JavaScript Interview Questions:

1. What is JavaScript, and what are its key features?
2. How does JavaScript differ from Java?
3. Explain the differences between "undefined" and "null" in JavaScript.
4. What are the different data types in JavaScript?
5. How does prototypal inheritance work in JavaScript?
6. Describe the event bubbling and event capturing mechanisms in JavaScript.
7. What is a closure in JavaScript, and how can it be used?
8. How does "this" keyword work in JavaScript?
9. What is the purpose of "use strict" in JavaScript?
10. Explain the concept of callbacks and how they are used in asynchronous JavaScript.

HTML & CSS Interview Questions:

1. What is the purpose of HTML and CSS?
2. What is the box model in CSS, and how does it affect the layout?
3. Explain the differences between "display: block", "display: inline", and "display: inline-block" in CSS.
4. How can you center an element horizontally and vertically using CSS?
5. What are media queries in CSS, and how do they enable responsive design?
6. What is the purpose of the "alt" attribute in HTML images?
7. How do you link an external CSS stylesheet to an HTML document?
8. What are the different CSS selectors and how do they work?
9. How can you create a fixed header that stays at the top of the page while scrolling?
10. What is the difference between margin and padding in CSS?

React.js Interview Questions:

1. What is React.js, and what are its main features?
2. Explain the concept of virtual DOM and how it helps improve performance in React.
3. What are React components, and how do you create them?
4. How can you pass data from a parent component to a child component in React?
5. What is the purpose of "state" in React, and how do you use it?
6. How can you handle events in React?
7. Explain the differences between controlled and uncontrolled components in React.
8. What are React hooks, and how do they differ from class components?
9. How can you perform routing in React applications?
10. What is the significance of "key" prop in lists in React?

MySQL Interview Questions:

1. What is MySQL, and what are its key features?
2. Explain the differences between "CHAR" and "VARCHAR" data types in MySQL.
3. How do you create a new database and a new table in MySQL?
4. What is the purpose of "PRIMARY KEY" and "INDEX" in MySQL tables?
5. How can you retrieve data from a MySQL database using SELECT queries?
6. What are the different types of joins in MySQL, and when would you use each one?
7. How can you prevent SQL injection in MySQL queries?
8. Explain the differences between "GROUP BY" and "HAVING" clauses in MySQL.
9. How do you add, update, and delete records in a MySQL table?
10. What are stored procedures and how can you use them in MySQL?

MongoDB Interview Questions:

1. What is MongoDB, and how is it different from traditional SQL databases?
2. What is a document in MongoDB?
3. How do you create a new database and a new collection in MongoDB?
4. Explain the differences between "find" and "findOne" methods in MongoDB.
5. How can you perform filtering and sorting of data in MongoDB?
6. What are indexes in MongoDB, and why are they important?
7. How can you perform aggregation operations in MongoDB?
8. What is the purpose of "mongodump" and "mongorestore" commands in MongoDB?
9. How do you perform data backups in MongoDB?
10. Explain the concept of sharding in MongoDB and how it helps with scalability.

Node.js Interview Questions:

1. What is Node.js, and why is it popular for server-side development?
2. How does Node.js handle asynchronous operations?
3. What is the purpose of the "npm" package manager in Node.js?
4. How can you handle file system operations in Node.js?
5. What are modules in Node.js, and how do you export and import them?
6. How can you create a simple HTTP server using Node.js?
7. What is middleware in the context of Node.js web frameworks like Express?
8. Explain the differences between "callback", "promises", and "async/await" in Node.js.
9. How can you handle errors in Node.js applications?
10. What is clustering in Node.js, and how can it improve performance?

1. Prototype:

Each object has a private property which holds a

link to another object called its prototype.

2. The popular 'this' keyword points to the gobal window object in the

context of browsers when written inside a regualr function call. The

'this' keyword points to the object to which the method is bond.

3. The methods - bind(), call(), apply() are used to bind a function while

passing the desired scope to the inner function.

4. Block elements:

<div>, <p> - always starts on a new line

browser add some space before and after the element

Always takes up the full available width

5. Inline elements:

<span> - does not start on a new line

Always takes up as much width as neccessary

6. Table:

<table> - table

<tr> - table row

<th> - table header

<td> - table data

7. inline:

no line break without space

8. block:

line break with space

9. inline-block:

no line break with space

10. Pseudo-classes:

A pseudo-class is used to define a special state of an element.

11. Position property in CSS:

Static(default), Relative(normal flow), absolute(depends on parent),

fixed(absolute), sticky(relative)

12. Mobile First Approach/Design:

It is easier to scale a design up to larger screens than to scale

it down to smaller screens. Due to the massive growth of Mobile

browsing, this makes mobile first design a great development

workflow.

13. Bootstrap is a free and open-source CSS framework directed at responsive,

mobile-first front-end web development. It contains HTML, CSS and

JavaScript-based design templates for typography, forms, buttons,

navigation, and other interface components.

Container: The container is the fundamental building block of the Bootstrap

layout. It is used to center and align your content on the page,

providing a responsive and fluid design.

14. DOM - Document Object Model

The programming interface for interacting with an HTML document represented

as a tree data structure. Each HTML element in the document is a node in

the DOM tree, with nested content represented as children in the tree.

--> provides an API

--> Tree structure

--> Elements & Text are nodes

append --> Node + Text

appendChild --> Only Node object

15. JavaScript is a single-threaded synchronous language which executes all the code

line by line in the order in which it is written.

16. In Asynchronous programming, you can do taks without waiting for a task to

complete its execution.

17. Callback - pass a function as an argument to another function. The primary purpose

of a callback is to execute code in response to an event.

18. Callback Hell: nested callback is called callback hell

19. Callback serve a great purpose in asynchronous programming,

There are 2 cases in which Callback is used,

a. Mark completion of a long-running task.

b. Execute a task after a long-running task.

20. async & await - keywords introduced in ES8(ES2017), which are

internally based on promises but makes the code more readable as

compared to promises.

21. When the keyword 'async' is prepended to a function, it can before

safely assumed that a promise is returned from that function.

22. Declaring variables with two new keywords - let and const. The former is used when you wish to change the value of the variable in future and the latter is used when you do not wish to change the value of the variable. Unlike var, the variables declared using let/const keywords are not hoisted.

23. Classes, which are syntactic sugar over JavaScript's existing functions and help in achieving what is so-called prototype-based inheritance in ES5.

24. Arrow functions, which is a new syntax over traditional functions and are used to write code quickly and more effectively. They also solve some of the problems, as we see with the context w.r.t. 'this' keyword.

25. Array methods - map(), filter(), reduce() methods, which are used to iterate over an array and is a much simpler and quicker way to perform operations on an array as compared to its alternative methods.

26. The powerful three dots - referred to as the spread operator as well as the rest parameters, depending upon the context in which they are used. When used as spread operator, they are used to segregate an array into its individual elements. When used as the rest parameters, they are used to treat individual arguments passed to a function as a combined array in the function parameters.

27. Template Literals, which are better way of concatenating string literals.

28. Ways of destructuring an array or an object to extract the its individual elements or keys.

29. import and export keywords - used for importing and exporting modules (including variables, functions, classes, etc.) defined in one file to be used in another file so as to maintain code reusability and reduce redundancy.  
  
  
  
JSX:

1. Adjacent JSX elements must be wrapped in an enclosing tag

2. Closing tag required

3. JSX properties are not similar to HTML attributes

4. Case Sensitiveness

5. Injecting Data using curly braces {}

Folder Structure:

Following are the basic points to note about some files that you see inside your application folder:

1 .gitignore file

It is used by Git to determine which files and directories to ignore before a commit is made.

It should be committed into the repository to share ignore rules with other users who clone the repository.

The node\_modules  folder is included inside the .gitignore file so that the user who clones the application is not required to clone this folder. The user simply needs to run the command

npm install

in the root folder of the project. This command creates the node\_modules folder and installs all the dependencies (packages)

needed for the application.

2. package.json file

It consists of the name and version of the application, the combination of which should be unique in order to publish the package.

It comprises of dependencies that list all the packages needed to be installed for the application.

It also includes scripts that specify the commands to be run at various points in the application lifecycle.

3. package-lock.json file

It is automatically generated for any operation where npm modifies either the node\_modules  tree or the package.json  file.

It locks the version of the full dependency tree of packages.

It guarantees the generation of an identical dependency tree when the application is cloned by other developers.

4. node\_modules folder

Its contents are defined by the package.json file and it consists of all the packages required for running your application.

5. public folder

Nothing inside this folder is processed by Webpack.

It is used for keeping small files that are not required to be bundled.

It can be used to contain images when there are thousands of them, and their paths need to be referenced dynamically.

Any file inside this folder needs to be referenced at other places using the %PUBLIC\_URL%/  keyword, which gets replaced with the path of the public folder during the application's build process.

6. index.html file

It is the starting point of the application.

It should always remain with the name index.html and inside the public folder; otherwise, the code will fail to run.

It can only reference files that are inside the public  folder.

7. manifest.json file

It is a simple JSON file telling the browser about the web application and how the application should behave when it is installed on the user’s mobile device or computer.

8. src folder

It consists of the real application code.

It consists of all the files that are needed to get bundled by Webpack.

9. index.js file

It is the entry point for JavaScript.

The filename should remain index.js and the location should be inside the src folder; otherwise, the code will not run.

10. index.css file

It is the stylesheet for index.html.

11. App.js file

It is the JavaScript file for the App component.

12. App.css file

It is the stylesheet for the App component.

13. App.test.js file

It is the test file for the App component.

It contains unit tests for the application.

It runs test cases for all the files that changed since the last commit of the application.

14. logo.svg file

SVG is an acronym for Scalable Vector Graphics.

An SVG file is an XML-based vector image format for 2D graphics with support for interactivity and animation.

It is similar to raster-based image formats such as JPEG, PNG, BMP, GIF, etc.

It offers a bandwidth-friendly way of rendering images; no matter how large a graphic gets, it transmits only the XML describing the graphic to the client.

It helps to render resolution-independent and SEO-friendly images.

It makes up the icon for your application and appears alongside the title in the browser tab.

It gets saved along with the bookmark.

Component:

1. A Component's name must start with a capital letter

2. A class component must extend from a base/parent class named component

3. A component must always return something

4. export and import

5. A component can have a extension as .js or .JSX

6. In import statement, .jsx or .js extension is not required

7. Independant & re-use

Props:

1. Props help you to pass values from a parent component to child component

so that they can be accessed within the child component.

Event handling:

1. An event is an action to be taken as a result of user interactions.

2. An event handler is a method to be called when an event occurs.

State:

1. State is something that is controlled within a component, unlike Props,

which are controlled by a parent component. Also, a change in state calls the

render() method again.

2. State can be maintained inside a calss component only.

3. State is always initialised inside the class constructor.

4. In case you define the constructor of a class, you need to call

the super() method in the first statement of the constructor definition.

This method calls the constructor of the parent class.

5. To set the state, you must always use the setState() method and never

directly manipulate the application's state. However, setState() should never be called

inside the constructor.

Hooks:

1. Hooks allow functional components to use React features like state etc.

2. Hooks can be used only inside React function components.

3. Hooks cannot be used inside conditional(if, for loop) logics

4. Hooks can only be called at the top level of the components.

Routing:

1. Routing is the process that helps in loading partial content.

2. Based on the URL that a user visits, specific content is loaded on the page.

3. It helps in displaying different content to the user without any need for the page refresh.

useParams():

1. It is used when we need to access the parameters of the current route.

Switch/Routes:

1. Switch/Routes is used to render components only when the path will be matched.

Context API:

1. It allows data to be passed through a component tree without having to pass

props manually at every level.

--> Introduction

--> Why use the context API?

--> Creating a context API

--> Creating a Context Provider

--> Creating context provide components

--> Drawbacks of Context API - When the context value changes, all the components

that use the same context will re-render. This can cause performance issue

in larger applications.

Redux:

1. Redux is a state management library.

2. It provides a predictable way to manage the state of your application

by centralizing it in a single store.

3. Redux uses a unidirectional data flow - means that data flows in one direction

only, from the store to the components.

Redux is based on three core principles:

--> Single soure of truth: The state of your whole application is stored

in a single object tree within a single store.

--> State is read-only

--> Change are made with pure functions

Redux vs Context API:

1. Approach is different

2. Redux is centralized, whereas Context API decentralized

3. Another main difference between Redux and Context API is

their PERFORMANCE. Redux can be more performant, especially

when dealing with large and complex application because

it minimizes the no of state updates.

4. On other hand Context API is simpler to use, we can use this

in smaller and medium sized application.