Blog

ReactJS Interview Questions

★★★★★(5.0) | 27559 Ratings

Last Updated: 18.04.2018

If you're looking for ReactJS Interview Questions and Answers for Experienced & Freshers, you are at right place. There are lot of opportunities from many reputed companies in the world. According to research, ReactJS has a market share of about 7.3%. So, You still have opportunities to move ahead in your career in ReactJS. Mindmajix offers advanced ReactJS Interview Questions 2018 that helps you in cracking your interview & acquire your dream career as ReactJS Developer.

ReactJS Vs EmberJS	
ReactJS	Ember.js

Used to build user interfaces to render performance	Web Applications development
2013 - Open Sourced	Initially as SproutCore in 2007, later in 2011 it is made available as Ember.JS
Large web applications whose data is set to change frequently	Dynamic SPAs
Good code reuse.	Performance focus.
Easy interface design and learning API.	Friendly docs and API.
Fully component based architecture.	Convention over configuration.

Q: What is React?

React is an open-source JavaScript (https://mindmajix.com/javascript-training) library developed by Facebook Engineers for building complex and interactive User Interfaces in web & mobile applications (https://mindmajix.com/mobile-application-testing-training).

Q: How is React different?

Since React is a little small concentrated on building UI components, it is essentially not the same as a considerable measure of other JavaScript frameworks.

For instance, AngularJS (1.x) approaches building an application by expanding HTML markup and infusing different develops (e.g. Directives, Controllers, Services) at runtime. Subsequently, AngularJS is extremely obstinate about the more architecture design of your application — these reflections are absolutely helpful now and again, yet much of the time, they come at the cost of adaptability.

Q: Why ReactJS is used?

React is used to handle the view part of Mobile application and Web application (https://mindmajix.com/web-application-development-courses).

Q: How ReactJS framework is different as compared to others?

Basically, ReactJS is a limited library that builds UI parts, it is essentially not quite the same as a considerable measure of other JavaScript structures. One common example is AngularJS approaches building an app simply by expanding HTML markup and infusing different develop such as controller at runtime. Therefore, AngularJS is exceptionally obstinate about the more noteworthy engineering of your application.

Q: Does ReactJS use HTML?

No, It uses JSX which is simiar to HTM.

Q: What do you know about the component lifecycle in ReactJS?

Component lifecycle is an essential part of this platform. Basically, they have lifecycle events that fall in the three prime categories which are property updates, Initialization and third are Destruction. They are generally considered as a method of simply managing the state and properties of every reach component.

Q: What do you mean by ReactJS?

It is nothing but a JavaScript library which was developed by the popular social media giant facebook. The prime aim was to build user interfaces. The good thing is that developers can simply create renewable UI components. There are certain factors that have contributed to its success and the one among them is the support of a very large community.

Q: What are the life Cycle of ReactJS?

1. Initialization
2. State/Property Updates
3. Destruction

Q: When ReactJS released?

March 2013

Q: How is ReactJs different from AngularJS?

The first difference between both of them is their code dependency. ReactJS depends less to the code whereas AngularJS needs a lot of coding to be done. The packaging on React is quite strong as compared to the AngularJS. Another difference is React is equipped with Virtual Dom while the Angular has a Regular DOM. ReactJS is all about the components whereas AngularJS focus mainly on the Models, View as well as on Controllers. AngularJS was developed by Google while the ReactJS is the outcome of facebook. These are some of the common differences between the two.

Q: What do you mean by Redux?

Many times there is a need to handle the data of an app in a reliable manner. For such tasks, Redux is used. It accurately performs its task and always makes sure that the entire data has been controlled. It is also possible to apply filters in case only a specific part of data is required.

Q: What do you know about Flux?

Basically, Flux is a basic illustration that is helpful in maintaining unidirectional data stream. It is meant to control construed data unique fragments to make them interface with that data without creating issues. Flux configuration is insipid; it's not specific to React applications, nor is it required to collect a React application. Flux is basically a straightforward idea, however in you have to exhibit a profound comprehension of its usage.

Q: What is current stable version of ReactJS?

Version: 15.5
Release on: April 7, 2017

Q: What is Repository URL of ReactJS?

https://github.com/facebook/react

Q: What is JSX?

It is basically a novel dialect of the popular JavaScript that simply integrates the HTML templates into the code of JavaScript. The browser is not capable to read the code simply and thus there is a need for this integration. Generally, WebPack or Babel tools are considered for this task. It has become a very popular approach in the present scenario among the developers.

Q: Do you think ReactJS has any limitations? If so, tell a few?

Yes, there are a few drawbacks which are associated with this platform. The leading drawback of the ReactJS is the size of its library. It is very complex and creates a lot of confusion among the developers. Also, there are lots of developers all over the world which really don't like the JSX and inline templating. In addition to this, there is another major limitation of ReactJS and i.e. only cover one layer of the app and i.e.View. Thus to manage the development, developers have to depend on several other technologies which consume time.

Q: What are the feature of ReactJS?

1. JSX
2. Components
3. One direction flow (Unidirectional Flux)

Q: What are the Advantages of ReactJS?

Uses **virtual DOM** which is JavaScript object which will improve apps performance.
Can be used on client and server side
Component and Data patterns improve readability.
Improves SEO Performance
Can be used with other framework also.

Q: How the parent and child components exchange information?

This task is generally performed with the help of functions. Actually, there are several functions which are provided to both parent and child components. They simply make use of them through props. Their communication should be accurate and reliable. The need of same can be there anytime and therefore functions are considered for this task. They always make sure that information can be exchanged easily and in an efficient manner among the parent and child components.

Q: How to embed two components in One component?

```
import React from 'react';
class App extends React.Component {
   render() {
      return (
         <div>
            <Header/>
            <Content/>
         </div>
      );
   }
}

class Header extends React.Component {
   render() {
      return (
         <div>
            <h1>Header</h1>
         </div>
      );
   }
}

class Content extends React.Component {
   render() {
      return (
         <div>
            <h2>Content</h2>
            <p>The content text!!!</p>
         </div>
      );
   }
}

export default App;
```


Q: Give one basic difference between pros and state?

Pros are immutable while the state is mutable. Both of them can update themselves easily.

Q: How do you tell React to build in Production mode and what will that do?

Ordinarily you'd utilize Webpack's **DefinePlugin** strategy to set **NODE_ENV** to production. This will strip out things like propType approval and additional notices. Over that, it's likewise a smart thought to minify your code in light of the fact that React utilizes Uglify's dead-code end to strip out advancement just code and remarks, which will radically diminish the measure of your package.

Q: What do you understand with the term polling?

The server needs to be monitored to for updates with respect to time. The primary aim in most of the cases is to check whether novel comments are there or not. This process is basically considered as pooling. It checks for the updates approximately after every 5 seconds. It is possible to change this time period easily. Pooling help keeping an eye on the users and always make sure that no negative information is present on the servers. Actually, it can create issues related to several things and thus pooling is considered.

Q: When would you use a Class Component over a Functional Component?

If your component has state or a lifecycle method(s), use a Class component. or else, use a Functional component.

Q: What do you mean by virtual DOM?

For all the available DOM objects in ReactJS, there is a parallel virtual DOM object. It is nothing but can be considered as the lighter version of the true copy and is powerful in eliminating the complex code. It is also used as a Blue Print for performing several basic experiments. Many developers also use it while practicing this technology.

Q: Compare MVC with Flux?

MVC approaches are presently considered as outdated. Although they are capable to handle data concerns, controllers as well as UI, many developers found that it doesn't properly work when applications size increases. However, they are capable to handle some of the key issues such as eliminating the lack of data integrity as well as managing the data flow which is not properly defined. On the other side, Flux works perfectly with all the sizes irrespective of their size.

Q: What's the difference between an Element and a Component in React?

Basically, a React component describes what you need to see on the screen. Not all that basically, a React element is a protest portrayal of some UI.

A React component is a function or a class which alternatively acknowledges input and returns a React component (ordinarily by means of **JSX** which gets transpiled to a createElement invocation).

Q: Tell us three reasons behind the success of ReactJS?

ReactJS is a technology that can be trusted for complex tasks. While performing any task through it, developers need not worry about the bugs. It always ensures error free outcomes and the best part is it offers scalable apps. It is very fast technology and can simply be trusted for quality outcomes.

Q: In which lifecycle event do you make AJAX requests and why?

AJAX solicitations ought to go in the **componentDidMount** lifecycle event.
There are a couple of reasons behind this,

Fiber, the following usage of React's reconciliation algorithm, will be able to begin and quit rendering as required for execution benefits. One of the exchange offs of this is **componentWillMount**, the other lifecycle event where it may bode well to influence an AJAX to ask for, will be "non-deterministic". This means React may begin calling **componentWillMount** at different circumstances at whenever point it senses that it needs to. This would clearly be a bad formula for AJAX requests.

You can't ensure the AJAX request won't resolve before the component mounts. In the event that it did, that would imply that you'd be attempting to setState on an unmounted component, which won't work, as well as React will holler at you for. Doing AJAX in componentDidMount will ensure that there's a component to update.

Q: What is the difference between createElement and cloneElement?

createElement is the thing that JSX gets transpiled to and is the thing that React uses to make React Elements (protest representations of some UI). cloneElement is utilized as a part of request to clone a component and pass it new props. They nailed the naming on these two.

Q: What do you mean by prop?

ReactJS is equipped with a very amazing feature. It enables developers to simply add our attributes while using the defined components. These attributes are commonly called as props. They are used for rendering the dynamic data and using them is not at all a big deal. Developers can save a lot of time and eliminates the chances of occurrence of bugs and errors by using Prop.

Q: What is meant by event handling?

To capture the user's information and other similar data, event handling system is considered. This is generally done through DOM elements which are present in the code. This task is simple to accomplish. Two-way communication is considered in this approach.

Q: What is the second argument that can optionally be passed to setState and what is its purpose?

A callback work which will be conjured when **setState** has completed and the part is re-rendered. Something that is not talked about a great deal is that **setState** is asynchronous, which is the reason it takes in a moment callback function. Ordinarily it's best to utilize another lifecycle strategy instead of depending on this callback function, however it's great to know it exists.

```
this.setState(
  { username: 'tylermcginnis33' },
  () => console.log('setState has finished and the component has re-rendered.')
)
```

Q: How many outermost elements can be there in a JSX expression?

It must have one JSX element present so that the task can be accomplished easily. Having more than one expression is not an issue but probably it will slow down the process. There are also chances of confusion with more than one expression if you are new to this technology.

Q: What are controlled and uncontrolled components?

There are components in the ReactJS that maintain their own internal state. They are basically considered as uncontrolled components. On the other side, the components which don't maintain any internal state are considered as controlled components in ReactJS. Controlled components can easily be controlled by several methods. Most of the React components are controlled components.

Q: Mention the key benefits of Flux?

Applications that are built on Flux have components which can simply be tested. By simply updating the store, developers are able to manage and test any react component. It cut down the overall risk of data affection. All the applications are highly scalable and suffer no compatibility issues.

Q: What'ss wrong with following code?

```
this.setState((prevState, props) => {
  return {
    streak: prevState.streak + props.count
  }
})
```

Nothing isn't right with it. It's once in a while utilized and not outstanding, but rather you can likewise pass a function to setState that gets the past state and props and returns another state, similarly as we're doing above. Furthermore, is nothing amiss with it, as well as effectively recommended in case you're setting state in light of previous state.

Q: Why browsers cannot read JSX?

Actually, JSX is not considered as a proper JavaScript. Browsers cannot read it simply. There is always a need to compile the files that contain JavaScript Code. This is usually done with the help of JSX compiler which performs its task prior to file entering the browser. Also, compiling is not possible in every case. It depends on a lot of factors such as source or nature of file or data.

Q: What are pure functional Components?

Traditional React Components as we have seen so far are making a class with class Example extends React.Component or React.createClass(). These make stateful components on the off chance that we at any point set the state (i.e. this.setState(), getInitialState(), or this.state = {} inside a constructor()).

In the event that we have no expectation for a Component to require state, or to require lifecycle methods, we can really compose Components with an pure function, consequently the expression "pure function Component":

```
function Date(props){
    let {msg="The date is:"} = props
    let now = new Date()
    return <div>
        <span>{msg}</span>
        <time>{now.toLocaleDateString()}</time>
    </div>
}
```

This function that returns a React Element can be used whereever we see fit:

```
DOM.render(<div><Date msg="Today is"/><div>)
```

You might notice that also takes a prop – we can still pass information into the Component.

Q: How DOM and Virtual Dom object are different from one another?

Virtual DOM is not capable to affect the HTML directly. As compared to a regular DOM, Virtual is quite faster and can perform its task reliably. In addition to this, Virtual Dome is capable to automate itself. Also, Virtual DOM is capable to handle more tasks without facing any of the issues.

Q: What happens during the lifecycle of a React component?

A standout amongst the most valuable parts of React is its segment lifecycle — so seeing precisely how segments components after some time is instrumental in building a viable application.

Q: What exactly you can do if the expression contains more than one line?

In such a situation, enclosing the multi-line JSX expression is an option. If you are a first time user, it may seem awkward but later you can understand everything very easily. Many times it becomes necessary to avoid multi-lines to perform the task reliably and for getting the results as expected.

Q: Is it possible to use the word "Class" in JSX. Why or why not?

No, it is not possible in the JSX. This is because the word "Class" is a reticent (occupied) word in the JavaScript. However, you can use you are free to use the word "ClassName". If you use the word "Class" the JSX will be translated to JavaScript immediately.

a) High-Level Component Lifecycle:

At the most highest level, React components have lifecycle eventsthat fall into 3 general classifications:

Initialization
State/Property Updates
Destruction

Each React component defines these eventsas a system for dealing with its properties, state, and rendered output. Some of these events just happen once, others happen more as often as possible; understanding these 3 general classes should help you clearly visualize when certain logic required to be applied.

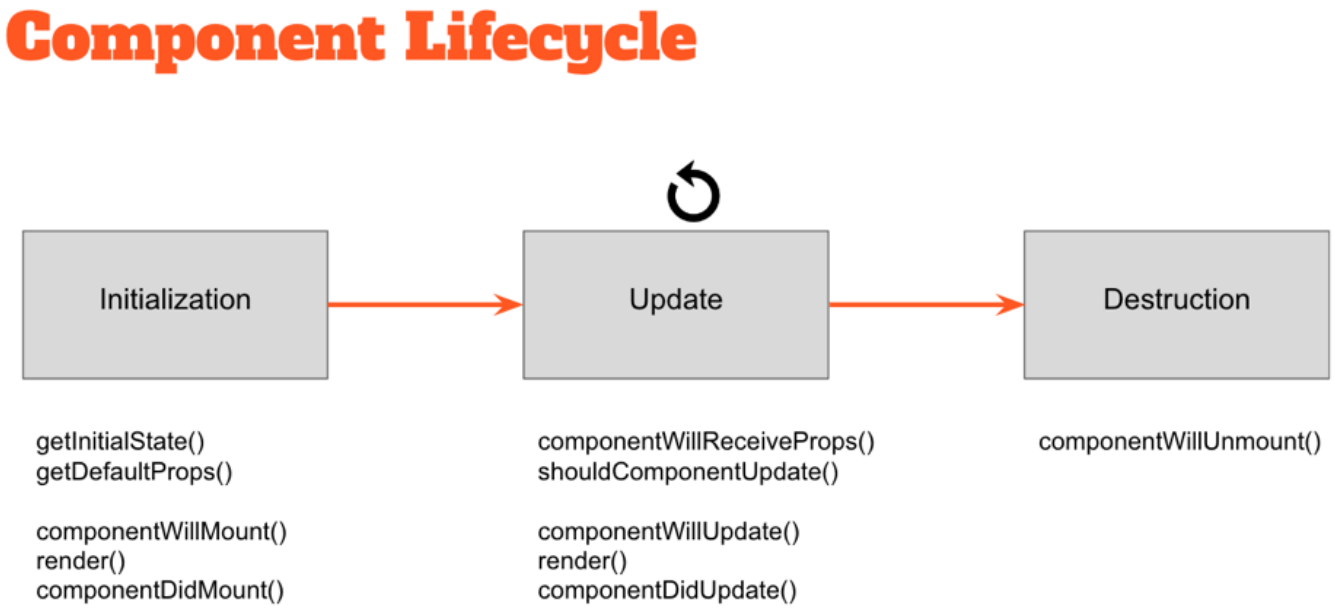
For instance, a component may need to add event audience to the DOM when it initially mounts. In any case, it ought to likely expel those event listeners when the component unmounts from the DOM with the goal that not relevant handling that does't occur.

```
class MyComponent extends React.Component {
  // when the component is added to the DOM...
  componentDidMount() {
    window.addEventListener('resize', this.onResizeHandler);
  }

  // when the component is removed from the DOM...
  componentWillUnmount() {
    window.removeEventListener('resize', this.onResizeHandler);
  }

  onResizeHandler() {
    console.log('The window has been resized!');
  }
}
```

b) Low-Level Component Lifecycle:



Inside these 3 general buckets exist various particular lifecycle hooks — basically unique techniques - that can be used by any React component to all the more precisely manage updates. Seeing how and when these hooks fire is vital to building stable components and will empower you to control the rendering procedure (enhancing execution).

Observe the diagram above. The events under "Initialization" just happen when a component is first initialized or added to the DOM. Thus, the events under "Devastation" just happen once (when the component is expelled from the DOM). However, the events under "Update" happen each time the properties or state of the component change.

For instance, components will naturally re-render themselves whenever their properties or state change. However, at times a component should not update - so keeping the component from re-rendering may enhance the execution of our application.

```
class MyComponent extends React.Component {
  // only re-render if the ID has changed!
  shouldComponentUpdate(nextProps, nextState) {
    return nextProps.id === this.props.id;
  }
}
```

Q: What do you know about React Router?

Rendering the components is an important task in ReactJS. React router is used to decide which components is to be rendered and which one should not. It also performs dictation during several activities.

Q: Compare Flux vs MVC

Conventional MVC designs have functioned admirably to separate the worries of data (Model), UI (View) and logic (Controller) — however many web engineers have found impediments with that approach as applications develop in measure. In particular, MVC architectures as often as possible experience 2 primary issues:

Ineffectively defined data flow: The cascading updates which happen crosswise over perspectives frequently prompt a tangled web of events which is hard to debug.

Lack of data integrity: Model data can be changed from anyplace, yielding erratic results over the UI.

With the Flux pattern complex UIs never again experience the ill effects of cascading updates; any given React component will have the capacity to recreate its state in light of the information gave by the store. The flux pattern likewise upholds data integrity by limiting direct access to the shared data.

While a technical interview, it is awesome to talk about the contrasts between the Flux and MVC configuration designs inside the setting of a particular illustration:

For instance, imagine we have an "master/detail" UI in which the client can choose a record from a rundown (master view) and alter it utilizing an auto-populated form (detail view).

With a MVC architecture, the data contained inside the Model is shared between both the master and detail Views. Each of these perspectives may have its own particular Controller assigning updates between the Model and the View. Anytime the information contained inside the Model may be updated — and it's hard to know where precisely that change happened. Did it occur in one of the Views sharing that Model, or in one of the Controllers? Since the Model's information can be transformed by any performing artist in the application, the danger of information contamination in complex UIs is more prominent than we'd like.
With a Flux architecture, the Store data is correspondingly shared between different Views. However this data can't be straightforwardly changed — the greater part of the solicitations to update the data must go through the Action > Dispatcher chain first, eliminating of the risk of arbitrary data pollution. At the point when refreshes are made to the data , it's presently significantly less demanding to find the code requesting for those progressions.

Q: What are stateless components?

On the off chance that React components are basically state machines that produce UI markup, at that point what are stateless segments?

Stateless components (a kind of "reusable" components) are simply pure functions that render DOM construct exclusively with respect to the properties gave to them.

```
const StatelessCmp = props => {
  return (
    <div className="my-stateless-component">
      {props.name}: {props.birthday}
    </div>
  );
};

// ---
ReactDOM.render(
  <StatelessCmp name="Art" birthday="10/01/1980" />,
  document.getElementById('main')
);
```

As you can see, this component has no requirement for any internal state — not to mention a constructor or lifecycle handlers. The yield of the component is absolutely a function of the properties gave to it.

Q: What is one of the core types in React?

ReactNode

Q: What do you mean by the state?

It is basically a JavaScript object that is meant to effectively handle the changes into the data. Generally, it is present inside in all the components that are used. It is considered as an important part of RectJS apps which boost user interfaces. As it represents the data that change over the time, certain errors can be eliminated and developers can always ensure quality.

Q: What is redux?

A method os handling the state (or data) of an application.

Q: Is it possible to display props on a parent component?

Yes, it is possible. The best way to perform this task is using spread operator. It can also be done with listing the properties but this is a complex process.

Q: In ReactJS, why there is a need to capitalize the components?

It is necessary because components are not the DOM element but they are constructors. If they are not capitalized, they can cause various issues and can confuse developers with several elements. At the same time, the problem of integration of some elements and commands can be there.

Q: What do you know about synthetic events?

ReactJS is capable to simply transform original events of browsers by monitoring the behavior of browser. This is done to make sure that events have logical properties beyond the browsers that are different. Actually, they act as cross browser envelope around the logical event system of the browser.

Q: Explain DOM diffing?

When the components are rendered twice, Virtual Dom begins checking the modifications elements have got. They represent the changed element on the page simply. There are several other elements that don’t go through changes. To cut down the changes to the DOM as an outcome of user activities, DOM doffing is considered. It is generally done to boost the performance of the browser. This is the reason for its ability to perform all the tasks quickly.

Q: Is it possible to nest JSX elements into other JSX elements?

It is possible. The process is quite similar to that of nesting the HTML elements. However, there are certain things that are different in this. You must be familiar with the source and destination elements to perform this task simply.

Social Share

(https://www.facebook.com/sharer.php?u=https://mindmajix.com/reactjs-interview-questions) (https://twitter.com/intent/tweet?text=https://mindmajix.com/reactjs-interview-questions) (https://plus.google.com/share?url=https://mindmajix.com/reactjs-interview-questions) (https://www.linkedin.com/shareArticle?mini=true&url=https://mindmajix.com/reactjs-interview-questions&title=https://mindmajix.com/reactjs-interview-questions&summary=&source=)

Previous (https://mindmajix.com/python-regular-expression-cheatsheet)

Next (https://mindmajix.com/certified-ethical-hacker-exam)

Mahesha Reddy • 3 months ago

I really like your question and answers, am a fresher and wanted to achieve my career in Reactjs, thanks a lot for the post!!

• Reply • Share ›

Josna Peters Mod Mahesha Reddy • 2 months ago

Nice to hear it dear Mahesha. Wish you a great career advancements.

Stay Tuned
MindMajix.com

• Reply • Share ›

ALSO ON MINDMAJIX

Ethereum VS Hyperledger

1 comment • 6 months ago

Audrey Signs — I think it's a shame that Corda that is created by R3 isn't added to compare. It's also not a Hyperledger project.

Define Elasticsearch Custom Analyzer with Example

1 comment • 10 months ago

Mallik Arjun — Nice article.

Oracle Fusion SCM Interview Questions

4 comments • 7 months ago

murali krishna — Thank you for sharing such a nice and interesting blog with us. I have seen that all will say the same thing repeatedly. But in

Advanced SAP ESS MSS Interview Questions And Answers

2 comments • 10 months ago

Josna Peters — Thanks Swetanksingh...Stay TunedMindmajix.com

Subscribe Add Disqus to your siteAdd DisqusAdd Disqus' Privacy PolicyPrivacy PolicyPrivacy

Popular Courses in 2018

Salesforce Training (https://mindmajix.com/...

★★★★★ (5.0)

2357 Enrolled

Selenium Training (https://mindmajix.com/...

★★★★★ (5.0)

3370 Enrolled

Splunk Training (https://mindmajix.com/

★★★★★ (5.0)

1324 Enrolled

SEARCH

CATEGORIES

- Hibernate (https://mindmajix.com/hibernate/)
- SAP MDM (https://mindmajix.com/sap-mdm/)
- Artificial Intelligence (https://mindmajix.com/artificial-intelligence/)
- Ansible (https://mindmajix.com/ansible/)
- Hibernate Validator (https://mindmajix.com/hibernate-validator/)
- FortiMail (https://mindmajix.com/fortimail/)
- CATIA (https://mindmajix.com/catia/)
- FortiAnalyzer (https://mindmajix.com/fortianalyzer/)
- IBM BPM (https://mindmajix.com/ibm-bpm/)
- Python (https://mindmajix.com/python/)

VIEW MORE (https://mindmajix.com/blog-category)

RELATED POSTS

ReactJS Interview Questions (https://mindmajix.com/reactjs-interview-questions)

Aug 24, 2017

Introduction to ReactJS (https://mindmajix.com/introduction-to-react-js)

Aug 23, 2017

Majix

(https://mindmajix.com/)

Mindmajix - Online global training platform connecting individuals with the best trainers around the globe. With the diverse range of courses, Training Materials, Resume formats and On Job Support, we have it all covered to get into IT Career. Instructor Led Training - Made easy.

in
(https://www.linkedin.com/company/mindmajix-technologies-pvt-

(https://www.linkedin.com/company/Mindmajix-Technologies)

Company

Home (https://mindmajix.com/)

About Us (https://mindmajix.com/about)

Courses (https://mindmajix.com/course)

Sample Resumes (https://mindmajix.com/resume)

Blog (https://mindmajix.com/blog)

Contact Us (https://mindmajix.com/contact)

Call us on

Leave a Message

Reviews (https://mindmajix.com/review)

Trending Courses

Tableau Training (https://mindmajix.com/tableau-training)

Oracle DBA training (https://mindmajix.com/oracle-dba-training)

Qlikview Training (https://mindmajix.com/qlikview-training)

Docker Training (https://mindmajix.com/docker-training)

JBoss Training (https://mindmajix.com/jboss-training)

Informatica Training (https://mindmajix.com/informatica-training)

Cassandra Training (https://mindmajix.com/apache-cassandra-training)

View All... (https://mindmajix.com/course)

Contact Info

244 Fifth Avenue, Suite 1222
New York(NY)
United States (US) - 10001

info@mindmajix.com

USA : +1 917 456 8403

#811, 10th A Main, Suite No.506
1st Floor, Indira Nagar
Bangalore, India - 560038

info@mindmajix.com

INDIA: +91 905 240 3388