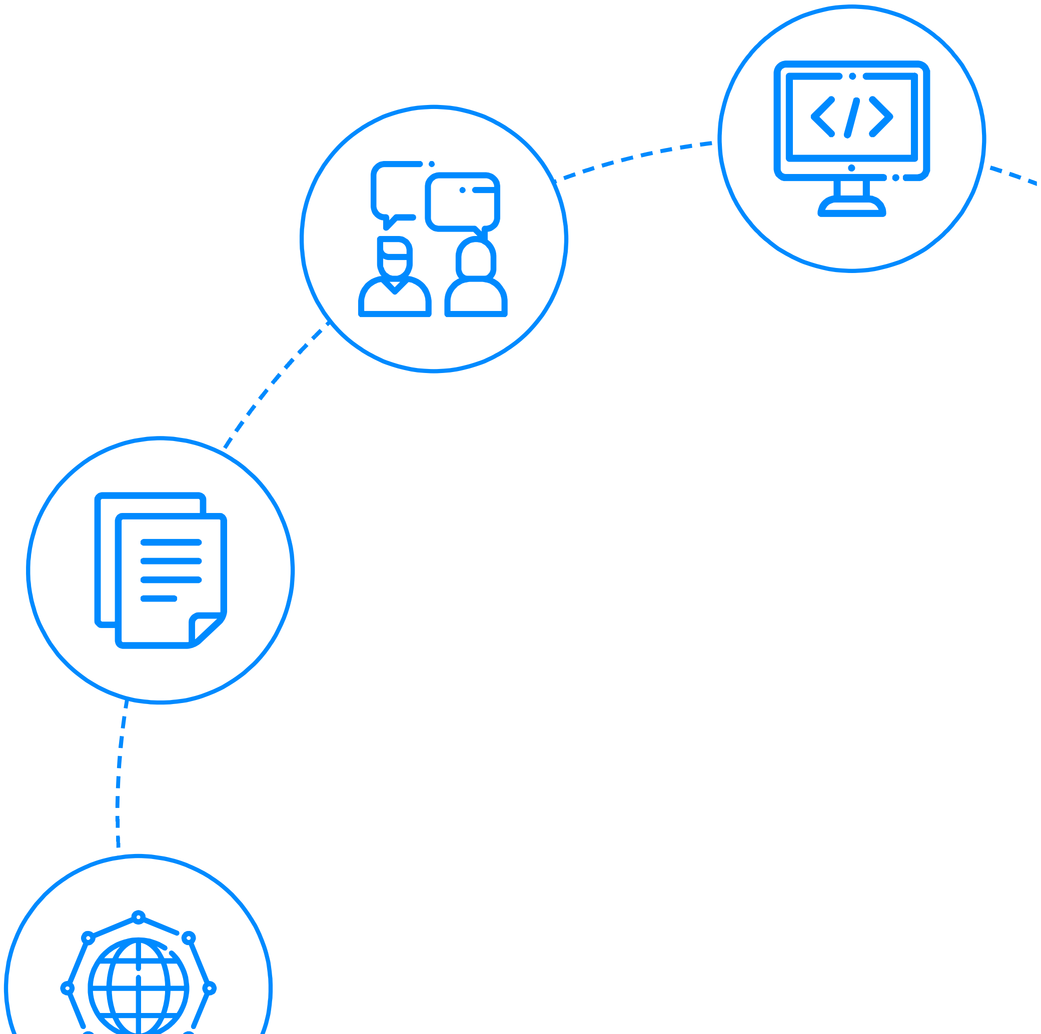
Redux Interview Questions



To view the live version of the page, click here.

Contents

Redux Interview Questions for Experienced

(.....Continued)

19. Explain the typical data flow in an application made using React and Redux (Redux Lifecycle for an application).
20. What is the mental model of redux saga?
21. Describe what is meant by a "store" in Redux.
22. Name all the Redux Store methods.
23. Give an example depicting the usage of a Redux store.
24. Explain with an example how to set the initial state in Redux.
25. What do you understand about Redux Thunk?
26. What are the workflow features in Redux?
27. Differentiate between Relay and Redux.
28. How can we use connect from react redux?
29. What are Redux forms? What are its major features?
30. How can we structure the top level directories in Redux?
31. How can we access a redux store outside a react component?
32. Differentiate between React Redux and React's Context API.
33. What do you understand about the Redux Saga?
34. What are the things which we should never do inside a reducer?
35. How can the addition of multiple middlewares to Redux be done?

Let's get Started

Introduction to Redux

Developed by Dan Abramov and Andrew Clark in the year 2015, Redux is nothing but an open-source library made using the scripting language JavaScript. Redux's primary use lies in managing and centralizing application states and it is usually used along with JavaScript libraries, for instance, React or Angular to build UIs (User Interfaces).

Redux's architecture is somewhat inspired by the architecture of Flux by Meta (or Facebook). Redux can help developers develop applications that behave consistently, run in different environments (client, server, and native), and are easy to test. Redux can help the developers to centralize their application's state and logic. Because of this, a variety of amazing capabilities, for instance, Undo and Redo, state persistence, etc. are unlocked by Redux. Redux also provides a large number of add ons and can work with all sorts of User Interfaces. The Dev Tools provided by Redux also help us debug our applications.

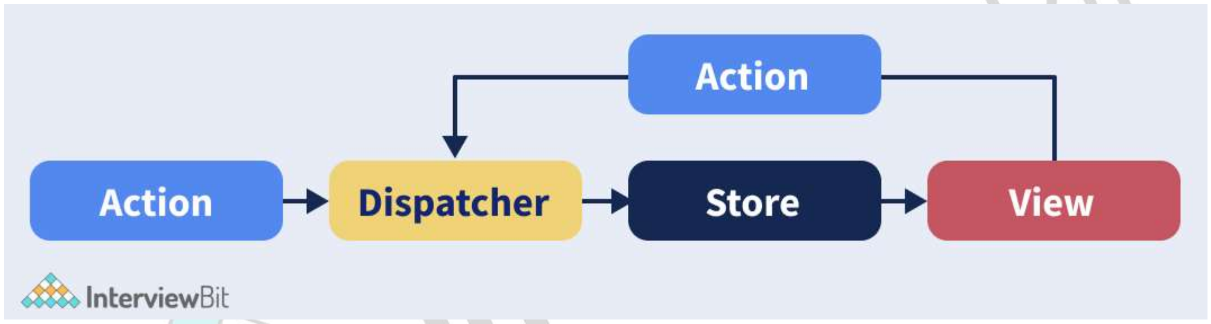
Redux Interview Questions for Freshers

1. What is Redux?

Redux is an open-source library made using the scripting language JavaScript. Redux's primary use lies in managing and centralizing application state and it is usually used along with JavaScript libraries, for instance, React or Angular in order to build UIs (User Interfaces). It is a predictable state container for applications built using JavaScript. It is based on the Flux design pattern. Redux is very small in size (around 2 kilobytes) and has no dependencies.

2. What is Flux?

Flux is an application design paradigm just like the Model View Controller design pattern. Flux is nothing but a new kind of architecture that complements React and the concept of Unidirectional Data Flow. The image given below shows how the workflow between dispatcher, stores and views components with distinct inputs and outputs are in Flux:



3. Difference between Redux and Flux.

The main differences in the comparison: Redux vs Flux are as follows:

COMPARISON PARAMETER	REDUX	FLUX
Number of stores per application	Redux includes a single Store per application. Rather than placing state information in multiple Stores across the app, Redux keeps everything in one region of the application	Flux includes multiple Stores per application.
Architecture	Redux is an open-source JavaScript library used for creating User Interfaces.	Flux's architecture is designed to build client-side web apps.

4. What is Redux in React js?

Redux in React is the official React binding for Redux which allows the components in React to read data from a Redux Store, and dispatch Actions to the Store for updating the data. The purpose of Redux is to help applications scale well by providing means to manage the state via a unidirectional data flow model.

5. State the core principles of Redux.

The core principles of Redux are as follows:

- **Single source of truth:** The global state of our application is always put away in an object tree inside one store.
- **The state is read-only:** The only way to change the state of our application is by emitting an action, an object explaining what has happened.
- **Changes are made with pure functions:** This principle means that in order to define how the state tree is being transformed by the actions, we have to write pure reducers.

6. What are the advantages of using Redux?

Some of the advantages of using Redux are as follows:

- Redux provides extremely easy state transfer between the components.
- The states are always predictable in Redux and its maintenance is relatively easy.
- Debugging and testing code in Redux is simple through logging behaviour and status.
- Redux provides great performance. It might occur to us that keeping the application's state global would result in bad performance. However, usually, that is not the case as React Redux implements a lot of performance optimizations internally so that our own connected component only re-renders when it actually needs to.
- Redux also offers state persistence by storing the application's state to local storage and restoring it after a refresh.

7. Is it true that Redux can only be used with React?

No, it is not true that Redux can only be used with React. Redux is being used as a data store for lots of UI layers. There are bindings available in Redux for React, Angular, Angular 2, Vue, etc.

8. What do you understand about Redux Toolkit?

Redux Toolkit is Redux's official, opinionated, batteries included toolset for efficient Redux development. It also consists of the most widely used Redux add-ons, for instance, Redux Thunk for asynchronous logic, Reselect for writing selector functions and many more for making development easy for developers and saving them time.

9. What are some of the major features of Redux DevTools?

Some of the major features of Redux DevTools are as follows:

- Redux DevTools is nothing but a time travel environment that makes it possible for us to live edit in Redux with a variety of functionalities like action replay, hot reloading, and customizable UI.
- Redux DevTools makes it possible for us to inspect all the states and action payload. We can go back into the time simply by cancelling the actions.
- Each stage action is re-evaluated in case we change the code of the reducer.
- We can also continue our debug sessions across page reloads with the help of persistState() store enhancer.

10. Is it necessary to keep all the component states in the Redux store?

No, it is not necessary to keep all the component states in the Redux store. We have to keep your application state as small as possible and therefore, we should do it only if it makes a difference for us to keep something there or maybe if it makes the use of Dev Tools easier.

11. Highlight the key differences between mapStateToProps() and mapDispatchToProps()?

The key differences between mapStateToProps() and mapDispatchToProps() are given below:

mapStateToProps()	mapDispatchToProps()
The mapStateToProps() method is used to render the stored data to the component.	The mapDispatchToProps() method is used to render the action creators with props to the component.
The entirety of the results of the mapStateToProps() method is a plain object which is later merged into the component’s prop.	In the mapDispatchToProps() method, each action creator is wrapped in the dispatcher call so that they can be called upon directly and later merged into the component’s prop.
This method's use is to connect the redux state to the props of the react component.	This method's use is to connect redux actions to the react props.

12. What do you understand by an action in Redux's architecture?

In the Redux architecture, actions are nothing but the plain JavaScript objects which contain a type field. They can be thought of as an event that is used to describe something which has happened in the application. Actions contain only a tiny bit of information that is required to mention what has happened.

13. Give an example of the usage of Actions in Redux's architecture.

An example of the usage of Actions in Redux's architecture is given below:

```
const addingTodoAction = {
     type: 'ADD',
     payload: 'Do-homework'
}
```

14. Where can we use Redux?

Redux is primarily being used along with React. However, we can also use it along with Angular, Vue, Meteor, etc. using the bindings it has to offer to us.

15. What are constants in Redux?

Constants in Redux help us in easily finding all the usages of a particular functionality across our entire project when we are using an Integrated Development Environment (IDE). Using constants, we can avoid silly bugs caused because of typing errors or typos as it shows a "ReferenceError" whenever a typo is made.

16. Show using code how constants can be used in Redux.

First of all, we can store all the constants in a single file in our project named constants.js or something else as follows:

```
export const ADDING_TODO = 'ADDING_TODO';
export const DELETING_TODO = 'DELETING_TODO';
export const EDITING_TODO = 'EDITING_TODO';
export const COMPLETING_TODO = 'COMPLETING_TODO';
export const COMPLETING_ALL = 'COMPLETING_ALL';
export const CLEARING_COMPLETED = 'CLEARING_COMPLETED';
```

After storing the constants in one place, we can use them in two ways in our project:

- During actions creation (in actions.js file of our project):

```
 import { DELETING_TODO } from './constants';

export function deletingTodo(text) {
  return { type: DELETING_TODO, text };
}
```

- In Reducers (in reducer.js file of our project):

```
 import { EDITING_TODO } from './constants';

export default (state = [], action) => {
  switch (action.type) {
    case EDITING_TODO:
      return [
        ...state,
        {
          text: action.text,
          completed: false
        }
      ];
    default:
      return state
  }
};
```

17. What are reducers in Redux's architecture?

Reducers in Redux's architecture are pure functions that are used to take the previous state and an action and return the next state. Its syntax is given below:

```
 (previous_state, action) => new_state
```

18. Show with an example how reducers are used in Redux.

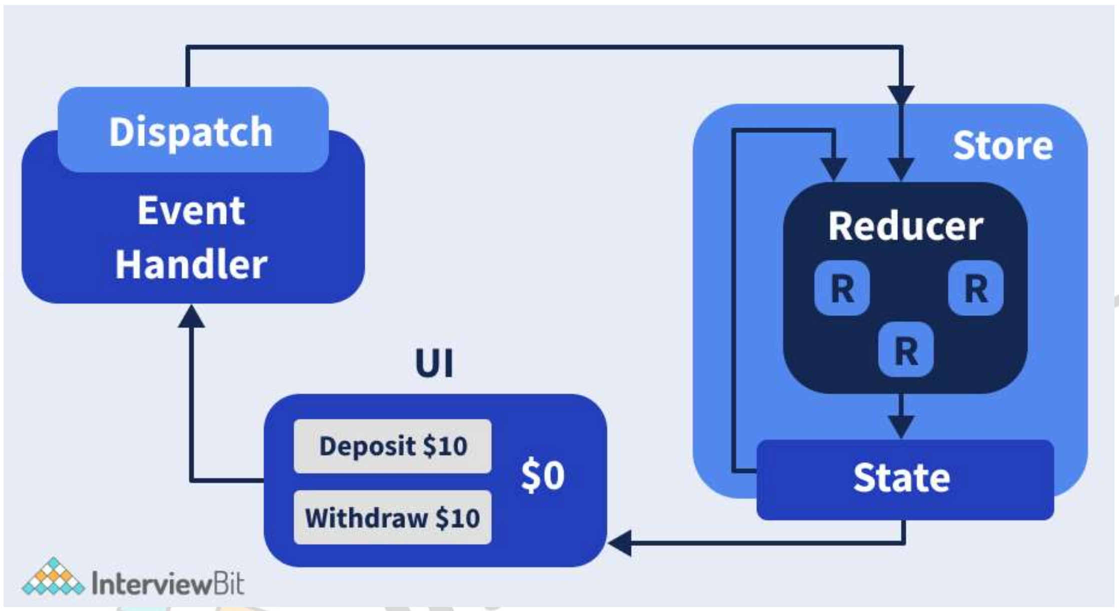
An example of how reducers are used in Redux is given below:

```
const initial_state = { value: 0 }
function countReducer(state = initial_state, action) {
 // Checking to see if the reducer cares about this action
 if (action.type === 'counter/incremented') {
   // If the action is of type "counter" or "incremented", we make a copy of `state`
   return {
     ...state,
     // We also update the copy with the new value
     value: state.value + 1
   }
 }
 // If not, we return the original state unchanged
 return state
}
```

Redux Interview Questions for Experienced

19. Explain the typical data flow in an application made using React and Redux (Redux Lifecycle for an application).

The typical data flow in Redux starts with a call back from the User Interface component which dispatches an action with a payload. After that, the reducers intercept and receive the dispatched actions, generating a new application state. After that, the actions are propagated down through a hierarchy of components from the Redux store.



20. What is the mental model of redux saga?

Redux Saga functions as a separate thread in our programme which is solely responsible for side effects. Redux Saga is a redux middleware. In other words, it means that it can be started, paused, and aborted from the main application using standard Redux actions, has access to the entire Redux application state, and can also dispatch Redux actions.

21. Describe what is meant by a "store" in Redux.

“Store” in Redux is used to carry together all the states, reducers, and actions which create the app. Some of the responsibilities of the store are as follows:

- The state of the current application from inside is held by the Redux Store.
- We can access the current state using store.getState().
- We can update the state using store.dispatch(action).
- We can also register listener callbacks using the store.subscriber(listener).

22. Name all the Redux Store methods.

All the Redux Store Methods are as follows:

- getState()
- subscribe(listener)
- dispatch(action)
- replaceReducer(nextReducer)

23. Give an example depicting the usage of a Redux store.

An example depicting the usage of a Redux store is given below:

```
 import { createStore } from 'redux'
const store = createStore(todos, ['Use Redux'])
function deletingTodo(text) {
 return {
   type: 'DELETING_TODO',
   text
 }
}
store.dispatch(deletingTodo('Do the homework'))
store.dispatch(deletingTodo('Buy coffee'))
```

24. Explain with an example how to set the initial state in Redux.

In order to set the initial state in Redux, we have to pass the initial state as the second argument to createStore as shown below:

```
 const rootReducer = combineReducers({
  todos: todos,
  visibilityFilter: visibilityFilter
});

const initialState = {
  todos: [{id:100, name:'ritik', completed: true}]
};

const store = createStore(
  rootReducer,
  initialState
);
```

25. What do you understand about Redux Thunk?

Using Redux Thunk middleware, we can write action creators returning a function instead of an action. This thunk can postpone the dispatch of an action, or do conditional dispatchment. The arguments passed to the inner function are the store methods dispatch and getState(). In the event of an action creator returning a function, the function gets executed by the Redux Thunk middleware and it does not have to be pure. In other words, the function is allowed to have side effects, including executing asynchronous API calls. It can even dispatch actions. Redux thunk is used to delay the dispatch of an action, or to dispatch in the event of a certain condition being met. At the time of dispatch of a function instead of an action object, if Redux Thunk middleware is enabled, the middleware will call that function with the dispatch method itself as the first argument.

26. What are the workflow features in Redux?

The workflow features in Redux are as follows:

- **Reset**: The state of the store is allowed to be reset.
- **Revert**: Revert or Rollback to the last committed state is allowed.
- **Sweep**: Every disabled action which we have fired unintentionally will be removed.
- **Commit**: The current state is made the initial state.

27. Differentiate between Relay and Redux.

The key differences between Relay and Redux are given below:

Relay	Redux
The state originating from the server is taken care of by Relay.	All the states of the application are taken care of by Redux.
Relay can be used for caching and optimizing the data.	Redux is not responsible for handling data fetching (it can be done manually though).

28. How can we use connect from react redux?

In order to use connect from React Redux, we will have to follow a couple of steps to use our store in our container:

- Firstly, we use the mapStateToProps(): This would map the state variables from our store to the props which we specify.
- Secondly, we connect the above props to our container: The object returned by the mapStateToProps component is connected to the container.

A sample code for connect from react-redux is given below:

```
import React from 'react';
import { connect } from 'react-redux';

 class App extends React.Component {
   render() {
     return <div>{this.props.containerData}</div>;
   }
 }

 function mapStateToProps(state) {
   return { containerData: state.appData };
 }

 export default connect(mapStateToProps)(App);

function mapStateToProps(state) {
  return { containerData: state.data };
}

export default connect(mapStateToProps)(App);
```

29. What are Redux forms? What are its major features?

Redux Forms present in React and Redux help in enabling a form in React to use Redux for storing all of its states. They can be used with raw inputs in HTML5. Redux forms work extremely well with User Interface (UI) frameworks, for instance, Material UI, React Widgets, React Bootstrap and many more.

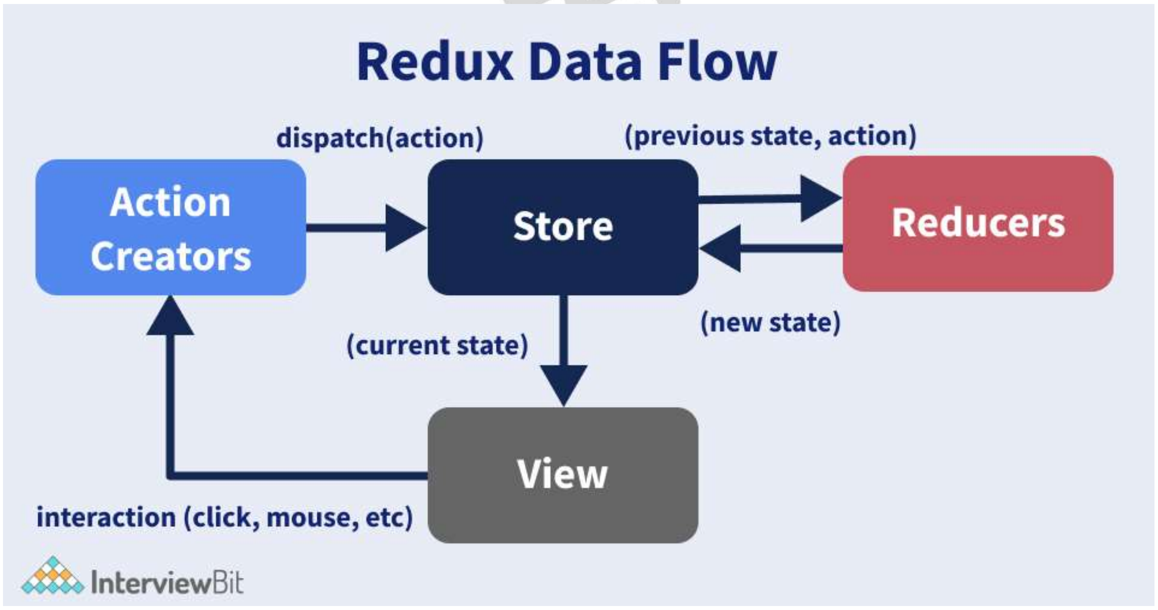
The **major features of Redux forms** are as follows:

- Field values persistence through the Redux store.
- Validation (synchronous/asynchronous) and submission.
- Formatting, parsing and normalization of field values.

30. How can we structure the top level directories in Redux?

Every Redux application has multiple top-level directories as given below:

- **Components**: Components are used for “dumb” React components unfamiliar with Redux.
- **Containers**: Containers are used for “smart” React components that are connected to Redux.
- **Actions**: Actions are used for all the action creators, where the file name should be corresponding to the part of the app.
- **Reducers**: Reducers are used for all the reducers where the file name is corresponding to the state key.
- **Store**: Stores are used for store initialization. This directory works best in small and mid-level size apps.



31. How can we access a redux store outside a react component?

For accessing a redux store outside a react component, we can export the store from the module where it has been created with createStore as done in the following example:

```
store = createStore(reducer);
export default store;
```

32. Differentiate between React Redux and React's Context API.

The key differences in the comparison: Context Api vs Redux are as follows:

React Redux	React Context API
In order to use React-Redux in our application, we need to code it separately and then merge it into the main project.	React Context can be used in the application directly.
The number of features provided by React-Redux is comparatively more than React Context.	The number of features provided by React Context is comparatively less than React-Redux.

33. What do you understand about the Redux Saga?

Redux Saga is a middleware library that can be useful for allowing a Redux store to interact with the resources outside of itself in an asynchronous manner, for example, making HTTP requests to external services, accessing browser storage, executing Input/Output operations and many more. These operations are also called side effects.

34. What are the things which we should never do inside a reducer?

The things which we should never do inside a reducer are as follows:

- Modify the argument of the reducer
- We should assure that we do not perform any side operations such as routing transitions, API calls, etc.
- We should not call non-pure functions, for instance Date.now(), Math.random(), etc.

35. How can the addition of multiple middlewares to Redux be done?

In order to add multiple middlewares to Redux, the usage of applyMiddleware can do the work. In applyMiddleware, we can pass every piece of middleware as a new argument. Therefore, our job is to just pass each piece of middleware that we want. In the example given below, we have added Redux Thunk and logger middlewares as an argument:

```
import { createStore, applyMiddleware } from 'redux'
const createStoreWithMiddleware = applyMiddleware(ReduxThunk, logger)(createStore);
```

Conclusion

The purpose of this article was to explain to our readers what Redux actually is, what is it used for and what are the most important questions asked in the interviews. It is one of the most important JavaScript libraries which is being used alongside React for building a number of applications seamlessly and therefore, we would suggest that any budding talent of today should be familiar with the concepts of Redux to get an edge over others.

Given below are some of the references and recommended resources to learn more about Redux:

- Redux Documentation and API usages
- Redux Github
- For using Redux with Node Package Manager - NPM
- React
- Angular

Links to More Interview Questions

C Interview Questions	Php Interview Questions	C Sharp Interview Questions
Web Api Interview Questions	Hibernate Interview Questions	Node Js Interview Questions
Cpp Interview Questions	Oops Interview Questions	Devops Interview Questions
Machine Learning Interview Questions	Docker Interview Questions	Mysql Interview Questions
Css Interview Questions	Laravel Interview Questions	Asp Net Interview Questions
Django Interview Questions	Dot Net Interview Questions	Kubernetes Interview Questions
Operating System Interview Questions	React Native Interview Questions	Aws Interview Questions
Git Interview Questions	Java 8 Interview Questions	Mongodb Interview Questions
Dbms Interview Questions	Spring Boot Interview Questions	Power Bi Interview Questions
Pl Sql Interview Questions	Tableau Interview Questions	Linux Interview Questions
Ansible Interview Questions	Java Interview Questions	Jenkins Interview Questions