Q1. What do you understand about JavaScript?

JavaScript is a popular web scripting language and is used for client-side and server-side development The JavaScript code can be inserted into HTML pages that can be understood and executed by web browsers while also supporting object-oriented programming abilities

Q2. What are Client side and Server side?

A client is a device, application, or software component that requests and consumes services or resources from a server.

A server is a device, computer, or software application that provides services, resources, or functions to clients.

Q3. What are variables? What is the difference between var, let, and const ?

1 Var creates a function-scoped variable.

2. let creates a block-scoped variable

3 Const can be assigned only once, and its value cannot be changed afterwards.

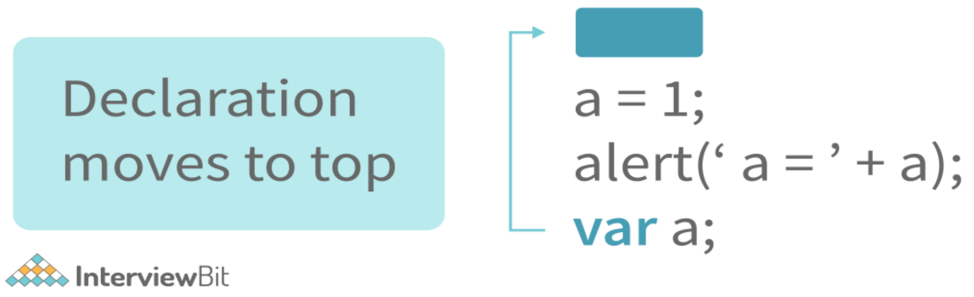
Q4. What is the latest version of JavaScript?

14th Edition – ECMAScript 2023

The 14th edition, ECMAScript 2023, was published in June 2023. This version introduces the toSorted , toReversed , with , findLast , and findLastIndex methods on Array.

Q5. Explain Hoisting in javascript.

Hoisting is the default behaviour of JavaScript where all the variable and function declarations are moved on top.



Q6. Why do we use the word “debugger” in javascript?

The debugger for the browser must be activated in order to debug the code. Built-in debuggers may be switched on and off, requiring the user to report faults. The remaining section of the code should stop execution before moving on to the next line while debugging.

Q7. Difference between “ == “ and “ === “ operators.

Both are comparison operators. The difference between both the operators is that “==” is used to compare values whereas, “ === “ is used to compare both values and types.

Q7. Explain Implicit Type Coercion in javascript.

Implicit type coercion in javascript is the automatic conversion of value from one data type to another. It takes place when the operands of an expression are of different data types.

Example:-

```
var x = 3;
var y = "3";

x + y // Returns "33"

var x = 24;
var y = "Hello";

x + y   // Returns "24Hello";
```

Q8. Is javascript a statically typed or a dynamically typed language?

JavaScript is a dynamically typed language. In a dynamically typed language, the type of a variable is checked during **run-time** in contrast to a statically typed language, where the type of a variable is checked during **compile-time.**

Q9. What do you mean by strict mode in javascript and characteristics of javascript strict-mode?

JavaScript's strict mode was introduced in ECMAScript 5. It enforces stricter parsing and error handling on the code at runtime. It also helps you write cleaner code and catch errors and bugs that might otherwise go unnoticed

```
"use strict";
x = 3.14;
```

Characteristics of strict mode in javascript

Duplicate arguments are not allowed by developers.

In strict mode, you won't be able to use the JavaScript keyword as a parameter or function name.

The 'use strict' keyword is used to define strict mode at the start of the script. Strict mode is supported by all browsers.

Engineers will not be allowed to create global variables in 'Strict Mode.

Q10. Explain Higher Order Functions in javascript.

Higher order functions are functions that take one or more functions as arguments, or return a function as their result

Q 11. Explain "this" keyword.

The this keyword refers to the current object in a method or constructor. The most common use of the this keyword is to eliminate the confusion between class attributes and parameters with the same name.

Q12. Explain call(), apply() and, bind() methods.

1. call():

It's a predefined method in javascript.

This method invokes a method (function) by specifying the owner object.

call() method allows an object to use the method (function) of another object.

2. bind():

This method returns a new function, where the value of "this" keyword will be bound to the owner object, which is provided as a parameter.

3.apply()
The apply method is similar to the call() method. The only difference is that, call() method takes arguments separately whereas, apply() method takes arguments as an array.

Q13. What is the difference between exec () and test () methods in javascript?

test () and **exec ()** are RegExp expression methods used in javascript.

We'll use **exec ()** to search a string for a specific pattern, and if it finds it, it'll return the pattern directly; else, it'll return an 'empty' result.

We will use a **test ()** to find a string for a specific pattern. It will return the Boolean value 'true' on finding the given text otherwise, it will return 'false'.

Q14. What is currying in JavaScript?

Currying is an advanced technique to transform a function of arguments n, to n functions of one or fewer arguments.

Example of a curried function:

```
function add (a) {
  return  function(b){
    return a + b;
  }
}
add(3)(4)
```

For Example, if we have a function **f(a,b)**, then the function after currying, will be transformed to **f(a)(b).**
By using the currying technique, we do not change the functionality of a function, we just change the way it is invoked.

Q15. Explain Scope and Scope Chain in javascript.

Scope in JS determines the accessibility of variables and functions at various parts of one's code.
In general terms, the scope will let us know at a given part of code, what are variables and functions we can or cannot access.
There are three types of scopes in JS:

Global Scope

Local or Function Scope

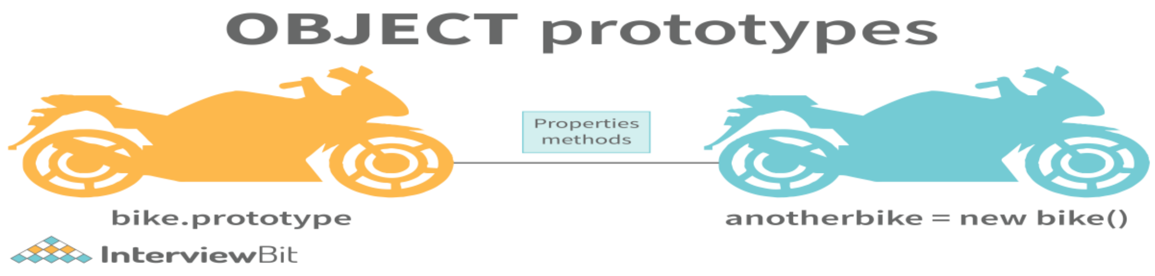
Block Scope

Q16. Explain Closures in JavaScript.

JavaScript **closure** is a feature that allows inner functions to access the outer scope of a function. Closure helps in binding a function to its outer boundary and is created automatically whenever a function is created. A block is also treated as a scope since ES6. Since JavaScript is event-driven so closures are useful as it helps to maintain the state between events.

Q17. What are object prototypes?

On top of the chain is **Object.prototype.** Every prototype inherits properties and methods from the Object.prototype.**A prototype is a blueprint of an object. The prototype** allows us to use properties and methods on an object even if the properties and methods do not exist on the current object.



Q18. What are callbacks?

A callback is a function that will be executed after another function gets executed. In javascript, functions are treated as first-class citizens, they can be used as an argument of another function, can be returned by another function, and can be used as a property of an object.

Q19. What is the use of a constructor function in javascript?

Constructor functions are used to create objects in javascript.

When do we use constructor functions?

If we want to create multiple objects having similar properties and methods, constructor functions are used.

Note- The name of a constructor function should always be written in Pascal Notation: every word should start with a capital letter.

Q20. What is DOM?

- DOM stands for Document Object Model. DOM is a programming interface for HTML and XML documents.
- When the browser tries to render an HTML document, it creates an object based on the HTML document called DOM. Using this DOM, we can manipulate or change various elements inside the HTML document.

Q21. What do you mean by BOM?

Browser Object Model is known as BOM. It allows users to interact with the browser. A browser's initial object is a window. As a result, you may call all of the window's functions directly or by referencing the window. The document, history, screen, navigator, location, and other attributes are available in the window object.

Q22. What are arrow functions?

Arrow functions were introduced in the ES6 version of javascript. They provide us with a new and shorter syntax for declaring functions. Arrow functions can only be used as a function expression.

Q23. what do mean by prototype design pattern in javascript

The prototype design pattern is a creational design pattern in JavaScript that allows objects to be created from existing objects, serving as prototypes. The prototype pattern is implemented using the Object.create() method. This method takes two arguments: the prototype object and an optional properties object.

```
const person = {
  name: 'John Doe',
  age: 30,
};
const newPerson = Object.create(person);
console.log(newPerson.name); // John Doe
console.log(newPerson.age); // 30
```

Q24. What is the rest parameter and spread operator?

Rest parameters are used to create functions that accept any number of arguments.

The spread syntax is used to pass an array to functions that normally require a list of many arguments.

Q25. In JavaScript, how many different methods can you make an object?

In JavaScript, there are several ways to declare or construct an object.

Object.

- using Class.

- create Method.

- Object Literals.

- using Function.

- Object Constructor.

What is the use of promises in javascript?

Q26. Promises are used to handle asynchronous operations in javascript.

Before promises, callbacks were used to handle asynchronous operations. But due to the limited functionality of callbacks, using multiple callbacks to handle asynchronous code can lead to unmanageable code.
Promise object has four states -

- Pending - Initial state of promise. This state represents that the promise has neither been fulfilled nor been rejected, it is in the pending state.
- Fulfilled - This state represents that the promise has been fulfilled, meaning the async operation is completed.
- Rejected - This state represents that the promise has been rejected for some reason, meaning the async operation has failed.
- Settled - This state represents that the promise has been either rejected or fulfilled.

Q27. What are generator functions?

Generator functions are special types of functions that can be paused and resumed during runtime, making them ideal for dealing with asynchronous operations, handling large datasets, and writing custom iterators. They are defined using the function* keyword and use the yield keyword to pause the function's execution and produce a value. When a generator function is called, it returns an iterator object, which can be used to control the function's execution.

```
function* generateNumbers() {
  yield 1;
  yield 2;
  yield 3;
}
const generator = generateNumbers();
console.log(generator.next().value); // 1
console.log(generator.next().value); // 2
console.log(generator.next().value); // 3
```

Q28. Explain WeakSet in javascript.

A WeakSet is a collection of unique objects. It is a JavaScript object that allows you to store a collection of objects and test whether an object is in the collection. WeakSet objects are not iterable, and do not have the size property.

-Unlike Set, WeakSet only has three methods, add() , delete() and has() .

Q29. Explain WeakMap in javascript.

In javascript, Map is used to store key-value pairs. The key-value pairs can be of both primitive and non-primitive types. WeakMap is similar to Map with key differences:

- The keys and values in weakmap should always be an object.
- If there are no references to the object, the object will be garbage collected.

```
const map1 = new Map();
map1.set('Value', 1);

const map2 = new WeakMap();
map2.set('Value', 2.3); // Throws an error

let obj = {name:"Vivek"};
const map3 = new WeakMap();
map3.set(obj, {age:23});
```

Q30. What is a Temporal Dead Zone?

Temporal Dead Zone is a behaviour that occurs with variables declared using **let** and **const** keywords. It is a behaviour where we try to access a variable before it is initialized. Examples of temporal dead zone:

```
x = 23; // Gives reference error
let x;

function anotherRandomFunc(){
  message = "Hello"; // Throws a reference error

  let message;
}
anotherRandomFunc();
```

In the code above, both in the global scope and functional scope, we are trying to access variables that have not been declared yet. This is called the **Temporal Dead Zone**

Q30. What do you mean by JavaScript Design Patterns?

JavaScript design patterns are repeatable approaches for errors that arise sometimes when building JavaScript browser applications. They truly assist us in making our code more stable.

They are divided mainly into 3 categories

1. Creational Design Pattern
2. Structural Design Pattern
3. Behavioral Design Pattern.

- **Creational Design Pattern:** The object generation mechanism is addressed by the JavaScript Creational Design Pattern. They aim to make items that are appropriate for a certain scenario.
- **Structural Design Pattern:** The JavaScript Structural Design Pattern explains how the classes and objects we've generated so far can be combined to construct bigger frameworks. This pattern makes it easier to create relationships between items by defining a straightforward way to do so.
- **Behavioral Design Pattern:** This design pattern highlights typical patterns of communication between objects in JavaScript. As a result, the communication may be carried out with greater freedom.

Q31. Difference between Async/Await and Generators usage to achieve the same functionality.

- Generator functions are run by their generator yield by yield which means one output at a time, whereas Async-await functions are executed sequentially one after another.
- Async/await provides a certain use case for Generators easier to execute.
- The output result of the Generator function is always value: X, done: Boolean, but the return value of the Async function is always an assurance or throws an error.

Q32. What are the primitive data types in JavaScript?

A primitive is a data type that isn't composed of other data types. It's only capable of displaying one value at a time. By definition, every primitive is a built-in data type (the compiler must be knowledgeable of them) nevertheless, not all built-in datasets are primitives. In JavaScript, there are 5 different forms of basic data. The following values are available:

1. Boolean
2. Undefined
3. Null
4. Number
5. String

Q33. Q. What are Functions in JS? What are the types of function?

There are three main types of functions in JavaScript: named functions, anonymous functions, and arrow functions

Named Functions

Named functions are function declarations that include a name. They are also sometimes referred to as traditional functions or function declarations. Here is an example:

```
function helloWorld() {
  console.log('Hello World!');
}
```

Anonymous functions

Anonymous functions are function expressions that do not have a name associated with them. They are also sometimes referred to as lambda functions or function expressions. Here is an example:

```
let helloWorld = function () {
  console.log('Hello World!');
}
```

Anonymous functions are typically used when the same code needs to be executed only once or twice throughout a program. They can be assigned to variables or passed as parameters to other functions.

Arrow Functions

Arrow functions are a shorthand syntax for anonymous functions that was introduced in ECMAScript 6 (ES6). They are also sometimes referred to as fat arrow functions or lambda shorthand. Here is an example:

```
let helloWorld = () =&gt; {
  console.log('Hello World!');
}
```

Arrow functions are typically used when writing concise code and when the same code needs to be executed only once or twice throughout a program. They can be assigned to variables or passed as parameters to other functions.

Q34. What is JSON?

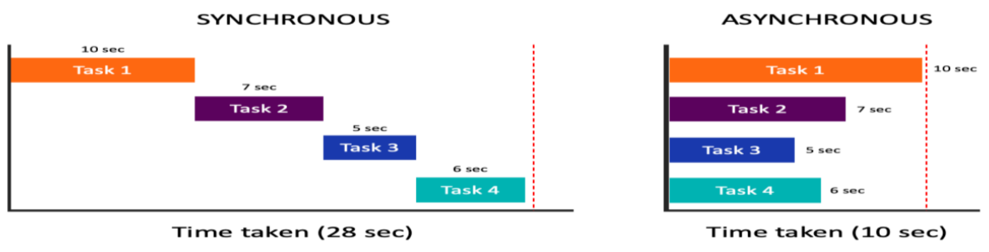
JSON (JavaScript Object Notation) is a lightweight data interchange format.

- JSON consists of key-value pairs

Q35. What is asynchronous programming in JS? What is its use?

Asynchronous programming allows multiple tasks or operations to be initiated and executed concurrently.

Asynchronous operations do not block the execution of the code.



Q 36. What is the difference between the slice() and splice() methods of an Array?

The slice() method is used get a subset of the array from the start index to the end index(end not included).

The splice() method is used to add, remove, or replace elements in an array.

Q 37. Explain the difference between dot notation and bracket notation.

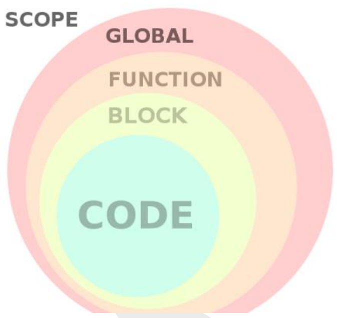
Both dot notation and bracket notation are used to access properties or methods of an object.

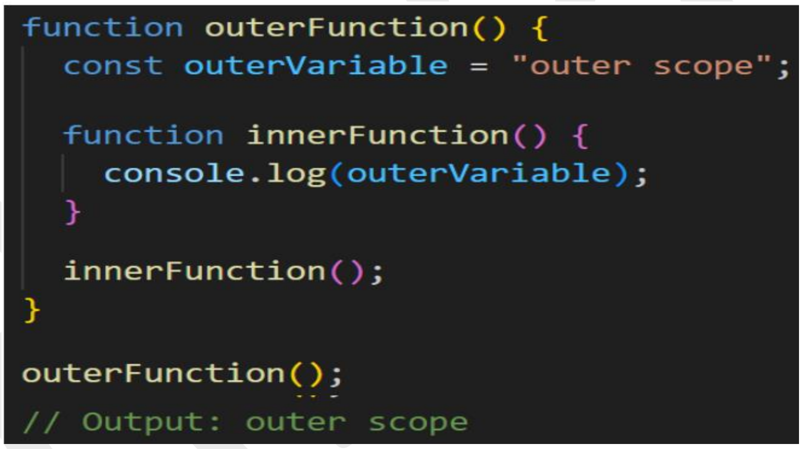
Dot notation is more popular and used due to its simplicity.

Limitation of dot notation - In some scenarios bracket notation is the only option, such as when accessing properties when the property name is stored in a variable.

Q 38. Explain the concept of Lexical Scoping?

The concept of lexical scoping ensures that variables declared in an outer scope are accessible in nested functions.





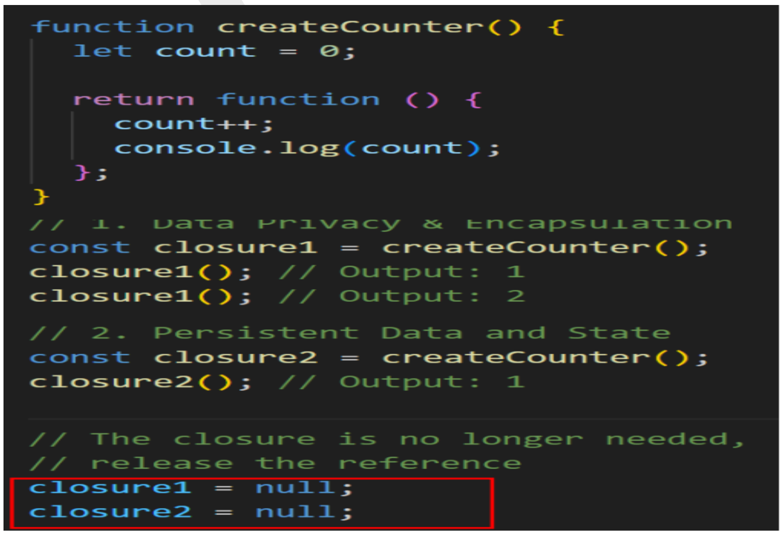
```javascript
function outerFunction() {
  const outerVariable = "outer scope";

  function innerFunction() {
    console.log(outerVariable);
  }

  innerFunction();
}

outerFunction();
// Output: outer scope
```

Q 39. What is Closure?

A closure in JavaScript is a combination of a function and the lexical environment Closures are used to modify data or variables safely.

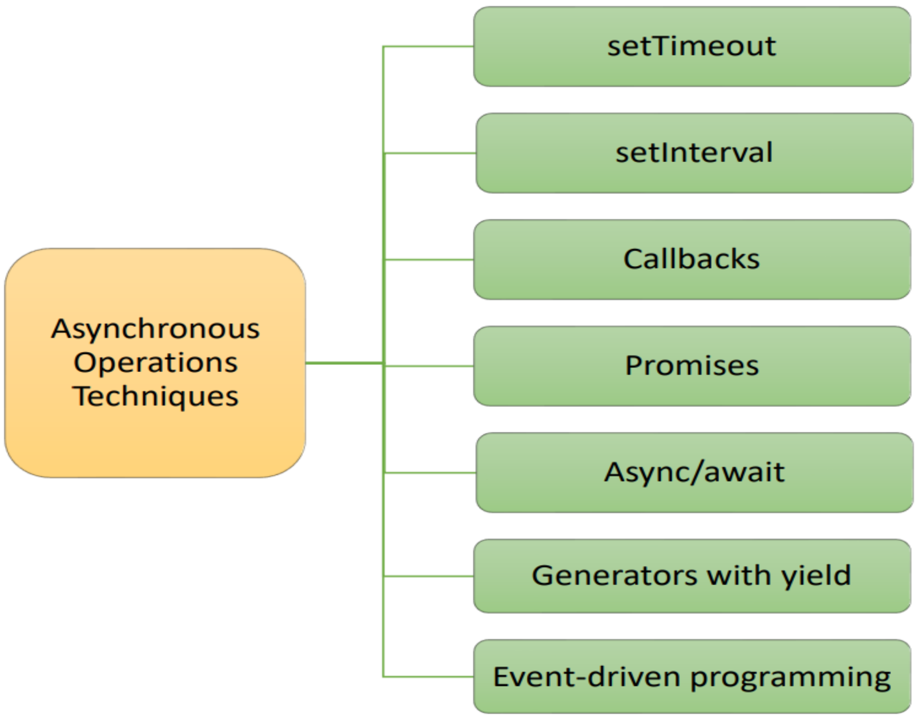
Q 40. How can you release the variable references or closures from memory?

You can release the reference to the closure by setting closure to null.



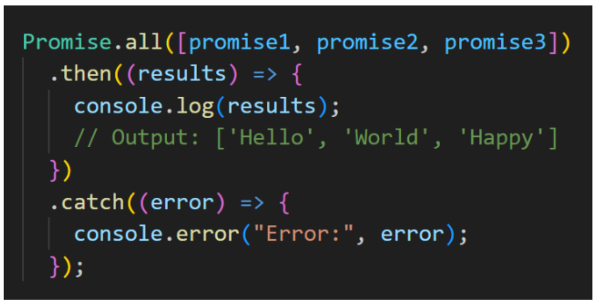
```javascript
function createCounter() {
  let count = 0;

  return function () {
    count++;
    console.log(count);
  };
}
// 1. Data Privacy & Encapsulation
const closure1 = createCounter();
closure1(); // Output: 1
closure1(); // Output: 2

// 2. Persistent Data and State
const closure2 = createCounter();
closure2(); // Output: 1

// The closure is no longer needed,
// release the reference
closure1 = null;
closure2 = null;
```

Q41. What are the techniques for achieving asynchronous operations in JS.

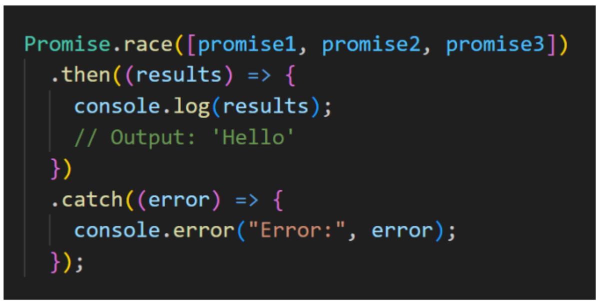


Q 42. What is the difference between Promise.all() and Promise.race()?

Promise.all()	**Promise.race()**
1. Promise.all() is used when you want to wait for **all** the input promises to settle.	Promise.race() is used when you want to react as soon as the **first** promise settles.
2. The returned promise resolves with an **array** of resolved values from the input promises, in the same order as the input promises.	The settled **value** (fulfilled value or rejection reason) of the first settled promise is used as the settled value of the returned promise.



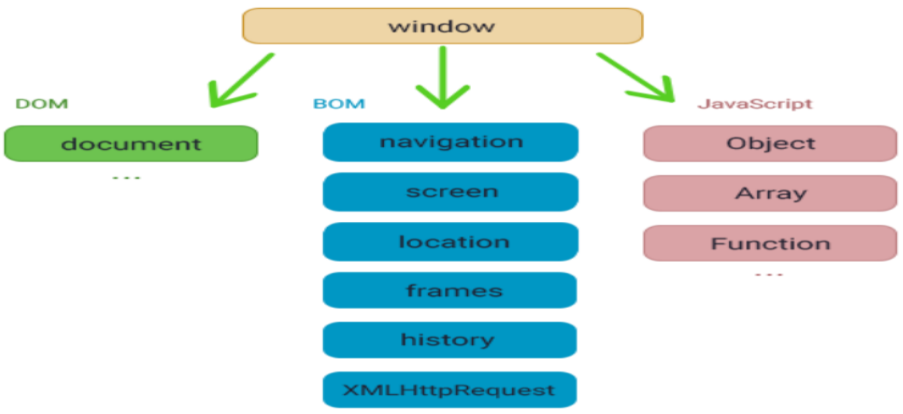
```javascript
Promise.all([promise1, promise2, promise3])
  .then((results) => {
    console.log(results);
    // Output: ['Hello', 'World', 'Happy']
  })
  .catch((error) => {
    console.error("Error:", error);
  });
```



```javascript
Promise.race([promise1, promise2, promise3])
  .then((results) => {
    console.log(results);
    // Output: 'Hello'
  })
  .catch((error) => {
    console.error("Error:", error);
  });
```

Q43. What is a window object?

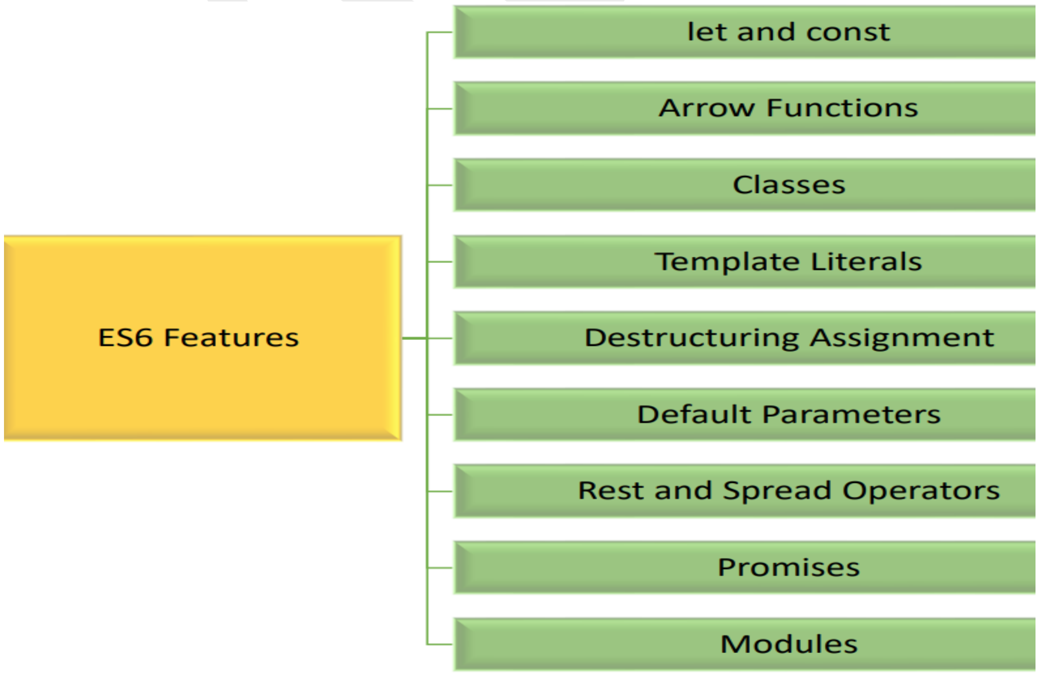
The window object in JavaScript is the global object in the browser environment. It represents the browser window or frame in which the JavaScript code is running and provides access to the browser's functionalities. The window object has various properties and methods that can be used to interact with the browser and the document loaded in the window.



Q 44. What is the difference between cookies and web storage?

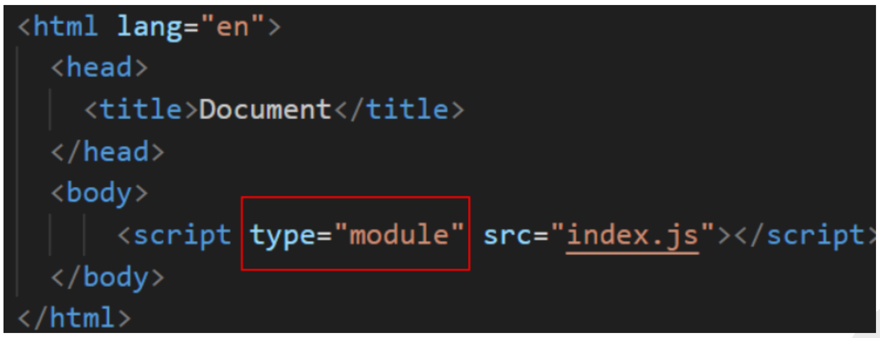
Cookies	Web Storage(Local/ Session)
1. Cookies have a **small storage** capacity of up to 4KB per domain.	Web storage have a **large storage** capacity of up to 5-10MB per domain.
2. Cookies are automatically sent with **every request**.	Data stored in web storage is not automatically sent with each request.
3. Cookies can have an **expiration date** set.	Data stored in web storage is not associated with an expiration date.
4. Cookies are accessible both on the client-side (via JavaScript) and server-side (via HTTP headers). This allows server-side code to read and modify cookie values.	Web Storage is accessible and modifiable only on the client-side.

Q 45. What is ES6? What are the new features introduced by it?



Q 46. What are Modules in JS?

Modules in JS are a way to organize code into separate files, making it easier to manage and reuse code across different parts of an application.



```html
<html lang="en">
  <head>
    <title>Document</title>
  </head>
  <body>
      <script type="module" src="index.js"></script
  </body>
</html>
```

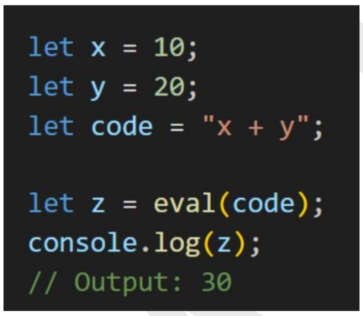
Q 47. Q. What is the difference between named exports and default exports?

- Named exports allow you to export multiple elements from a module.

- Default export allows you to export a single element as the default export from a module.

Q 48. What is eval() function in JS?

eval() is a built-in function that evaluates a string as a JavaScript code and dynamically executes it.



```javascript
let x = 10;
let y = 20;
let code = "x + y";

let z = eval(code);
console.log(z);
// Output: 30
```