



Q What do you mean by CORS (Cross-Origin Resource Sharing)?

CORS refers to cross-origin resource sharing. It's a browser mechanism that allows web pages in one domain to have controlled access to resources in other domains (cross-domain requests). This allows scripts that are run on a browser client to interact with and access resources from other origins. In doing so, it extends and provides greater flexibility to the SOP (Same-Origin Policy). Same-origin policies restrict the ability of a website to access resources outside of its source domain. For example, a JavaScript app that wants to make a call to an API (Application Programming Interface) that runs on another domain will be blocked due to the SOP. A CORS policy was implemented to circumvent restrictions caused by same-origin policies.

Q What is Callback Hell?

Callback Hell, or Pyramid of Doom, is a common anti-pattern seen in asynchronous programming code (multiple functions running at the same time). This slang term describes a large number of nested "if" statements or functions. In simple terms, Callback hell is a situation where you have multiple asynchronous functions. Those functions depend on one another, so it could get quite messy with so many callback functions nested in so many layers. The use of callback functions leaves you with code that is difficult to read and maintain, and looks like a pyramid as shown below:

This also makes it more difficult to identify the flow of the application, which is the main obstacle to debugging, which is the reason for the famous name of this problem: Callback Hell.

Q. Explain Long Polling.

Long polling is defined as a web application development technique used to push information/data from servers to clients as quickly as possible. When a request is made from the client to the server, long-polling maintains the connection between the two. This connection is maintained until the information is ready to be sent from the server to the client. Once a server receives a request from a client, the connection does not close immediately; the connection is only closed once the server has sent the data back to the client or when a timeout threshold has been reached (connection timeout)

Q. State difference between GraphQL and REST (Representational State Transfer).

GraphQL	REST
It is a Query Language for APIs that enables declarative data fetching to provide clients control over which data to retrieve from the API.	It is an API design architectural style that defines a set of constraints to create web services.
GraphQL is known for its high predictability. With this, you can send requests to your API and get the exact results you are looking for without having to include anything you don't need. GraphQL queries give predictable results, which improves their usability significantly.	On the other hand, REST's behaviour depends on the HTTP and URI methods used. API consumers can therefore be unsure of what to expect when calling an endpoint.
API security can be ensured by GraphQL, though the security features aren't as mature as those of REST. GraphQL, for instance, assists in integrating data validation, but users are left to figure out how to apply authentication and authorization measures.	API security can be enforced in several ways using REST. You can implement multiple methods of API authentication, such as HTTP authentication, to ensure REST API security.
With GraphQL, you can retrieve everything you need through a single API request. You can specify the structure of the information you need, and the server will return the same structure to you, so there is no need to over-and under-fetch.	Consequently, since REST APIs have rigid data structures that return the specified data whenever they are accessed, you may end up with unwanted data or have to make multiple requests before getting the right data. As a result, the server's response time (to return information) can be prolonged due to these shortcomings.
It only supports JSON format.	XML, YAML, JSON, HTML, and other formats are supported, as well.
The main use cases of GraphQL are mobile applications and multiple microservices.	Rest is mainly used for simple applications and resource-driven applications.

Q What is CI (Continuous Integration)?

CI (Continuous Integration), as its name implies, is the process of automating and integrating code changes into a single software project, often several times a day. The purpose of this DevOps practice is to enable developers to merge their code changes into a central repository where automated tests and builds can run. Automated tools are used to assert the new code's correctness before integration. A source code version control system is the crux of the CI process. The version control system is also supplemented with other checks like automated code quality tests, syntax style review tools, and more.

Q. Explain the meaning of multithreading.

The thread is an independent part or unit of a process (or an application) that is being executed. Whenever multiple threads execute in a process at the same time, we call this "multithreading". You can think of it as a way for an application to multitask.

Q. Explain the benefits and drawbacks of using "use strict".

In ECMAScript5, a new feature known as strict mode allows you to run a program or function within a strict operating context. Certain actions, therefore, cannot be taken due to this strict context, and more exceptions are thrown. When the "use strict" statement is used, the browser is instructed to use "strict" mode, which is a more restricted and safer JavaScript feature set. You can specify "use strict" at the top of a function to evaluate the JS in strict mode. In strict mode, more errors are thrown and some features are disabled to make your code more robust, clear, and accurate.

Q Explain event loop in Node.js.

The event loop in JavaScript enables asynchronous programming. With JS, every operation takes place on a single thread, but through smart data structures, we can create the illusion of multi-threading. With the Event Loop, any async work is handled by a queue and listener.

Therefore, when an async function (or an I/O) needs to be executed, the main thread relays it to another thread, allowing v8 (Javascript engine) to continue processing or running its code. In the event loop, there are different phases, like pending callbacks, closing callbacks, timers, idle or preparing, polling, and checking, with different FIFO (First-In-First-Out) queues.

Q Explain dependency injection.

The Dependency Injection (DI) pattern is a design pattern for implementing the Inversion of Control (IoC). Dependent objects can be created outside of classes and made available to classes in different ways. Three types of classes are involved in Dependency Injection as follows:

Client Class: A client class (dependent class) is one that depends on the service class.

Service Class: Service (dependency) classes provide services to client classes.

Injector Class: This class injects the objects from the service class into the client class.

Q. What are the limitations of React?

The few limitations of React are as given below:

1. React is not a full-blown framework as it is only a library.
2. The components of React are numerous and will take time to fully grasp the benefits of all.
3. It might be difficult for beginner programmers to understand React.
4. Coding might become complex as it will make use of inline templating and JSX.

Q What is useState() in React?

The useState() is a built-in React Hook that allows you for having state variables in functional components. It should be used when the DOM has something that is dynamically manipulating/controlling.

Q What are keys in React?

A key is a special string attribute that needs to be included when using lists of element. Keys help react identify which elements were added, changed or removed.

1. Keys should be given to array elements for providing a unique identity for each element.
2. Without keys, React does not understand the order or uniqueness of each element.
3. With keys, React has an idea of which particular element was deleted, edited, and added.
4. Keys are generally used for displaying a list of data coming from an API

Q What are the differences between functional and class components?

Before the introduction of Hooks in React, functional components were called stateless components and were behind class components on a feature basis. After the introduction of Hooks, functional components are equivalent to class components.

Q. What are the differences between controlled and uncontrolled components?

Controlled and uncontrolled components are just different approaches to handling input from elements in react.

Feature	Uncontrolled	Controlled	Name attrs
One-time value retrieval (e.g. on submit)	✓	✓	✓
Validating on submit	✓	✓	✓
Field-level Validation	✗	✓	✓
Conditionally disabling submit button	✗	✓	✓
Enforcing input format	✗	✓	✓
several inputs for one piece of data	✗	✓	✓

Q Why do React Hooks make use of refs?

Earlier, refs were only limited to class components but now it can also be accessible in function components through the useRef Hook in React.

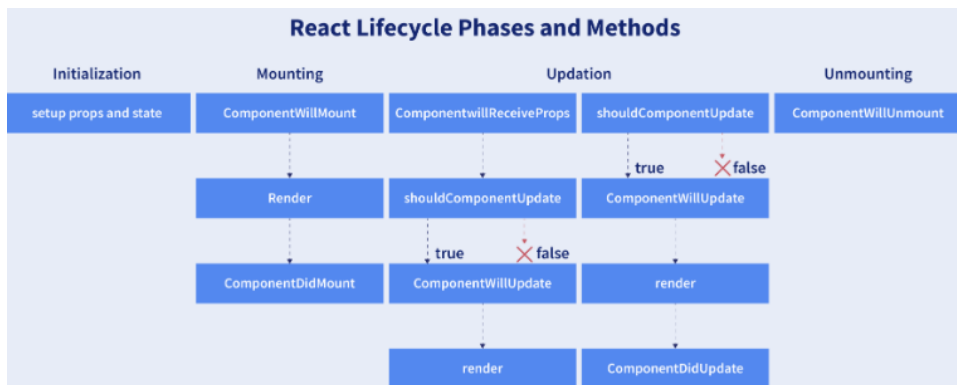
The refs are used for:

1. Managing focus, media playback, or text selection.
2. Integrating with DOM libraries by third-party.
3. Triggering the imperative animations.

Q When do we need a Higher Order Component?

While developing React applications, we might develop components that are quite similar to each other with minute differences. In most cases, developing similar components might not be an issue but, while developing larger applications we need to keep our code **DRY**, therefore, we want an **abstraction** that allows us to define this logic in a single place and share it across components. HOC allows us to create that abstraction

Q. What are the different phases of the component lifecycle?



Q Does React Hook work with static typing?

Static typing refers to the process of code check during the time of compilation for ensuring all variables will be statically typed. React Hooks are functions that are designed to make sure about all attributes must be statically typed. For enforcing stricter static typing within our code, we can make use of the React API with custom Hooks.

Q Differentiate React Hooks vs Classes.

React Hooks	Classes
It is used in functional components of React.	It is used in class-based components of React.
It will not require a declaration of any kind of constructor.	It is necessary to declare the constructor inside the class component.
It does not require the use of <code>this</code> keyword in state declaration or modification.	Keyword <code>this</code> will be used in state declaration (<code>this.state</code>) and in modification (<code>this.setState()</code>).
It is easier to use because of the <code>useState</code> functionality.	No specific function is available for helping us to access the state and its corresponding <code>setState</code> variable.
React Hooks can be helpful in implementing Redux and context API.	Because of the long setup of state declarations, class states are generally not preferred.

Q. What is React Router?

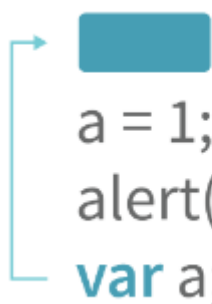
React Router refers to the standard library used for routing in React. It permits us for building a single-page web application in React with navigation without even refreshing the page when the user navigates. It also allows to change the browser URL and will keep the user interface in sync with the URL. React Router will make use of the component structure for calling the components, using which appropriate information can be shown. Since React is a component-based framework, it's not necessary to include and use this package. Any other compatible routing library would also work with React.

The major components of React Router are given below:

1. **BrowserRouter:** It is a router implementation that will make use of the HTML5 history API (pushState, popstate, and event replaceState) for keeping your UI to be in sync with the URL. It is the parent component useful in storing all other components.
2. **Routes:** It is a newer component that has been introduced in the React v6 and an upgrade of the component.
3. **Route:** It is considered to be a conditionally shown component and some UI will be rendered by this whenever there is a match between its path and the current URL.
4. **Link:** It is useful in creating links to various routes and implementing navigation all over the application. It works similarly to the anchor tag in HTML.

Q Explain Hoisting in javascript.

Hoisting is the default behaviour of javascript where all the variable and function declarations are moved on top. This means that irrespective of where the variables and functions are declared, they are moved on top of the scope. The scope can be both local and global.



```
a = 1;  
alert(' a = ' + a);  
var a;
```

Q 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.

Q 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.

Q Difference between var and let keyword in javascript.

Some differences are:

From the very beginning, the 'var' keyword was used in JavaScript programming **whereas the keyword 'let'** was just added in 2015.

The keyword 'Var' has a function scope. Anywhere in the function, the variable specified using var is accessible but in 'let' the scope of a variable declared with the 'let' keyword is limited to the block in which it is declared. Let's start with a Block Scope.

In ECMAScript 2015, let and const are hoisted but not initialized. Referencing the variable in the block before the variable declaration results in a ReferenceError because the variable is in a "temporal dead zone" from the start of the block until the declaration is processed.

Q Explain Higher Order Functions in javascript.

Functions that operate on other functions, either by taking them as arguments or by returning them, are called higher-order functions.

```
function higherOrder(fn) {  
  fn();  
}  
  
higherOrder(function() { console.log("Hello world") });
```


Q. Explain “this” keyword.

The “this” keyword refers to the object that the function is a property of.

The value of the “this” keyword will always depend on the object that is invoking the function.

```
var obj = {  
  name: "vivek",  
  getName: function(){  
    console.log(this.name);  
  }  
}  
  
obj.getName();
```

In the above code, at the time of invocation, the getName function is a property of the object obj , therefore, this keyword will refer to the object obj, and hence the output will be “vivek”.

Q What do you mean by Self Invoking Functions?

Without being requested, a self-invoking expression is automatically invoked (initiated). If a function expression is followed by (), it will execute automatically. A function declaration cannot be invoked by itself.

Normally, we declare a function and call it, however, anonymous functions may be used to run a function automatically when it is described and will not be called again. And there is no name for these kinds of functions.

Q What are some advantages of using External JavaScript?

External JavaScript is the JavaScript Code (script) written in a separate file with the extension.js, and then we link that file inside the <head> or <body> element of the HTML file where the code is to be placed.

Some advantages of external javascript are

- i. It allows web designers and developers to collaborate on HTML and javascript files.
- ii. We can reuse the code.
- iii. Code readability is simple in external javascript.

Q What is memoization?

Memoization is a form of caching where the return value of a function is cached based on its parameters. If the parameter of that function is not changed, the cached version of the function is returned.

Q What is DOM?

1. DOM stands for Document Object Model. DOM is a programming interface for HTML and XML documents.
2. 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.

Q 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.

Q What are some of the advantages of MongoDB?

Some advantages of MongoDB are as follows:

1. MongoDB supports field, range-based, string pattern matching type queries. for searching the data in the database
2. MongoDB support primary and secondary index on any fields
3. MongoDB basically uses JavaScript objects in place of procedures
4. MongoDB uses a dynamic database schema
5. MongoDB is very easy to scale up or down
6. MongoDB has inbuilt support for data partitioning (Sharding).

Q What is a Collection in MongoDB?

A collection in MongoDB is a group of documents. If a document is the MongoDB analog of a row in a relational database, then a collection can be thought of as the analog to a table. Documents within a single collection can have any number of different "shapes.", i.e. collections have dynamic schemas.

Q **What are some features of MongoDB?**

1. **Indexing:** It supports generic secondary indexes and provides unique, compound, geospatial, and full-text indexing capabilities as well.
2. **Aggregation:** It provides an aggregation framework based on the concept of data processing pipelines.
3. **Special collection and index types:** It supports time-to-live (TTL) collections for data that should expire at a certain time
4. **File storage:** It supports an easy-to-use protocol for storing large files and file metadata.
5. **Sharding:** Sharding is the process of splitting data up across machines.

Q **How to perform queries in MongoDB?**

The find method is used to perform queries in MongoDB. Querying returns a subset of documents in a collection, from no documents at all to the entire collection. Which documents get returned is determined by the first argument to find, which is a document specifying the query criteria.

Example:

```
> db.users.find({"age" : 24})
```

Q **When to use MongoDB?**

You should use MongoDB when you are building internet and business applications that need to evolve quickly and scale elegantly. MongoDB is popular with developers of all kinds who are building scalable applications using agile methodologies.

MongoDB is a great choice if one needs to:

1. Support a rapid iterative development.
2. Scale to high levels of read and write traffic
3. Scale your data repository to a massive size.
4. Evolve the type of deployment as the business changes.
5. Store, manage and search data with text, geospatial, or time-series dimensions.

Q **What are MongoDB Charts?**

MongoDB Charts is a new, integrated tool in MongoDB for data visualization.

MongoDB Charts offers the best way to create visualizations using data from a MongoDB database.

It allows users to perform quick data representation from a database without writing code in a programming language such as Java or Python.

The two different implementations of MongoDB Charts are:

1. MongoDB Charts PaaS (Platform as a Service)
2. MongoDB Charts Server

Q What is a Replica Set in MongoDB?

To keep identical copies of your data on multiple servers, we use replication. It is recommended for all production deployments. Use replication to keep your application running and your data safe, even if something happens to one or more of your servers.

Such replication can be created by a replica set with MongoDB. A replica set is a group of servers with one primary, the server taking writes, and multiple secondaries, servers that keep copies of the primary's data. If the primary crashes, the secondaries can elect a new primary from amongst themselves.

Q What is Node.js and how it works?

Node.js is a virtual machine that uses JavaScript as its scripting language and runs Chrome's V8 JavaScript engine. Basically, Node.js is based on an event-driven architecture where I/O runs asynchronously making it lightweight and efficient. It is being used in developing desktop applications as well with a popular framework called electron as it provides API to access OS-level features such as file system, network, etc.

Q. Promises vs Callbacks.

The main advantage of using promise is you get an object to decide the action that needs to be taken after the async task completes. This gives more manageable code and avoids callback hell.

Q Why is Node.js single-threaded?

Node.js was created explicitly as an experiment in async processing. This was to try a new theory of doing async processing on a single thread over the existing thread-based implementation of scaling via different frameworks.

Q. How many types of API functions are there in Node.js?

There are two types of API functions:

1. **Asynchronous, non-blocking functions** - mostly I/O operations which can be fork out of the main loop.
2. **Synchronous, blocking functions** - mostly operations that influence the process running in the main loop.

Q If Node.js is single threaded then how does it handle concurrency?

The main loop is single-threaded and all async calls are managed by libuv library.

Q What is middleware?

Middleware comes in between your request and business logic. It is mainly used to capture logs and enable rate limit, routing, authentication, basically whatever that is not a part of business logic. There are third-party middleware also such as body-parser and you can write your own middleware for a specific use case.

Q. How can we use async await in node.js?

```
// this code is to retry with exponential backoff
function wait (timeout) {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve()
    }, timeout);
  });
}
async function requestWithRetry (url) {
  const MAX_RETRIES = 10;
  for (let i = 0; i <= MAX_RETRIES; i++) {
    try {
      return await request(url);
    } catch (err) {
      const timeout = Math.pow(2, i);
      console.log('Waiting', timeout, 'ms');
      await wait(timeout);
      console.log('Retrying', err.message, i);
    }
  }
}
```

Q. Why should you separate Express app and server?

The server is responsible for initializing the routes, middleware, and other application logic whereas the app has all the business logic which will be served by the routes initiated by the server. This ensures that the business logic is encapsulated and decoupled from the application logic which makes the project more readable and maintainable.

Q What is an Event Emitter in Node.js?

EventEmitter is a Node.js class that includes all the objects that are basically capable of emitting events. This can be done by attaching named events that are emitted by the object using an eventEmitter.on() function. Thus whenever this object throws an even the attached functions are invoked synchronously.

Q Enhancing Node.js performance through clustering.

Node.js applications run on a single processor, which means that by default they don't take advantage of a multiple-core system. Cluster mode is used to start up multiple node.js processes thereby having multiple instances of the event loop. When we start using cluster

in a nodejs app behind the scene multiple node.js processes are created but there is also a parent process called the **cluster manager** which is responsible for monitoring the health of the individual instances of our application.

Q **What is a thread pool and which library handles it in Node.js**

The Thread pool is handled by the libuv library. libuv is a multi-platform C library that provides support for asynchronous I/O-based operations such as file systems, networking, and concurrency.

Q. **What are some commonly used timing features of Node.js?**

1. **setTimeout/clearTimeout** – This is used to implement delays in code execution.
2. **setInterval/clearInterval** – This is used to run a code block multiple times.
3. **setImmediate/clearImmediate** – Any function passed as the setImmediate() argument is a callback that's executed in the next iteration of the event loop.
4. **process.nextTick** – Both setImmediate and process.nextTick appear to be doing the same thing; however, you may prefer one over the other depending on your callback's urgency.