## Conceptual Questions

### 1. Why are promises used in JavaScript? Explain the advantages of using promises over traditional callback functions.

* Promises are used in JavaScript to handle asynchronous operations in a more organized and manageable way. They provide a cleaner alternative to traditional callback functions and offer several advantages:

1. **Readability:** Promises provide a cleaner and more readable syntax, making asynchronous code easier to understand.
2. **Avoiding Callback Hell**: Promises help to avoid callback hell by allowing sequential execution without nested callbacks, leading to more maintainable code.
3. **Error Handling:** Promises have built-in error handling through .catch(), simplifying error management compared to passing errors through callbacks.
4. **Composition and Chaining:** Promises support method chaining, enabling the composition of complex asynchronous workflows in a concise manner.
5. **Support for Parallelism:** Promises allow for parallel execution of asynchronous tasks using methods like Promise.all() and Promise.race().
6. **Global Error Handling:** Promises facilitate global error handling, improving debugging and error reporting across the application.

### 2. What is a closure in JavaScript? Provide an example.

* A closure is a function that retains access to its enclosing scope's variables even after the scope has closed. This allows the function to access and manipulate the variables from the outer scope even after the outer function has finished executing.
  + Example:

//closure

// function createCounter() {

// let count = 0;

// increment: function() {

// count += 1;

// },

// getCount: function() {

// return count;

// }

// };

// }

// const counter = createCounter();

// counter.increment();

// console.log(counter.getCount());

// counter.increment();

// console.log(counter.getCount());

### 3. What is a callback function and why is it used in JavaScript?

* A callback function is a function passed as an argument to another function to be executed later when a specific event occurs or when a task is completed. Callbacks are commonly used in asynchronous programming to handle tasks that depend on the completion of other tasks. Callback functions allow JavaScript to be asynchronous, enabling tasks like fetching data from a server, handling user input, and executing code after a delay.

4. What are async/await in JavaScript and how do they improve asynchronous programming?

* Async/await is a syntactic sugar introduced in ES6 for handling asynchronous code in a more synchronous-like manner. It allows developers to write asynchronous code in a way that resembles synchronous code, making it easier to read and understand. Async is used to define a function that returns a promise, and await is used to wait for a promise to resolve before continuing with the execution of the code. Async/await improves asynchronous programming by:
* Simplifying error handling with try/catch blocks.
* Avoiding callback nesting, making code more readable and maintainable.
* Allowing sequential execution of asynchronous tasks, which is easier to reason about.

5. Write the difference between ES6 and JS.

* ES6 (ECMAScript 2015) is a version of JavaScript, so they are not fundamentally different but rather related. ES6 is a specific version of the ECMAScript specification that introduced new features and syntax to the JavaScript language. JavaScript refers to the programming language itself, while ES6 is a specific version or iteration of that language specification. ES6 introduces new syntax, features, and improvements to JavaScript, such as arrow functions, classes, let and const declarations, destructuring, template literals, and more.

### 6. What are some of the major features introduced in ES6?

* Some of the major features introduced in ES6 include:
* Arrow functions
* Classes
* let and const declarations
* Template literals
* Destructuring assignment
* Default parameters
* Rest parameters
* Spread syntax
* Promises
* Modules (import/export)
* Enhanced object literals (computed property names, shorthand method names)
* Iterators and generators
* Symbol data type
* Enhanced regular expressions (sticky flag, flags property)