TOP 30

REACT JS

INTERVIEW QUESTION

Created by- TOPPERWORLD

Q 1. What is React?

Ans : React is a front-end and open-source JavaScript library which is useful in developing user interfaces specifically for applications with a single page. It is helpful in building complex and reusable user interface(UI) components of mobile and web applications as it follows the component-based approach.

The important features of React are:

1. It supports server-side rendering.
2. It will make use of the virtual DOM rather than real DOM (Data Object Model) as RealDOM manipulations are expensive.
3. It follows unidirectional data binding or data flow.
4. It uses reusable or composable UI components for developing the view.

Q 2. What are the advantages of using React?

Ans : MVC is generally abbreviated as Model View Controller.

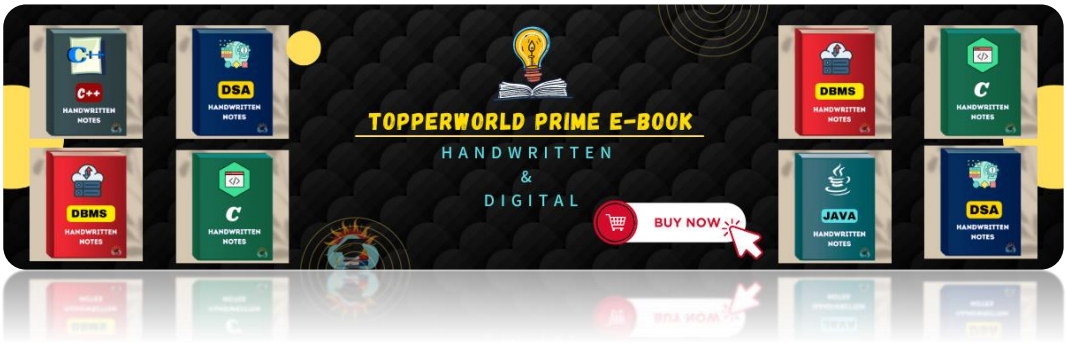
- **Use of Virtual DOM to improve efficiency:** React uses virtual DOM to render the view. As the name suggests, virtual DOM is a virtual representation of the real DOM. Each time the data changes in a react app, a new virtual DOM gets created. Creating a virtual DOM is much faster than rendering the UI inside the browser. Therefore, with the use of virtual DOM, the efficiency of the app improves.
- **Gentle learning curve:** React has a gentle learning curve when compared to frameworks like Angular. Anyone with little knowledge of javascript can start building web applications using React.
- **SEO friendly:** React allows developers to develop engaging user interfaces that can be easily navigated in various search engines. It also allows server-side rendering, which boosts the SEO of an app.
- **Reusable components:** React uses component-based architecture for developing applications. Components are independent and reusable bits of code. These components can be shared across various applications having similar functionality. The re-use of components increases the pace of development.

➢ **Huge ecosystem of libraries to choose from:** React provides you with the freedom to choose the tools, libraries, and architecture for developing an application based on your requirement.

Q 3. What are the limitations of React?

Ans : The few limitations of React are as given below:

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



Q 4. What is useState() in React?

Ans : 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.

```
import React, { useState } from 'react';


function Counter() {
  // Define state variable count and its setter function setCount
  const [count, setCount] = useState(0);


  return (
    <div>
      <p>You clicked {count} times</p>
      {/* Button to increment count */}
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
export default Counter;
```

Q 5. What are keys in React?

Ans : A key is a special string attribute that needs to be included when using lists of elements.

```
import React from 'react';

function List() {
 const items = [
  { id: 1, name: 'Item 1' },
  { id: 2, name: 'Item 2' },
  { id: 3, name: 'Item 3' }
 ];

 return (
  <div>
   <h2>List of Items</h2>
   <ul>
    {items.map(item => (
     // Assigning a unique key prop to each list item
     <li key={item.id}>{item.name}</li>
    ))}
   </ul>
  </div>
 );
}
export default List;
```

Importance of keys -

1. Keys help react identify which elements were added, changed or removed.

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

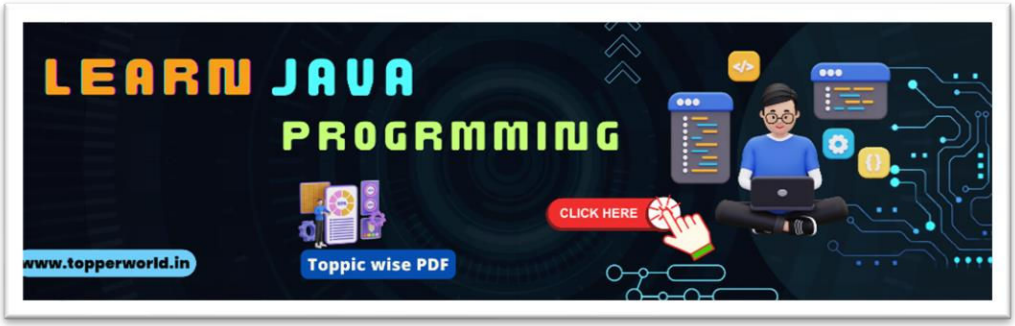
Q 6. What is JSX?

Ans : JSX stands for JavaScript XML. It allows us to write HTML inside JavaScript and place them in the DOM without using functions like appendChild( ) or createElement( ).

As stated in the official docs of React, JSX provides syntactic sugar for React.createElement( ) function.

Let ' s understand **how JSX works**:

Without using JSX, we would have to create an element by the following process:




```
const text = React.createElement('p', {}, 'This is a text');

const container = React.createElement('div','{}',text );

ReactDOM.render(container,rootElement);
```

Using JSX, the above code can be simplified:

```
const container = (
<div>
  <p>This is a text</p>
</div>
);
```

Q 7. What are the differences between functional and class components?

Ans :

Functional Components	Class Components
A functional component is just a plain JavaScript pure function that accepts props as an argument and returns a React element(JSX).	A class component requires you to extend from React. Component and create a render function that returns a React element.
There is no render method used in functional components.	It must have the render() method returning JSX (which is syntactically similar to HTML)
Functional components run from top to bottom and once the function is returned it can't be kept alive.	The class component is instantiated and different life cycle method is kept alive and is run and invoked depending on the phase of the class component.
Also known as Stateless components as they simply accept data and display them in some form, they are mainly responsible	Also known as Stateful components because they implement logic and state.

for rendering UI.	
React lifecycle methods (for example, componentDidMount) cannot be used in functional components.	React lifecycle methods can be used inside class components (for example, componentDidMount).

Hooks can be easily used in functional components to make them Stateful.
Example:
`const [name,SetName]= React.useState(' ')`
	It requires different syntax inside a class component to implement hooks.
	Example:
	`constructor(props) {`
	`  super(props);`
	`  this.state = {name: ' '}`
	`}`

Constructors are not used.	Constructor is used as it needs to store state.

Q 8. What are the lifecycle methods of React?

Ans : React lifecycle hooks will have the methods that will be automatically called at different phases in the component lifecycle and thus it provides good control over what happens at the invoked point. It provides the power to effectively control and manipulate what goes on throughout the component lifecycle.

For example, if you are developing the YouTube application, then the application will make use of a network for buffering the videos and it consumes the power of the battery (assume only these two). After playing the video if the user switches to any other application, then you should make sure that the resources like network and battery are being used most efficiently.

You can stop or pause the video buffering which in turn stops the battery and network usage when the user switches to another application after video play.

So we can say that the developer will be able to produce a quality application with the help of lifecycle methods and it also helps developers to make sure to plan what and how to do it at different points of birth, growth, or death of user interfaces.

The various lifecycle methods are:

1) **constructor():** This method will be called when the component is initiated before anything has been done. It helps to set up the initial state and initial values.
2) **getDerivedStateFromProps():** This method will be called just before element(s) rendering in the DOM. It helps to set up the state object depending on the initial props. The getDerivedStateFromProps() method will have a state as an argument and it returns an object that made changes to the state. This will be the first method to be called on an updating of a component.
3) **render():** This method will output or re-render the HTML to the DOM with new changes. The render() method is an essential method and will be called always while the remaining methods are optional and will be called only if they are defined.
4) **componentDidMount():** This method will be called after the rendering of the component. Using this method, you can run statements that need the component to be already kept in the DOM.
5) **shouldComponentUpdate():** The Boolean value will be returned by this method which will specify whether React should proceed further with the rendering or not. The default value for this method will be True.
6) **getSnapshotBeforeUpdate():** This method will provide access for the props as well as for the state before the update. It is possible to check the previously present value before the update, even after the update.
7) **componentDidUpdate():** This method will be called after the component has been updated in the DOM.
8) **componentWillUnmount():** This method will be called when the component removal from the DOM is about to happen.

Q 9. Does React Hook work with static typing?

Ans : 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 10. Explain about types of Hooks in React.

Ans : There are two types of Hooks in React. They are:

1. Built-in Hooks: The built-in Hooks are divided into 2 parts as given below:

Basic Hooks:

- useState(): This functional component is used to set and retrieve the state.
- useEffect(): It enables for performing the side effects in the functional components.
- useContext(): It is used for creating common data that is to be accessed by the components hierarchy without having to pass the props down to each level.

Additional Hooks:

- useReducer() : It is used when there is a complex state logic that is having several sub-values or when the upcoming state is dependent on the previous state. It will also enable you to

optimization of component performance that will trigger deeper updates as it is permitted to pass the dispatch down instead of callbacks.

- useMemo() : This will be used for recomputing the memoized value when there is a change in one of the dependencies. This optimization will help for avoiding expensive calculations on each render.
- useCallback() : This is useful while passing callbacks into the optimized child components and depends on the equality of reference for the prevention of unneeded renders.
- useImperativeHandle(): It will enable modifying the instance that will be passed with the ref object.
- useDebugValue(): It is used for displaying a label for custom hooks in React DevTools.
- useRef() : It will permit creating a reference to the DOM element directly within the functional component.
- useLayoutEffect(): It is used for the reading layout from the DOM and re-rendering synchronously.

2. Custom Hooks: A custom Hook is basically a function of JavaScript. The Custom Hook working is similar to a regular function. The "use" at the beginning of the Custom Hook Name is required for React to understand that this is a custom Hook and also it will describe that this specific function follows the rules of Hooks. Moreover, developing custom Hooks will enable you for extracting component logic from within reusable functions.

Q 11. Differentiate React Hooks vs Classes.

Ans :

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 this keyword in state declaration or modification.	Keyword this will be used in state declaration (this.state) and in modification (this.setState()).
It is easier to use because of the useState functionality.	No specific function is available for helping us to access the state and its corresponding setState 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 12. How does the performance of using Hooks will differ in comparison with the classes?

Ans : React Hooks will avoid a lot of overheads such as the instance creation, binding of events, etc., that are present with classes.

Hooks in React will result in smaller component trees since they will be avoiding the nesting that exists in HOCs (Higher Order Components) and will render props which result in less amount of work to be done by React.

Q 13. Do Hooks cover all the functionalities provided by the classes?

Ans : Our goal is for Hooks to cover all the functionalities for classes at its earliest. There are no Hook equivalents for the following methods that are not introduced in Hooks yet:

1) getSnapshotBeforeUpdate()
2) getDerivedStateFromError()
3) componentDidCatch()

Since it is an early time for Hooks, few third-party libraries may not be compatible with Hooks at present, but they will be added soon.

Q 14. What is React Router?

Ans : 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:

- **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.
- **Routes:** It is a newer component that has been introduced in the React v6 and an upgrade of the component.
- **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.

- **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 15. Can React Hook replaces Redux?

Ans : The React Hook cannot be considered as a replacement for Redux (It is an open-source, JavaScript library useful in managing the application state) when it comes to the management of the global application state tree in large complex applications, even though the React will provide a useReducer hook that manages state transitions similar to Redux. Redux is very useful at a lower level of component hierarchy to handle the pieces of a state which are dependent on each other, instead of a declaration of multiple useState hooks.

In commercial web applications which is larger, the complexity will be high, so using only React Hook may not be sufficient. Few developers will try to tackle the challenge with the help of React Hooks and others will combine React Hooks with the Redux.

Q 16. Explain conditional rendering in React.

Ans : Conditional rendering refers to the dynamic output of user interface markups based on a condition state. It works in the same way as JavaScript conditions. Using conditional rendering, it is possible to toggle specific application functions, API data rendering, hide or show elements, decide permission levels, authentication handling, and so on.

There are different approaches for implementing conditional rendering in React. Some of them are:

- Using if-else conditional logic which is suitable for smaller as well as for medium-sized applications
- Using ternary operators, which takes away some amount of complication from if-else statements

- Using element variables, which will enable us to write cleaner code.

Q 17. Explain how to create a simple React Hooks example program.

Ans : I will assume that you are having some coding knowledge about JavaScript and have installed Node on your system for creating a below given React Hook program. An installation of Node comes along with the command-line tools: npm and npx, where npm is useful to install the packages into a project and npx is useful in running commands of Node from the command line. The npx looks in the current project folder for checking whether a command has been installed there. When the command is not available on your computer, the npx will look in the npmjs.com repository, then the latest version of the command script will be loaded and will run without locally installing it. This feature is useful in creating a skeleton React application within a few key presses.

Open the Terminal inside the folder of your choice, and run the following command:

```
npx create-react-app react-items-with-hooks
```

Here, the create-react-app is an app initializer created by Facebook, to help with the easy and quick creation of React application, providing options to customize it while creating the application?

The above command will create a new folder named react-items-with-hooks and it will be initialized with a basic React application. Now, you will be able to open the project in your favourite IDE. You can see an src folder inside the project along with the main application component App.js. This file is having a single function App() which will return an element and it will make use of an extended JavaScript syntax(JSX) for defining the component.

JSX will permit you for writing HTML-style template syntax directly into the JavaScript file. This mixture of JavaScript and HTML will be converted by React toolchain into pure JavaScript that will render the HTML element.

It is possible to define your own React components by writing a function that will return a JSX element.

You can try this by creating a new file src/SearchItem.jsand put the following code into it.

```
import React from 'react';
export function SearchItem() {
 return (
  <div>
   <div className="search-input">
    <input type="text" placeholder="SearchItem"/>
   </div>
   <h1 className="h1">Search Results</h1>
   <div className="items">
    <table>
     <thead>
      <tr>
       <th className="itemname-col">Item Name</th>
       <th className="price-col">Price</th>
       <th className="quantity-col">Quantity</th>
      </tr>
     </thead>
    </table>
   </div>
  </div>
 );
}
```

Q 18. How to create a switching component for displaying different pages?

Ans : A switching component refers to a component that will render one of the multiple components. We should use an object for mapping prop values to components.A below-given example will show you how to display different pages based on page prop using switching component:

```
import HomePage from './HomePage'
import AboutPage from './AboutPage'
import FacilitiesPage from './FacilitiesPage'
import ContactPage from './ContactPage'
import HelpPage from './HelpPage'
const PAGES = {
 home: HomePage,
 about: AboutPage,
 facilitiess: FacilitiesPage,
 contact: ContactPage
 help: HelpPage
}
const Page = (props) => {
 const Handler = PAGES[props.page] || HelpPage
 return <Handler {...props} />    }
// The PAGES object keys can be used in the prop types for catching errors during dev-time.
Page.propTypes = {
 page: PropTypes.oneOf(Object.keys(PAGES)).isRequired
}
```

Q 19. How to re-render the view when the browser is resized?

Ans : It is possible to listen to the resize event in **componentDidMount()** and then update the width and height dimensions.

It requires the removal of the event listener in the **componentWillUnmount()** method.

Using the below-given code, we can render the view when the browser is resized.

```
class WindowSizeDimensions extends React.Component {
 constructor(props){
  super(props);
  this.updateDimension = this.updateDimension.bind(this);
 }

 componentWillMount() {
  this.updateDimension()
 }

 componentDidMount() {
  window.addEventListener('resize',
this.updateDimension)
 }

 componentWillUnmount() {
  window.removeEventListener('resize',
this.updateDimension)
 }
```

```
 updateDimension() {
  this.setState({width: window.innerWidth, height:
window.innerHeight})
 }

render() {
  return <span>{this.state.width} x
{this.state.height}</span>
 }
}
```

Q 20. What are Custom Hooks?

Ans : A Custom Hook is a function in Javascript whose name begins with 'use' and which calls other hooks. It is a part of React v16.8 hook update and permits you for reusing the stateful logic without any need for component hierarchy restructuring.

In almost all of the cases, custom hooks are considered to be sufficient for replacing render props and HoCs (Higher-Order components) and reducing the amount of nesting required. Custom Hooks will allow you for avoiding multiple layers of abstraction or wrapper hell that might come along with Render Props and HoCs.

The **disadvantage** of Custom Hooks is it cannot be used inside of the classes.

Q 21. Name a few techniques to optimize React app performance.

Ans : There are many ways through which one can optimize the performance of a React app, let's have a look at some of them:

- **Using useMemo( ) -**
  - It is a React hook that is used for caching CPU-Expensive functions.

- Sometimes in a React app, a CPU-Expensive function gets called repeatedly due to re-renders of a component, which can lead to slow rendering.
  useMemo( ) hook can be used to cache such functions. By using useMemo( ), the CPU-Expensive function gets called only when it is needed.

➢ Using React.PureComponent -

- It is a base component class that checks the state and props of a component to know whether the component should be updated.
- Instead of using the simple React.Component, we can use React.PureComponent to reduce the re-renders of a component unnecessarily.

➢ Maintaining State Colocation -

- This is a process of moving the state as close to where you need it as possible.
- Sometimes in React app, we have a lot of unnecessary states inside the parent component which makes the code less readable and harder to maintain. Not to forget, having many states inside a single component leads to unnecessary re-renders for the component.
- It is better to shift states which are less valuable to the parent component, to a separate component.

➢ Lazy Loading -

- It is a technique used to reduce the load time of a React app. Lazy loading helps reduce the risk of web app performances to a minimum.

Q 22. What are the different phases of the component lifecycle?

Ans : There are four different phases in the lifecycle of React component. They are:

1) **Initialization:** During this phase, React component will prepare by setting up the default props and initial state for the upcoming tough journey.
2) **Mounting:** Mounting refers to putting the elements into the browser DOM. Since React uses VirtualDOM, the entire browser DOM which has been currently rendered would not be refreshed. This phase includes the lifecycle methods componentWillMount and componentDidMount.
3) **Updating:** In this phase, a component will be updated when there is a change in the state or props of a component. This phase will have lifecycle methods like componentWillUpdate, shouldComponentUpdate, render, and componentDidUpdate.
4) **Unmounting:** In this last phase of the component lifecycle, the component will be removed from the DOM or will be unmounted from the browser DOM. This phase will have the lifecycle method named componentWillUnmount.

Q 23. Explain the building blocks of React?

Ans : The five main building blocks of React are:

1) **Components:** These are reusable blocks of code that return HTML.
2) **JSX:** It stands for JavaScript and XML and allows you to write HTML in React.
3) **Props and State:** props are like function parameters and State is similar to variables.
4) **Context:** This allows data to be passed through components as props in a hierarchy.
5) **Virtual DOM:** It is a lightweight copy of the actual DOM which makes DOM manipulation easier.

Q 24. Explain props and state in React with differences

Ans : Props are used to pass data from one component to another. The state is local data storage that is local to the component only and cannot be passed to other components.

Here is the difference table of props and state In react

PROPS	STATE
The Data is passed from one component to another.	The Data is passed within the component only.
It is Immutable (cannot be modified).	It is Mutable (can be modified).
Props can be used with state and functional components.	The state can be used only with the state components/class component (Before 16.0).
Props are read-only.	The state is both read and write.

Q 25. What is the difference between useref and createRef in React ?

Ans : Here is the difference table of useref and createRef in React

useRef	createRef
It is a hook.	It is a function.
It uses the same ref throughout.	It creates a new ref every time.
It saves its value between re-	It creates a new ref for every re-

useRef	createRef
renders in a functional component.	render.
It returns a mutable ref object.	It returns a read-only ref object.
The refs created using the useRef can persist for the entire component lifetime.	The refs created using the createRef can be referenced throughout the component.
It is used in functional components.	It is used in class components.

Q 26. What is react-redux?

Ans : React-redux is a state management tool which makes it easier to pass these states from one component to another irrespective of their position in the component tree and hence prevents the complexity of the application. As the number of components in our application increases it becomes difficult to pass state as props to multiple components. To overcome this situation we use react-redux.

Q 27. What are benefits of using react-redux?

Ans : They are several benfits of using react-redux such as:

- It provides centralized state management i.e. a single store for whole application
- It optimizes performance as it prevents re-rendering of component
- Makes the process of debugging easier
- Since it offers persistent state management therfore storing data for long times become easier.

Q 28. Explain the core components of react-redux?

Ans : There are four fundamental concepts of redux in react which decide how the data will flow through components

- Redux Store: It is an object that holds the application state
- Acrtion Creators: These are unctions that return actions (objects)
- Actions: Actions are simple objects which conventionally have two properties- type and payload
- Reducers: Reducers are pure functions that update the state of the application in response to actions

Q 29. How can we combine multiple reducers in React?

Ans : When working with Redux we sometimes require multiple reducers. In many cases, multiple actions are needed, resulting in the requirement of multiple reducers. However, this can become problematic when creating the Redux store. To manage the multiple reducers we have function called combineReducers in the redux. This basically helps to combine multiple reducers into a single unit and use them.

Q 30. What is context API?

Ans : Context API is used to pass global variables anywhere in the code. It helps when there is a need for sharing state between a lot of nested components. It is light in weight and easier to use, to create a context just need to call React.createContext(). It eliminates the need to install other dependencies or third-party libraries like redux for state management. It has two properties Provider and Consumer.

ABOUT US

At TopperWorld, we are on a mission to empower college students with the knowledge, tools, and resources they need to succeed in their academic journey and beyond.

➢ Our Vision

- ❖ Our vision is to create a world where every college student can easily access high-quality educational content, connect with peers, and achieve their academic goals.
- ❖ We believe that education should be accessible, affordable, and engaging, and that's exactly what we strive to offer through our platform.

➢ Unleash Your Potential

- ❖ In an ever-evolving world, the pursuit of knowledge is essential. TopperWorld serves as your virtual campus, where you can explore a diverse array of online resources tailored to your specific college curriculum.
- ❖ Whether you're studying science, arts, engineering, or any other discipline, we've got you covered.
- ❖ Our platform hosts a vast library of e-books, quizzes, and interactive study tools to ensure you have the best resources at your fingertips.

➢ The TopperWorld Community

- ❖ Education is not just about textbooks and lectures; it's also about forming connections and growing together.

- ToppperWorld encourages you to engage with your fellow students, ask questions, and share your knowledge.
- We believe that collaborative learning is the key to academic success.

➢ Start Your Journey with TopperWorld

- Your journey to becoming a top-performing college student begins with TopperWorld.
- Join us today and experience a world of endless learning possibilities.
- Together, we'll help you reach your full academic potential and pave the way for a brighter future.
- Join us on this exciting journey, and let's make academic success a reality for every college student.

"UNLOCK YOUR POTENTIAL"

With- TOPPERWORLD

Explore More

topperworld.in

DSA TUTORIAL

C TUTORIAL

C++ TUTORIAL

JAVA TUTORIAL

PYTHON TUTORIAL

Follow Us On

E-mail

topperworld.in@gmail.com