JS Concepts Assignment

1. The primitive data types in JavaScript are: string, number, boolean, null, and undefined.
2. A variable that is null is intentionally empty or non-existent. A variable that is undefined means it has been declared, but has not been assigned a value. An undeclared variable has not been declared in the current scope.
3. A while loop will continue to execute as long as the specified condition is true. A do-while loop will always execute the code block at least once, and then continue to execute as long as the specified condition is true.
4. For iterating over object properties, you can use a for-in loop. For iterating over array items, you can use a for loop or the forEach method.
5. A promise is an object that represents the result of an asynchronous operation. Promises have three states: pending, fulfilled, and rejected. You can use the then and catch methods to specify what should happen when the promise is fulfilled or rejected.
6. IIFE stands for Immediately Invoked Function Expression. It is a function that is defined and immediately called, without being assigned to a variable. IIFEs can be used to create a new scope and protect variables from being added to the global scope.
7. Event delegation is a technique for listening for events on multiple elements, by adding an event listener to a parent element instead of each individual element. This is useful when you have a large number of elements that all need the same event listener.
8. In JavaScript, the value of "this" depends on how a function is called. It refers to the object that is executing the current function. In ES6, the value of "this" can be more explicitly set using the bind, call, or apply methods.
9. Prototypal inheritance is a way of creating objects that inherit from other objects. In JavaScript, objects can have a prototype object, which they can inherit properties and methods from. You can use the Object.create method to create a new object with a specified prototype object.
10. A closure is a function that has access to variables in its parent scope, even after the parent function has finished executing. Closures are often used to create private variables or to create functions with a specific context.
11. The Array.forEach loop will execute a provided function for each element in the array. The Array.map method will create a new array with the results of calling a provided function on every element in the original array. You would use Array.forEach when you want to perform an action on each element in an array, and Array.map when you want to transform each element in an array into a new value.
12. Anonymous functions are functions that do not have a name. They are often used as arguments for other functions, or as a quick way to define a function for a single use.
13. Native objects are objects that are built into the JavaScript language, such as Array and Date. Host objects are objects provided by the environment in which JavaScript is running, such as the window object in a web browser.

By Deepak Gauttam

14. Function Person() {} defines a function constructor. var person = Person() calls the Person function. var person = new Person() creates a new instance of the Person object using the Person function as a constructor.
15. When defining a function using the function keyword, it is added to the current scope. When defining a function using the var keyword, it is added to the current scope as a variable.
16. Function.call and Function.apply are methods that allow you to call a function with a specific this value and arguments. The difference between the two is in the way that the arguments are passed to the function. Function.call takes individual arguments, while Function.apply takes an array of arguments.
17. Function.prototype.bind creates a new function with a specified this value and arguments. The new function can be called later with a different this value.
18. Feature detection is a technique for checking if a specific feature or capability is available in the current environment. Feature inference is a technique for making assumptions about the availability of a feature based on other features that are present. Using the UA string refers to checking the user agent string of the browser to determine what features are available.
19. Hoisting is a behavior in JavaScript where declarations of variables and functions are moved to the top of the current scope, before any code is executed. This means that variables and functions can be used before they are declared, as they are treated as if they were declared at the top of the scope.
20. Event bubbling refers to the way that events propagate from the element that triggered the event, up through its parent elements, and eventually to the top of the document.
21. Event capturing is the opposite of event bubbling, where events propagate from the top of the document down to the element that triggered the event.
22. An attribute is a default value for a property. A property is the current value of an attribute, which can be changed by the script.
23. Extending built-in JavaScript objects can be useful for adding additional functionality, but it can also cause conflicts with other code that expects the object to behave in a certain way. It is generally better to create your own objects or to use a library that provides the additional functionality you need, rather than modifying built-in objects.
24. The == operator performs type coercion, which means that it converts the operands to the same type before making the comparison. The === operator does not perform type coercion, so the operands must be the same type in order for the comparison to be true.
25. The same-origin policy is a security measure that prevents a web page from making requests to a different domain than the one that served the web page. This is to prevent a malicious web page from making unauthorized requests to a different site on behalf of the user.
26. The Ternary operator is called that because it takes three operands. The word "Ternary" means "composed of three". The Ternary operator is a concise way of writing an if-else statement.
27. Strict mode is a way to opt in to a stricter set of rules for JavaScript code. It eliminates some of the silent errors that are allowed in normal JavaScript code, and throws more

exceptions. The advantages of strict mode are that it can help you catch mistakes in your code, and it can make your code more efficient by eliminating some unnecessary features. The disadvantage is that it can make your code more difficult to write, if you are not used to the stricter rules.

28. Writing JavaScript code in a language that compiles to JavaScript can offer some advantages, such as additional features or a more familiar syntax. However, it can also add complexity to your workflow, as you need to set up a build process to compile the code, and there may be a learning curve for the new language.
29. Some tools and techniques for debugging JavaScript code include using the console to log values or print error messages, using the debugger statement to pause execution and inspect variables, and using a debugger tool in the browser or in an editor to set breakpoints and step through code.
30. Mutable objects can be modified after they are created. Immutable objects cannot be changed once they are created. An example of an immutable object in JavaScript is a String object created using the String.fromCharCode method. The pros of immutability are that it can make your code easier to reason about and can help prevent unintended side effects. The cons are that it can require more memory and processing power to create new objects instead of modifying existing ones. To achieve immutability in your own code, you can use Object.freeze to prevent object properties from being modified, and use methods like Object.assign or the spread operator to create new objects based on existing ones.
31. Synchronous functions are executed in a blocking manner, which means that the code will stop and wait for the function to finish before continuing. Asynchronous functions are executed concurrently with the rest of the code, and do not block the execution.
32. The event loop is a mechanism in JavaScript that checks the call stack and executes any code that is waiting to be run. The call stack is a data structure that stores the execution context of synchronous code, while the task queue is a data structure that stores the execution context of asynchronous code.
33. Variables created using the let keyword are block-scoped, which means they are only available within the block of code in which they are defined. Variables created using the var keyword are function-scoped, which means they are available within the entire function in which they are defined. Variables created using the const keyword are also block-scoped, but their value is read-only and cannot be reassigned.
34. ES6 classes are a syntax for defining object-oriented classes in JavaScript. They are similar to function constructors, but the syntax is more concise and easier to read. ES5 function constructors are a way to create objects using a constructor function and the new keyword.
35. The arrow => function syntax can be used as a concise way to define a function. It is especially useful for anonymous functions that are being passed as arguments, or for defining methods in an object literal. The main difference between the arrow function syntax and other functions is that the value of "this" is lexically scoped, which means it is determined by the surrounding code.
36. Using the arrow syntax for a method in a constructor allows the method to use the same this value as the constructor, without the need to bind the this value.

37. A higher-order function is a function that takes one or more functions as arguments, or returns a function as its result. Higher-order functions are useful for abstracting common patterns of function composition or for creating reusable utility functions.
38. An example of destructuring an object:

```
const obj = {a: 1, b: 2, c: 3};
const {a, b} = obj;
console.log(a); // 1
console.log(b); // 2
```

An example of destructuring an array:

```
const arr = [1, 2, 3];
const [a, b] = arr;
console.log(a); // 1
console.log(b); // 2
```

39. An example of generating a string with ES6 Template Literals:

```
const name = 'Deepak';
const age = 28;
console.log(`My name is ${name} and I am ${age} years old.`);
// Output: "My name is Deepak and I am 28 years old."
```

40. Here is an example of a curry function:

```
function calculateVolume(length) {
  return function (breadth) {
    return function (height) {
      return length * breadth * height;
    }
  }
}
console.log(calculateVolume(4)(5)(6));
```

The advantage of using curry is that it allows you to create a new function with a fixed number of arguments, which can be useful for creating reusable utility functions or for partial application of arguments.

41. Some benefits of using spread syntax include:

- Concise syntax for expanding elements of an iterable into individual elements.
- Can be used in function calls, array literals, and object literals.
- Can be used to create a shallow copy of an array or object.

The rest syntax is similar to spread syntax, but it is used to capture multiple elements into an array. It is used in function parameter definitions, and it allows you to capture an indefinite number of arguments into an array.

42. There are several ways to share code between files in JavaScript:

- You can use a module bundler, such as Webpack or Rollup, to bundle your code into a single file that can be included in your HTML using a script tag.
- You can use a module system, such as Common JS or ES6 modules, to import and export code between files.
- You can include multiple script tags in your HTML, each pointing to a separate JavaScript file.

43. You might want to create static class members in JavaScript when you want to create methods or properties that are associated with the class itself, rather than with instances of the class. Static class members can be accessed using the class name, rather than an instance of the class. Here is an example of a static method:

```
class MyClass {
  static staticMethod() {
    console.log('This is a static method');
  }
}
MyClass.staticMethod(); // Output: "This is a static method"
```