20 ReactJS Questions and Answers (Part 2)

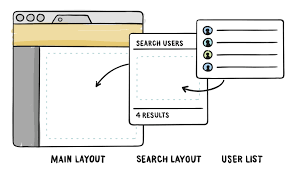
**11. What is ReactDOMServer?**

The ReactDOMServer object enables you to render components to static markup(typically used on node server). This object is mainly used for server-side rendering(SSR).

**12. What is create-react-app and benefits of it?**

Create React App is a tool built by developers at Facebook to help you build React applications. It saves you from time-consuming setup and configuration. You simply run one command and create react app sets up the tools you need to start your React project.

**13. What is React Router?**



**React Router** is a collection of navigational components that compose declaratively with your application. It helps you add new screens and flows to your application incredibly quickly, all while keeping the URL in sync with what’s being displayed on the page.

**14. What is Shallow Renderer in ReactJS testing?**

When writing unit tests for React, shallow rendering can be helpful. Shallow rendering lets you render a component “one level deep” and assert facts about what its render method returns, without worrying about the behavior of child components, which are not instantiated or rendered. This does not require a DOM.

For example, if you have the following component:

function MyComponent() {  
 return (  
 <div>  
 <span className="heading">Title</span>  
 <Subcomponent foo="bar" />  
 </div>  
 );  
}

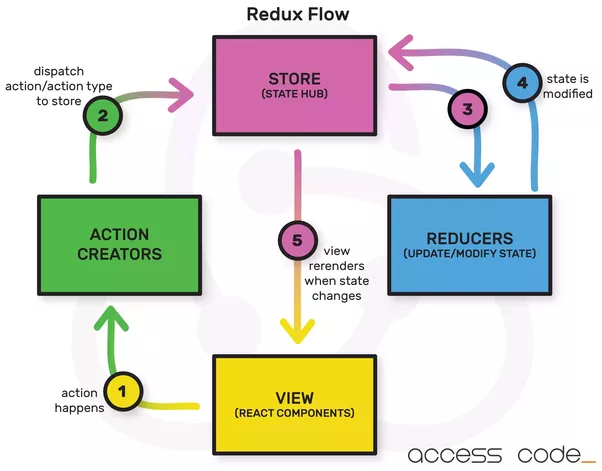
Then you can assert:

import ShallowRenderer from 'react-test-renderer/shallow';  
  
// in your test:  
const renderer = new ShallowRenderer();  
renderer.render(<MyComponent />);  
const result = renderer.getRenderOutput();  
  
expect(result.type).toBe('div');  
expect(result.props.children).toEqual([  
 <span className="heading">Title</span>,  
 <Subcomponent foo="bar" />  
]);

**15. What is JEST?**

Jest is a JavaScript unit testing framework made by Facebook based on Jasmine and provides automated mock creation and a jsdom environment. It’s often used for testing React components.

# 16. What is Redux?



Redux is a predictable state container for JavaScript apps. (Not to be confused with a WordPress framework — Redux Framework.)

It helps you write applications that behave consistently, run in different environments (client, server, and native), and are easy to test. On top of that, it provides a great developer experience, such as live code editing combined with a time traveling debugger.

You can use Redux together with React, or with any other view library. It is tiny (2kB, including dependencies).

## There are Three Principles of Redux:

**Single source of truth: The state of your whole application is stored in an object tree within a single store.**

This makes it easy to create universal apps, as the state from your server can be serialized and hydrated into the client with no extra coding effort. A single state tree also makes it easier to debug or inspect an application; it also enables you to persist your app’s state in development, for a faster development cycle. Some functionality which has been traditionally difficult to implement — Undo/Redo, for example — can suddenly become trivial to implement, if all of your state is stored in a single tree.

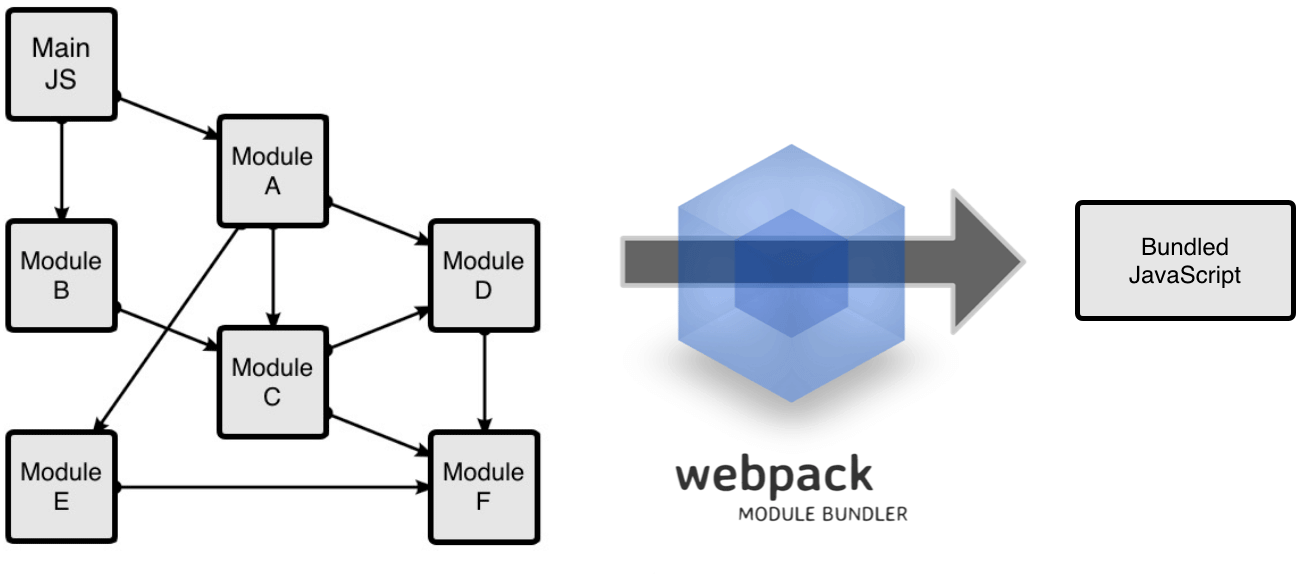
**State is read-only: The only way to change the state is to emit an**[**action**](https://redux.js.org/glossary#action)**, an object describing what happened.**

This ensures that neither the views nor the network callbacks will ever write directly to the state. Instead, they express an intent to transform the state. Because all changes are centralized and happen one by one in a strict order, there are no subtle race conditions to watch out for. As actions are just plain objects, they can be logged, serialized, stored, and later replayed for debugging or testing purposes.

**Changes are made with pure functions: To specify how the state tree is transformed by actions, you write pure**[**reducers**](https://redux.js.org/glossary#reducer)**.**

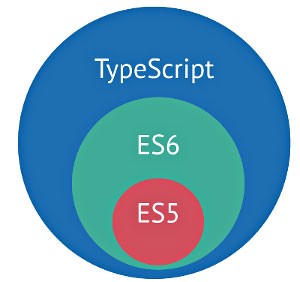
Reducers are just pure functions that take the previous state and an action, and return the next state. Remember to return new state objects, instead of mutating the previous state. You can start with a single reducer, and as your app grows, split it off into smaller reducers that manage specific parts of the state tree. Because reducers are just functions, you can control the order in which they are called, pass additional data, or even make reusable reducers for common tasks such as pagination.

# 17. What is Webpack?



Webpack is a module bundler: A tool that can analyze your project’s structure, find JavaScript modules and other assets to bundle and pack them for the browser.

# 18. What is Typescript?



[TypeScript](http://www.typescriptlang.org/) is a superset of JavaScript which primarily provides optional static typing, classes and interfaces. One of the big benefits is to enable IDEs to provide a richer environment for spotting common errors as you type the code.

# 19. What are the popular React-specific linting?

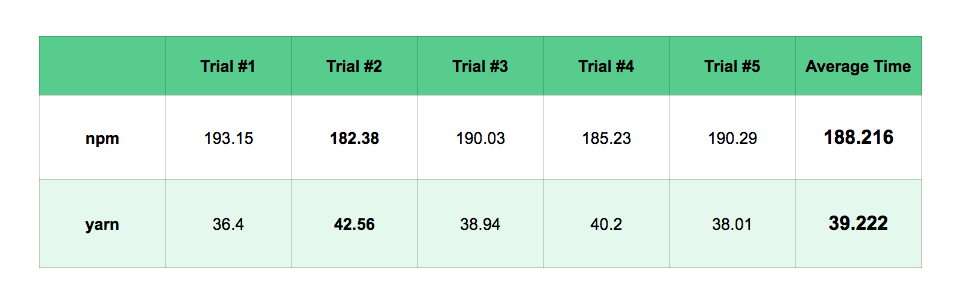
ESLint is a popular linter for various JavaScript projects. There are plugins available that analyze specific code styles. One of the most common for React is an npm package called eslint-plugin-react. Beside that, TSLint is an extensible static analysis tool that checks [TypeScript](https://github.com/Microsoft/TypeScript) code for readability, maintainability, and functionality errors.

# 20. Yarn vs NPM?

**Yarn and npm** are two well-known JavaScript package managers. If you’re not familiar with what a package manager does, it essentially is a way automate the process of installing, updating, configuring, and removing pieces of software (AKA packages) retrieved from a global registry.

## Speed:

Yarn proves to be consistently faster than npm.



# Other awesome features of Yarn

* Yarn uses **checksums** to verify the integrity of every installed package before executing code.
* Concise**lockfile format**, and a deterministic algorithm for installs. This means that Yarn is able to guarantee that an install that worked on one system will work exactly the same way on any other system. Isn’t that what you always wanted?
* **npm and bower** — Install any package from either npm or Bower and keep your package workflow the same.
* **Flat mode**— Resolve mismatching versions of dependencies to a single version to avoid creating duplicates.
* **Network Resilience** — A single request failing won’t cause an install to fail. Requests are retried upon failure.