JAVASCRIPT:

Q1. What is JavaScript and what is it used for?

Ans: A high-level, dynamic, interpreted programming language called JavaScript is used largely for developing interactive and dynamic user interfaces in web pages. It is browser-based and enables developers to dynamically change a web page's content and respond to user actions without requiring a page refresh. JavaScript is also used for server-side programming with technologies like Node.js, as well as for creating desktop and mobile applications using tools like Electron and React Native.

Q2. Can you explain hoisting in JavaScript?

Ans: Hoisting is a behavior in JavaScript where variables and function declarations are moved to the top of their scope, regardless of where they appear in the code. In other words, variables declared with the "var" keyword and function declarations are "hoisted" to the top of their scope, and are accessible before they are declared in the code.

Q3. Can you describe the difference between "==" and "===" in JavaScript?

Ans: In JavaScript, == checks for equality after type coercion (changing the type of the values to match), while "===" checks for equality without type coercion. This means that "===" only returns true if the values being compared have the same type and value.

Q4. How do you declare a variable in JavaScript?

Ans: In JavaScript, variables can be declared using the "var", "let", or "const" keywords:
"var" is function scope, meaning it is accessible within the function it is declared. "let" and "const" are block-scoped, meaning they are only accessible within the block they are declared. "const" declares a constant variable that cannot be re-assigned.

Q5. Can you explain closure in JavaScript?

Ans: A closure in JavaScript is an inner function that has access to the outer function's variables and its own variables. The closure has access to variables in its outer scope even after the outer function has returned. This allows a closure to "remember" the values of its outer function's variables and use them even if they are no longer in scope. A closure is created when a function is declared inside another function. The closure has access to all variables in its outer scope, including those declared in its outer function and those passed as arguments to the outer function.
Closures are useful for creating private variables, preserving states across multiple invocations, and implementing factory functions and other design patterns.

Q6. Can you describe event bubbling and capture in JavaScript?

Ans: The sequence in which the elements get the event depends on the event propagation method. When an event is bubbling, the innermost element initially records and manages it before propagating to the surrounding components. When an event is captured, the outermost element initially records it before propagating it to the inner components.

Q7. Can you explain the difference between null and undefined in JavaScript?
Ans:
Null: An deliberate absence of value is represented by the symbol null. It symbolizes a variable whose value is unknown. It only takes the value null as a value. In TypeScript, the Null type is defined with the Null keyword, but since we can only give it a null value, it serves no purpose.

Undefined: In TypeScript and JavaScript, it stands for uninitialized variables. It only has the value "undefined" as a value. The undefined keyword in TypeScript specifies the undefined type, but because we can only give it an undefined value, it serves no purpose.

Q8. What is an object in JavaScript?
Ans: An object in JavaScript is a separate entity having attributes and a type. Consider comparing it to a cup. A cup is an item with characteristics. A cup has a design, weight, color, material, and other characteristics. In a similar manner, JavaScript objects can have properties that specify their attributes.

Q9: Can you describe the difference between the let and var keywords in JavaScript?
Ans: The scope of the variables that let and var generate is where they differ from one another:
let-declared variables are exclusively accessible within the block in which they are defined. Var-declared variables are accessible for the duration of the function in which they are declared.

Q10. Can you explain the "this" keyword in JavaScript and its use cases?
Ans: The "this" keyword designates an entity that is carrying out the currently running program. It makes a reference to the object performing the active function. When a normal function is being referred to, "this" refers to the global object.

Q11. Can you describe the difference between synchronous and asynchronous code in JavaScript?
Ans: Asynchronous code execution enables the instantaneous execution of incoming instructions, whereas synchronous code execution occurs according to a certain sequence of instructions specified in the program. Asynchronous programming allows us to prevent task blocking brought on by earlier instructions.

Q12. Can you explain how to use promises in JavaScript?

Ans: a promise is a returned object to which we can attach callbacks, instead of passing callbacks into a function. A function, createAudioFileAsync(), which asynchronously generates a sound file given a configuration record and two callback functions: one called if the audio file is successfully created, and the other called if an error occurs.

Q13. Can you explain the difference between call, apply, and bind methods in JavaScript?
Ans: call: binds this value, invokes the function and allows you to pass a list of arguments.
apply: binds this value, invokes the function, and allows you to pass arguments as an array.

bind: binds this value, returns a new function and allows you to pass in a list of arguments.

Q14. Can you explain the concept of prototypes in JavaScript and how they work?

Ans: An existing feature of JavaScript is referred to as a prototype. JavaScript adds a prototype attribute to every function we write in the language. A prototype is an object that has the ability to modify the original object by adding new variables and methods.

Q15. Can you describe the use of Arrow functions in JavaScript and their benefits?

Ans: Arrow functions in JavaScript are a shorthand for writing anonymous functions. They provide a concise syntax for defining functions and automatically bind the value of this to the context in which they are defined. The benefits of using arrow functions include

1. Concise syntax: Arrow functions allow you to write functions with less code.
2. Implicit this binding: The value of this is automatically set to the context in which the function is defined, making it easier to work with this in your code.
3. No arguments object: Arrow functions do not have an arguments object, so you must use rest parameters instead.
4. Better for use in callbacks: Arrow functions are well-suited for use in callbacks and other functional programming constructs, as they are automatically bound to their surrounding context.

ES6 :

Q1: What is ES6?

Ans: JavaScript ES6 (also known as ECMAScript 2015 or ECMAScript 6) is the newer version of JavaScript that was introduced in 2015. ECMAScript is the standard that JavaScript programming language uses. ECMAScript provides the specification on how JavaScript programming language should work.

Q2. What are some of the new features added in ES6?

Ans: the best and most popular ES6 features that we can use in your everyday JavaScript coding are:

1. let and const Keywords
2. Arrow Functions
3. Multi-line Strings
4. Default Parameters
5. Template Literals
6. Destructuring Assignment
7. Enhanced Object Literals
8. Promises
9. Classes
10. Modules

Q3. What is the purpose of "let" and "const" in ES6?

Ans: `const` is a signal that the identifier won't be reassigned. `let` is a signal that the variable may be reassigned, such as a counter in a loop, or a value swap in an algorithm. It also signals that the variable will be used only in the block it's defined in, which is not always the entire containing function.

Q5. What is the difference between arrow functions and traditional functions in ES6?

Ans: In a normal function, the word "arguments" refers to a list of the parameters that were supplied into the function. Regular functions require you to return any value at all times, whereas Arrow functions allow you to omit the return keyword and write in a single line. Parameters for the arrow function have to be distinct.

Q6. What are template literals in ES6 and how are they used?

Ans: In ES6, a brand-new feature called template literals was included. It offers a simple method for string interpolation and multiline string creation. String literals are known as template literals, and they permit embedded expressions. Template literals were known as template strings prior to ES6.

Q7. What is destructuring in ES6 and how is it used?

Ans: An interesting addition to ES6 was the destructuring assignment. Destructuring is a JavaScript expression that enables the separation of array values or objects attributes into separate variables. In other words, we are able to assign data to variables by extracting it from arrays and objects.

Q8. How does the spread operator work in ES6?

Ans: The spread operator facilitates the expansion of iterable objects into separate parts. Iterable objects, such as Arrays, Maps, Sets, etc., are ones that may be used with a loop. In other words, we may transfer every element from an array or object into another array or object using the spread operator.

Q9. What are the different ways to define default function parameters in ES6?

Ans: If a function parameter's value is null or undefined, default values are set for that parameter. By default, the arguments of a JavaScript function are specified as undefined. However, using a different default value could be advantageous.

Q10. What are classes in ES6 and how are they used?

Ans: Through the use of functions and prototypes, the ubiquitous JavaScript technique of replicating class-like inheritance hierarchies is formalized in ES6 Classes. By providing a handy declarative form for class patterns that promote interoperability, they are essentially simple sugaring over prototype-based OO.

Q11. What is the purpose of the "for...of" loop in ES6?

Ans: The for-of loop, introduced in the sixth edition of ES6, allows the programmer to loop over the actual iterable objects. This means that when used in an array, the variable used inside a for-of loop will return the element itself, and not the index.

Q12.What is the difference between the "for...of" loop and the "for...in" loop?

Ans: Both for...in and for...of statements iterate over something. The main difference between them is in what they iterate over. The for...in statement iterates over the enumerable string properties of an object, while the for...of statement iterates over values that the iterable object defines to be iterated over.

Q13. What are symbols in ES6 and how are they used?

Ans: In ES6, a new basic type called symbols was added. Symbols serve as entirely distinct identification. They may be generated using the factory function Symbol(), which yields a Symbol, much like its primitive equivalents (Number, String, and Boolean). Every time the factory function is used, a brand-new, exclusive sign is produced.

Q14. What is a generator function in ES6 and how is it used?

Ans: A brand-new class of function called the ES6 generator allows for pauses at any point during execution, or several times, and resumes execution afterward. In a conventional function, control passes from the called function to the called function until the called function returns, but in ES6, the generator function gives the calling function control over the called function's execution.

Q15. How does the "import" statement work in ES6?

Ans: The import statement is used to import modules that some other module has exported. A file containing code that may be reused is called a module. Whether declared or not, the import modules are in strict mode. Direct ES6 import support is not available in Node JS.