1. **Difference b/w var, let, const**

VAR	LET	CONST
var is functionally or globally scoped.	let is block scoped.	const is block scoped.
Can be redeclared.	Cannot be re-declared within its scope.	Cannot be re-declared within its scope.
Can be declared without initialization.	Can be declared without initialization.	Must be initialized at the time of declaration.
Can be updated.	Can be updated.	Can never be updated.

2. **== vs ===**

   == only checks for the value
   === checks for value + type

3. **Map vs forEach**

   Map returns a new Array. Used to iterate over an Array. We can do chaining in Map.

   forEach does not return a new Array. . Used to iterate over an Array. We can't do chaining in forEach. We can change the original Array values in forEach.

4. **Global vs Functional vs block scope**

   Global (declaration outside of any function)

   Function (declaration inside a function)

   Block (declaration inside a block)

5. **What is "this" keyword in JS ?**

   In JavaScript, the this keyword refers to an object. Which object depends on how this is being invoked (used or called). The this keyword refers to different objects depending on how it is used: In an object method, this refers to the object. Alone, this refers to the global object. In a regular function this refers to the global object. In a function inside an object this refers to the function itself.

6. **Promises**
   A promise is an object that may produce a single value some time in the future: either a resolved value, or a reason that it’s not resolved (e.g., a network error occurred). A promise may be in one of 3 possible states: fulfilled, rejected, or pending. Promise users can attach callbacks to handle the fulfilled value or the reason for rejection.

   A promise is an object which can be returned synchronously from an

   asynchronous function. It will be in one of 3 possible states:

   - Fulfilled: onFulfilled() will be called (e.g., resolve() was called)
   - Rejected: onRejected() will be called (e.g., reject() was called)
   - Pending: not yet fulfilled or rejected

7. **What is Document Object Model(DOM)**
   The term "DOM" stands for **D**ocument **O**bject **M**odel and refers to the representation of the entire user interface of a web application as a tree data structure.

8. **What are Events in JS ?**

   An event is an action that occurs as per the user's instruction as input and gives the output in response. We can consider different types of inputs, such as mouse clicks, button presses, and when users press tab and text box change.

9. **What is event object in Js ?**
   When an event listener’s event occurs and it calls its associated function, it also passes a single argument to the function—a reference to the event object. The event object contains a number of properties that describe the event that occurred.

10. **CurrentTarget vs Target**

    currentTarget = The element that triggered the event (e.g., the user clicked on). Selects the whole element on which the event is applied on including all it's parents.

    target = The element that the event listener is attached to. Identifies the element on which it was occured. It selects only the element that is being targetted and not the whole element. This will only include itself and it's nested children.

11. **Propagation vs Bubbling vs Capturing vs Target.**

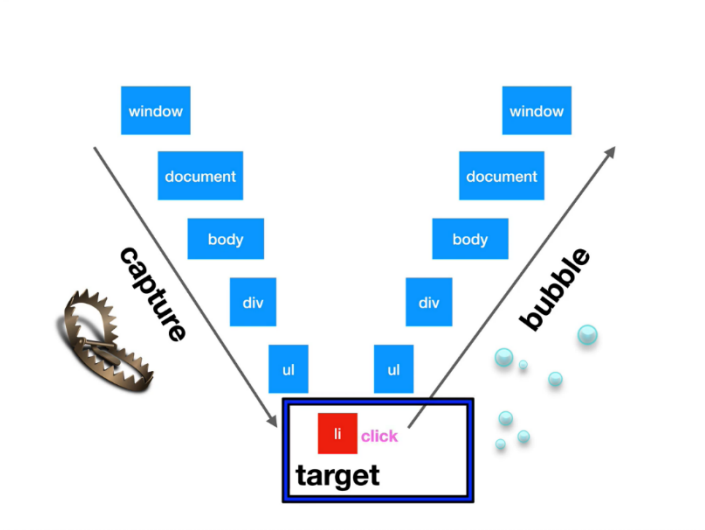
    Event Propagation = When an event is fired on a child element, that event actually propagates or spreads throughout the entire DOM hierarchy(i.e. to all it's parent elements). This process in done in Three Phases Capture, Bubble, Target.

    Let's say our DOM has
    Window → Document → body → div → ul → li

    Target = Let's say we fire an event on li, then element element(li) is called the target.

    Capturing = In capture phase the window element get's notified first of the click event, and then going down to the DOM tree Document element is notified next, then the body, div, ul and li.

    Bubbling = Once the target element has been notified by the capturing that an event has been fired on it, we go back up the DOM Tree i.e.
    li → ul → div → body → Document → Window. These elements are notified once again about the click event, though this time in reverse order.



The default phase of these actions occurring is Bubbling.
We can change the order of event fired from bubbling to capturing.

12. Call vs Apply vs Bind

call = runs instantly, i.e. we can't save it in a variable and access it later. Arguments = list of items. Used to call one object function onto another object without actually declaring that function inside that other object.

Apply = Runs instantly, Arguments = [array of items]. Also used to call function on objects.

Bind = Does not runs instantly, Can be assigned to a variable, to run later. Arguments = list of items.

13. IIFE

Immediately Invoked Function Expression. We don't need to invoke the function outside the function and do extra steps we can right away invoke the function as soon as the script runs.
IIFE is an old way, new and better way is to use Closure.
IIFE provides a simple approach to avoid global scope pollution or a good way at protecting the scope of your function and the variable within it.

14. **Spread vs Rest Operator**

    Spread Operator allows an iterable object to spread/expand individually inside reciever. Split into single items and creates a copy of it. Used to store data in Array, basically creates array with the iterable object items. Spread operator splits the items into single items.

    When Spread Operator is Passed inside as a function Argument, it becomes rest Operator, or if it is not passed as first item in the array object. What rest Operator allows us to do is to have an infinite number of inputs/argumets in our function. It works similar as Python args and *kwargs.

15. **forOf vs forIn loop**

    forOf = Loops through the values of an iterable object, string, array, map, set. Not Object Unlike forEach, in forOf we can use break and continue.

    forIn = Used to iterate over the object properties. Not advised to used in arrays, especially if order is important.

16. **What are modules in Js**

    A module in JavaScript is just **a file containing related code**. In JavaScript, we use the import and export keywords to share and receive functionalities respectively across different modules. The export keyword is used to make a variable, function, class or object accessible to other modules.

    In modules we divide our code into bunch of different files known as modules.

17. **What is Callback hell**

    Callback hell is a phenomenon that afflicts a JavaScript developer when he tries to execute multiple asynchronous operations one after the other. By nesting callbacks

```
boilWater();
function boilWater() {
  console.log("boiling water ............");
  setTimeout(() => {
    console.log("done boiling water");
    console.log("add carrots");
    setTimeout(() => {
      console.log("carrots are done boiling");
      console.log("chop Onion");
      setTimeout(() => {
        console.log("done boiling onion");
        console.log("soup is ready to serve");
      }, 5000);
    }, 5000);
  }, 10000);
}
```

18. **What is async await**

    The word “async” before a function means one simple thing: a function always returns a promise.

    The keyword "await" makes JavaScript wait until that promise settles and returns its result

19. **What is Ajax**

    AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.

    AJAX is not a programming language.

    AJAX just uses a combination of:

- A browser built-in XMLHttpRequest object (to request data from a web server)
- JavaScript and HTML DOM (to display or use the data)

AJAX allows web pages to be updated asynchronously by exchanging data with a web server behind the scenes. This means that it is possible to update parts of a web page, without reloading the whole page

20. Different ways to fetch an API in Js

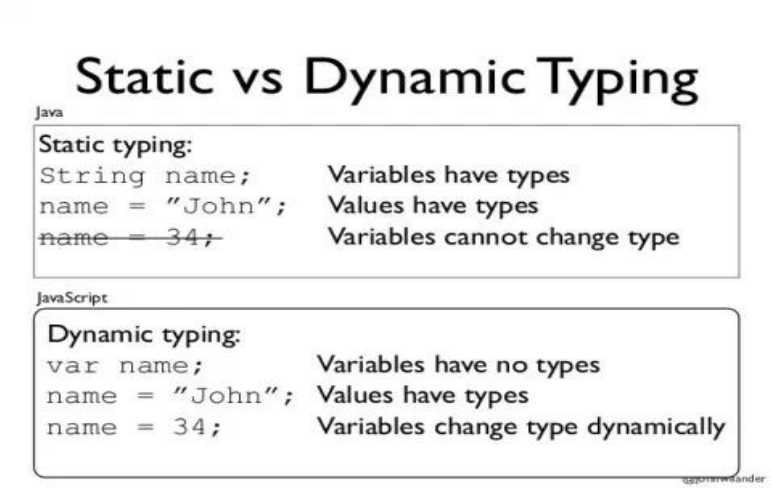
XMLHttpRequest(XML) – old way of fetching API's.

FetchAPI – Provided by browser

Axios – Js Library

21. Why do we call Js as Dynamic Programming Language

It means that JS does not require the explicit declaration of the variables before they're used. Here is a quick example that compares Java, a statically typed language, vs. JavaScript.



22. Why is JS Single Threaded Language

JavaScript is a single-threaded language, which means it has only one call stack that is used to execute the program. The call stack is the same as the stack data structure that you might read in Data structures. As we know stacks are FILO that is First In Last Out. Similarly, within the call stack, whenever a line of code gets inside the call stack it gets executed and move out of the stack. In this way, JavaScript is a single-thread language because of only one call stack.

23. Undefined vs null

Undefined = "JS cannot find the value for this". Ex.

a. Variable without value.
b. missing function arguments.
c. missing object properties.

null = "developer sets the value, null means nothing or empty value"

24. What is useStrict

The "use strict" directive was new in ECMAScript version 5.
It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.
The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
With strict mode, you can not, for example, use undeclared variables.

```
"use strict";
x = 3.14;    // This will cause an error because x is not declared
x = 3.14;       // This will not cause an error.
myFunction();

function myFunction() {
  "use strict";
  y = 3.14;   // This will cause an error
}
```

25. How does typeOf Operator works

A typeof is a unary operator that is placed before it's single operand, which can be of any type. It's value is a string indicating the data type of the operand.

26. innerHTML vs innerText vs textContent

innerHTML = Returns HTML node
textContext, innerText = Returns only readable text

27. Callback vs Higher Order Functions

Functions as First Class Objects - can be stored in a variable(expression), can be passed as an argument to another function,can be returned from another function.

Higher Order Functions - accepts another function as an argument or returns another function as a result.

CallBack Functions - passed to another function as an argument and executed inside that function.

28. What do you mean by the term functions as first class citizen in Js ?

A programming language is said to have First-class functions **when functions in that language are treated like any other variable**. For example, in such a language, a function can be passed as an argument to other functions, can be returned by another function and can be assigned as a value to a variable.

29. Currying in Js

It is a technique in functional programming, transformation of the function of multiple arguments into several functions of a single argument in sequence.

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

30. What is Lazy Loading.

Lazy loading is a strategy to identify resources as non-blocking(non-critical) and load these only when needed. It's a way to shorten the length of the critical rendering path, which translates into reduce page loading time.

In simple words:
Let's assume we have a website, and on that website we have 100's of pages i.e. home page, about page, contact page, dashboard, etc. Now when the user visits our website, lets say home page, but let's assume the user never visit any of the other pages, regardless of that all the page will load/download, i.e. user has to wait to download/load all the pages, unless he can finally see the page he's on, basically we are loading all the other pages unnecessarily. Let's say our home page is of 10 kb and our whole website is of 1mb, so we are basically download 1014kb of data unnecessarily, which results in slow page load time, especially when our internet is slow. This can also lead to security issues, especially for websites that have user authentication, otherwise user can view all the pages without even logging in. Similarly on Netflix the movies don't get loaded at first, only the thumbnail of that movie gets loaded, only when user clicks on the thumbnail, then only the actually video loads, if we don't do that, the page load time will drastically increase, since videos are a lot big in size, so user will end up wasting lot of it's time in loading and internet data. To fix this problem we use lazy-loading. Mostly we should lazy load images and videos since they are the ones that are big in size and can reduce performance of our website.

Types of lazy loading –

- Eagerly Loading Strategy – Downloads the whole website at once, after that only user can see a page.

- Lazy Loading Strategy – We only download the critical part i.e. only the resources that are required to view that vary page, the user is on.

- Pre-loading all modules – This basically means, first we download the page that the user is currently on, and once that page loads, we can asynchronously download all the other pages in the background.