ReferenceError,  This period before declaration is known as the Temporal Dead Zone (TDZ).
* No Re-declaration in the Same Scope: Attempting to redeclare a let variable within the same scope will throw a SyntaxError, avoiding accidental re-declarations.

1. **Differences between var and let**

|  |  |  |
| --- | --- | --- |
| Feature | var | let |
| Scope | Function-scoped or Global-scoped | Block-scoped |
| Redeclaration | Can be redeclared and updated | Can be updated, but not redeclared in the same scope |
| Hoisting | Hoisted and initialized with undefined | Hoisted but not initialized (TDZ) |
| Accidental Global | Can lead to accidental global variables | Prevents accidental global variable creation |

1. **JavaScript const**

The const keyword in JavaScript, also introduced in ES6, is used to declare variables whose values are intended to remain constant and cannot be reassigned after initialization.

Characteristics of const

* Block Scope: Similar to let, const variables are block-scoped.
* No Reassignment: Once a const variable is assigned a value, it cannot be reassigned. Attempting to do so will result in a TypeError.
* Must Be Initialized: A const variable must be assigned a value at the time of declaration. Declaring a const variable without an initial value will result in a SyntaxError.
* Immutable Binding, Not Value: It's important to note that const ensures the variable binding is immutable, meaning the variable cannot be reassigned to a different value. However, if the value assigned to const is an object or array, its properties or elements can still be modified.

1. **ES6 class fundamentals**

ES6 classes provide a more structured and organized way to create objects and implement object-oriented programming (OOP) principles like inheritance and encapsulation.

* Class Declaration: Classes are declared using the class keyword.
* constructor() Method: A special method within a class that is automatically called when a new instance of the class is created. It is used to initialize the properties of the object.
* Methods: Functions defined within a class that provide behavior for objects created from that class.
* this Keyword: Inside a class, the this keyword refers to the current instance of the class.

1. **ES6 class inheritance**

Class inheritance in ES6 allows one class to inherit properties and methods from another class, promoting code reuse and creating hierarchies between objects.

* extends Keyword: Used to establish inheritance between classes. The class being extended is the parent (or base/superclass), and the new class is the child (or derived/subclass).
* super() Keyword: Used within the child class's constructor to call the parent class's constructor and initialize inherited properties. It is crucial to call super() before accessing this in the child constructor.
* Inheritance Limitations: ES6 classes support single inheritance, meaning a class can only inherit from one parent class directly.

1. **ES6 arrow functions**

Arrow functions provide a shorter and more concise syntax for writing function expressions in ES6. They are defined using the fat arrow (=>) notation.

Key characteristics of arrow functions

* Concise Syntax: Omits the function keyword and curly braces for single-line expressions, and provides implicit return.
* Lexical this Binding: Arrow functions do not have their own this context. Instead, they inherit the this value from the enclosing lexical scope at the time they are defined. This makes them useful for callbacks and event handlers where preserving the context is important.
* No arguments Object: Arrow functions do not have their own arguments object. The rest parameters syntax (...args) can be used to access arguments if needed.
* Cannot be Constructors: Arrow functions cannot be used as constructors with the new keyword.

1. **Set() and Map()**

ES6 introduced new collection types, Set and Map, which offer advantages over traditional objects and arrays in specific scenarios.

* Set: A collection of unique values, meaning duplicates are automatically removed. It maintains the order of elements as they are inserted and provides efficient methods for adding, deleting, and checking the existence of values.
* Map: A collection of key-value pairs where the keys can be of any type, including objects or functions. It remembers the original insertion order of keys and provides efficient methods for managing key-value data.