React & Redux Interview Questions & Answers

NOTE:

Before going with REACT JS preparation

- **Please prepare JavaScript (ES5 and ECMASCRIPT 2015 and above, as the features are used during development)**
- **Learn about Webpack and JS Build Environment**

General Questions

What is Webpack?
Webpack in general is a module bundler, thanks to many plugins it provides although a lot more and it can be used to run tasks, clean build directories, check linting, handle typescript support, increase performance, provide chunking and a lot more.

How would you structure a React application?
This is an open question with many possible answers. The basic structure is usually module or feature based. We usually differentiate between UI and logic. There are many approaches to structure UI components with the most popular being atomic design. Data and business heavy applications use a more domain driven approach. The ideal combination for larger applications is having the domain logic separate and having the UI logic in an atomic structure. All this can be combined in features which are rendered on pages.

Explain concepts of functional programming.
Basic concepts are pure functions, side effects, data mutation and declarative over imperative. On top of that currying, compose and function composition are important concepts. Knowledge of libraries like RamdaJS provide useful utilities build applications in a more functional way. Functions being a first class citizen make it possible for JS to be really functional is important as well.

What is a REST API?
Important to know is, that a REST api provides a formal way of communicating with your back-end and sharing resources. It is based on the usage of http methods which indicate the intentions of the api call. These are POST, GET, PUT, PATCH and DELETE. REST is a standard and it enforces certain best practices on how to structure your urls, what kind of response format to send, how to handle security or use stateless communication. Common response formats are HYDRA or HAL and the most common authentication approach is via JWT tokens. An important concept for REST is also HATEOAS which is an approach on providing hypermedia information.

Explain WebSockets?
WebSockets is a protocol which provides full duplex communication in a single connection.

What is the difference between == and ===?
The usual answer would be one compares only the values while the other also compares the type. In this case, follow up question would be: Why null == undefined is true? As you can guess their values will never be equal.
A good answer is that == runs the loose equality check while === also compares the types. The loose equality check looks at the type on both sides and based on the types it decides what to do. The specifications states that if both sides of the comparison are null and undefined just return true.
So, it may just return true like in the case of null with undefined or it may try to convert one or the other side to the same type and compare their values depending on what is specified for a certain scenario.

What are closures and lexical scope?
A closure is when a function is able to remember and access its lexical scope even when that function is executed outside of that scope. A good example for closures is currying, a common accepted answer is also responding that an inner function returned by an outer function has access to its outer functions variables even after the outer function was called. Lexical scope is the scope model used by JavaScript which compared to dynamic scoping enables a lot of cool features. If someone also explains lexing time and goes into hoisting it is a big bonus.

What is data mutation and how to prevent it?
This is a bit tricky question as there are multiple correct answers and approaches. What we look for is understanding that arrays and objects can be passed around and since they are references, we are changing the original array or object. This leads to side effects and often to unwanted behaviour. The preferred way is never to mutate objects or arrays and when modifying, always create and return a new copy. We can take advantage of the spread operator, object.assign or libraries like ImmutableJS. Mentioning array.slice vs splice is a plus.

What is hoisting?
A common answer is: hoisting is a mechanism which moves all declarations to the top of the execution scope. A good answer is understanding that JavaScript compiles your code before execution. When the compiler enters an execution context, it will start lexing (splitting your code) and analysing the code while creating a list out of the declarations it finds. Each time the compiler encounters a declaration, it will create a binding to it and each time it encounters an assignment or evaluation it will ask the current scope for the binding, if it can't find the binding, it will go up until it reaches the global scope. If you have strict mode on, you will get an error and if you use the good old es5 it will create a new binding for you. This is why, you are able to assign to a variable which wasn't declared before. Anyway, after running through some steps, it will generate some compiled code which can be then executed :).

Explain inheritance in JavaScript.
JavaScript is a prototype based language and uses prototypal inheritance instead of the more common class inheritance, even though in ES6+ we use classes, they are just a syntactical sugar and they are still based on prototypes.
Prototypal inheritance is based on the prototype chain, in a simple example every object, objects instance, has a prototype. We can access it for instance trough the __proto__ property. This basically returns the objects parent. This way you can climb up the prototypal chain until you reach the top-level parent (object in this case) where the prototype is set to null. The advantage of the chain is, that if we call a property of an object, if it can't be found on the object, JavaScript will automatically look for the property on the prototype, if it can't find it on the prototype. It will look at the prototype of the prototype and so on until it reaches the top level and errors out.

What do you understand under asynchronous code in JS?
Synchronous code is code which is executed step by step, if we have a long running task we need to wait until it finishes executing before moving further and this basically blocks all other tasks if we don't have multithreading. In case of a 5s long AJAX call, we would block the browser for 5 seconds unable to do anything. This is solved by asynchronous code where tasks can be executed in a non-blocking way. This is ensured by the event loop and uses the call stack and callback queue to provide asynchronous behaviour. In simple terms, we can register asynchronous tasks with the event loop to be executed in the future. JavaScript provides a few ways to achieve asynchronous code: via promises, async/await or generators.

React and Redux Questions

Differentiate between Real DOM and Virtual DOM.

Real DOM vs Virtual DOM	
Real DOM	**Virtual DOM**
1. It updates slow.	1. It updates faster.
2. Can directly update HTML.	2. Can't directly update HTML.
3. Creates a new DOM if element updates.	3. Updates the JSX if element updates.
4. DOM manipulation is very expensive.	4. DOM manipulation is very easy.
5. Too much of memory wastage.	5. No memory wastage.

What is React?

- React is a front-end JavaScript library developed by Facebook in 2011.
- It follows the component-based approach which helps in building reusable UI components.
- It is used for developing complex and interactive web and mobile UI.
- Even though it was open-sourced only in 2015, it has one of the largest communities supporting it.

What are the features of React?

Major features of React are listed below:

i. It uses the **virtual DOM** instead of the real DOM.
ii. It uses **server-side rendering**.
iii. It follows **uni-directional data flow** or data binding.

List some of the major advantages of React.

Some of the major advantages of React are:

i. It increases the application's performance
ii. It can be conveniently used on the client as well as server side
iii. Because of JSX, code's readability increases
iv. React is easy to integrate with other frameworks like Meteor, Angular, etc
v. Using React, writing UI test cases become extremely easy

What are the limitations of React?

Limitations of React are listed below:

i. React is just a library, not a full-blown framework
ii. Its library is very large and takes time to understand
iii. It can be little difficult for the novice programmers to understand
iv. Coding gets complex as it uses inline templating and JSX

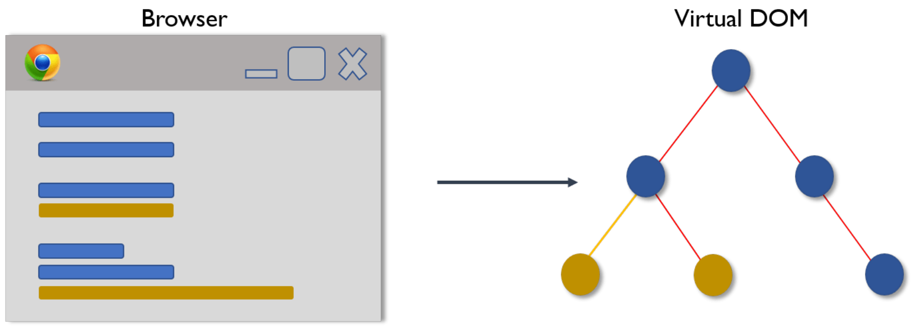
What is JSX?

JSX is a shorthand for JavaScript XML. This is a type of file used by React which utilizes the expressiveness of JavaScript along with HTML like template syntax. This makes the HTML file really easy to understand. This file makes applications robust and boosts its performance.

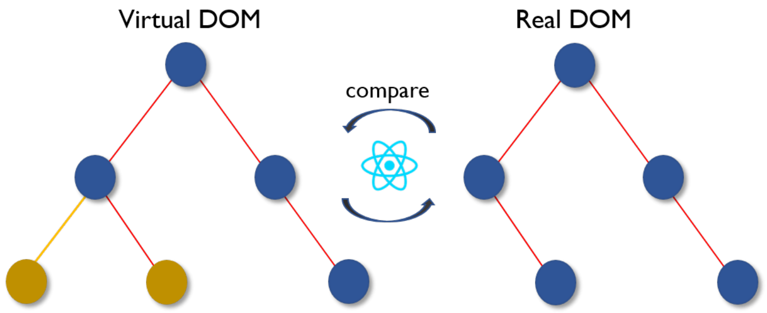
What do you understand by Virtual DOM? Explain its working.

A virtual DOM is a lightweight JavaScript object which originally is just the copy of the real DOM. It is a node tree that lists the elements, their attributes and content as Objects and their properties. React's render function creates a node tree out of the React components. It then updates this tree in response to the mutations in the data model which is caused by various actions done by the user or by the system. This Virtual DOM works in three simple steps.

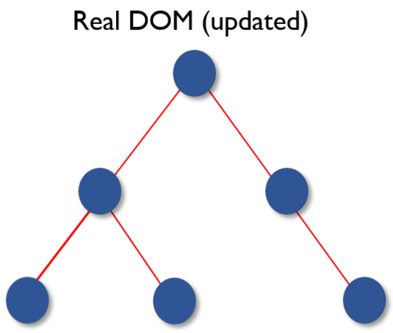
1. Whenever any underlying data changes, the entire UI is re-rendered in Virtual DOM representation.



2. Then the difference between the previous DOM representation and the new one is calculated.



3. Once the calculations are done, the real DOM will be updated with only the things that have actually changed.



4.

Why can't browsers read JSX?

Browsers can only read JavaScript objects but JSX in not a regular JavaScript object. Thus to enable a browser to read JSX, first, we need to transform JSX file into a JavaScript object using JSX transformers like Babel and then pass it to the browser.

How different is React's ES6 syntax when compared to ES5?

Syntax has changed from ES5 to ES6 in following aspects:

i. require vs import

```
// ES5
var React = require('react');

// ES6
import React from 'react';
```

ii. export vs exports

```
// ES5
module.exports = Component;

// ES6
```

```
export default Component;
```

iii. component and function

```
// ES5
var MyComponent = React.createClass({
   render: function() {
      return <h3>Hello World!</h3>;
   }
});

// ES6
class MyComponent extends React.Component {
   render() {
      return <h3>Hello World!</h3>;
   }
}
```

iv. props

```
// ES5
var Greeting = React.createClass({
   propTypes: {
      name: React.PropTypes.string
   },
   render: function () {
      return <h3>Hello, {this.props.name}</h3>;
   }
});

// ES6
import PropTypes from 'prop-types';

class Greeting extends React.Component {
   render() {
      return (
         <h1>Hello, {this.props.name}</h1>
      );
   }
}

Greeting.propTypes = {
   name: PropTypes.string
};
```

v. state

```
// ES5
var App = React.createClass({
   getInitialState: function() {
      return { name: 'World' };
   },
   render: function() {
      return <h3>Hello, {this.state.name}!</h3>;
   }
});

// ES6
class App extends React.Component {
   constructor() {
      super();
      this.state = { name: 'world' };
   }

   render() {
      return <h3>Hello, {this.state.name}!</h3>;
   }
}
```

How is React different from Angular?

React vs Angular		
TOPIC	**REACT**	**ANGULAR**
1. ARCHITECTURE	Only the View of MVC	Complete MVC
2. RENDERING	Server-side rendering	Client-side rendering
3. DOM	Uses virtual DOM	Uses real DOM
4. DATA BINDING	One-way data binding	Two-way data binding
5. DEBUGGING	Compile time debugging	Runtime debugging
6. AUTHOR	Facebook	Google

What do you understand from "In React, everything is a component."
Components are the building blocks of a React application's UI. These components split up the entire UI into small independent and reusable pieces. Then it renders each of these components independent of each other without affecting the rest of the UI.

Explain the purpose of render() in React.
Each React component must have a **render()** mandatorily. It returns a single React element which is the representation of the native DOM component. If more than one HTML element needs to be rendered, then they must be grouped together inside one enclosing tag such as **<form>, <group>,<div>** etc. This function must be kept pure i.e., it must return the same result each time it is invoked.

What is Props?
Props is the shorthand for Properties in React. They are read-only components which must be kept pure i.e. immutable. They are always passed down from the parent to the child components throughout the application. A child component can never send a prop back to the parent component. This help in maintaining the unidirectional data flow and are generally used to render the dynamically generated data.

What is a state in React and how is it used?
States are the heart of React components. States are the source of data and must be kept as simple as possible. Basically, states are the objects which determine components rendering and behavior. They are mutable unlike the props and create dynamic and interactive components. They are accessed via **this.state().**

What is arrow function in React? How is it used?

Arrow functions are more of brief syntax for writing the function expression. They are also called *'fat arrow'* (=>) the functions. These functions allow to bind the context of the components properly since in ES6 auto binding is not available by default. Arrow functions are mostly useful while working with the higher order functions.

Differentiate between stateful and stateless components.

Stateful vs Stateless	
Stateful Component	**Stateless Component**
1. Stores info about component's state change in memory	1. Calculates the internal state of the components
2. Have authority to change state	2. Do not have the authority to change state
3. Contains the knowledge of past, current and possible future changes in state	3. Contains no knowledge of past, current and possible future state changes
4. Stateless components notify them about the requirement of the state change, then they send down the props to them.	4. They receive the props from the Stateful components and treat them as callback functions.

What are the different phases of React component's lifecycle?

There are three different phases of React component's lifecycle:

i. *Initial Rendering Phase:* This is the phase when the component is about to start its life journey and make its way to the DOM.
ii. *Updating Phase:* Once the component gets added to the DOM, it can potentially update and re-render only when a prop or state change occurs. That happens only in this phase.
iii. *Unmounting Phase:* This is the final phase of a component's life cycle in which the component is destroyed and removed from the DOM.

Explain the lifecycle methods of React components in detail.

Some of the most important lifecycle methods are:

i. ***componentWillMount()*** – Executed just before rendering takes place both on the client as well as server-side.
ii. ***componentDidMount()*** – Executed on the client side only after the first render.
iii. ***componentWillReceiveProps()*** – Invoked as soon as the props are received from the parent class and before another render is called.
iv. ***shouldComponentUpdate()*** – Returns true or false value based on certain conditions. If you want your component to update, return **true** else return **false**. By default, it returns false.
v. ***componentWillUpdate()*** – Called just before rendering takes place in the DOM.
vi. ***componentDidUpdate()*** – Called immediately after rendering takes place.
vii. ***componentWillUnmount()*** – Called after the component is unmounted from the DOM. It is used to clear up the memory spaces.

What is an event in React?

In React, events are the triggered reactions to specific actions like mouse hover, mouse click, key press, etc. Handling these events are similar to handling events in DOM elements. But there are some syntactical differences like:

i. Events are named using camel case instead of just using the lowercase.
ii. Events are passed as functions instead of strings.

The event argument contains a set of properties, which are specific to an event. Each event type contains its own properties and behavior which can be accessed via its event handler only.

How do you modularize code in React?
We can modularize code by using the export and import properties. They help in writing the components separately in different files.
What are synthetic events in React?
Synthetic events are the objects which act as a cross-browser wrapper around the browser's native event. They combine the behavior of different browsers into one API. This is done to make sure that the events show consistent properties across different browsers.

What do you understand by refs in React?
Refs is the short hand for References in React. It is an attribute which helps to store a reference to a particular React element or component, which will be returned by the components render configuration function. It is used to return references to a particular element or component returned by render(). They come in handy when we need DOM measurements or to add methods to the components.

List some of the cases when you should use Refs.
Following are the cases when refs should be used:
- When you need to manage focus, select text or media playback
- To trigger imperative animations
- Integrate with third-party DOM libraries

How are forms created in React?
React forms are similar to HTML forms. But in React, the state is contained in the state property of the component and is only updated via setState(). Thus, the elements can't directly update their state and their submission is handled by a JavaScript function. This function has full access to the data that is entered by the user into a form.

What are Higher Order Components (HOC)?
Higher Order Component is an advanced way of reusing the component logic. Basically, it's a pattern that is derived from React's compositional nature. HOC are custom components which wrap another component within it. They can accept any dynamically provided child component but they won't modify or copy any behavior from their input components. You can say that HOC are 'pure' components.

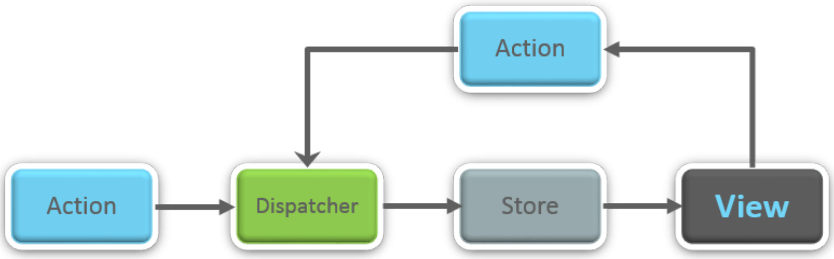
What can you do with HOC?
HOC can be used for many tasks like:
- Code reuse, logic and bootstrap abstraction
- Render High jacking
- State abstraction and manipulation
- Props manipulation

What are Pure Components?
Pure components are the simplest and fastest components which can be written. They can replace any component which only has a **render().** These components enhance the simplicity of the code and performance of the application.

What were the major problems with MVC framework?
Following are some of the major problems with MVC framework:
- DOM manipulation was very expensive
- Applications were slow and inefficient
- There was huge memory wastage
- Because of circular dependencies, a complicated model was created around models and views

Explain Flux.

Flux is an architectural pattern which enforces the uni-directional data flow. It controls derived data and enables communication between multiple components using a central Store which has authority for all data. Any update in data throughout the application must occur here only. Flux provides stability to the application and reduces run-time errors.



What is Redux?

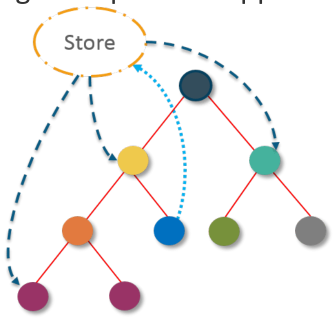
Redux is one of the hottest libraries for front-end development in today's marketplace. It is a predictable state container for JavaScript applications and is used for the entire applications state management. Applications developed with Redux are easy to test and can run in different environments showing consistent behavior.

What are the three principles that Redux follows?

i. ***Single source of truth:*** The state of the entire application is stored in an object/ state tree within a single store. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.
ii. ***State is read-only:*** The only way to change the state is to trigger an action. An action is a plain JS object describing the change. Just like state is the minimal representation of data, the action is the minimal representation of the change to that data.
iii. ***Changes are made with pure functions:*** In order to specify how the state tree is transformed by actions, you need pure functions. Pure functions are those whose return value depends solely on the values of their arguments.

What do you understand by "Single source of truth"?

Redux uses 'Store' for storing the application's entire state at one place. So, all the component's state is stored in the Store and they receive updates from the Store itself. The single state tree makes it easier to keep track of changes over time and debug or inspect the application.



List down the components of Redux.

Redux is composed of the following components:

i. **Action** – It's an object that describes what happened.
ii. **Reducer** – It is a place to determine how the state will change.
iii. **Store** – State/ Object tree of the entire application is saved in the Store.
iv. **View** – Simply displays the data provided by the Store.

Explain the role of Reducer.

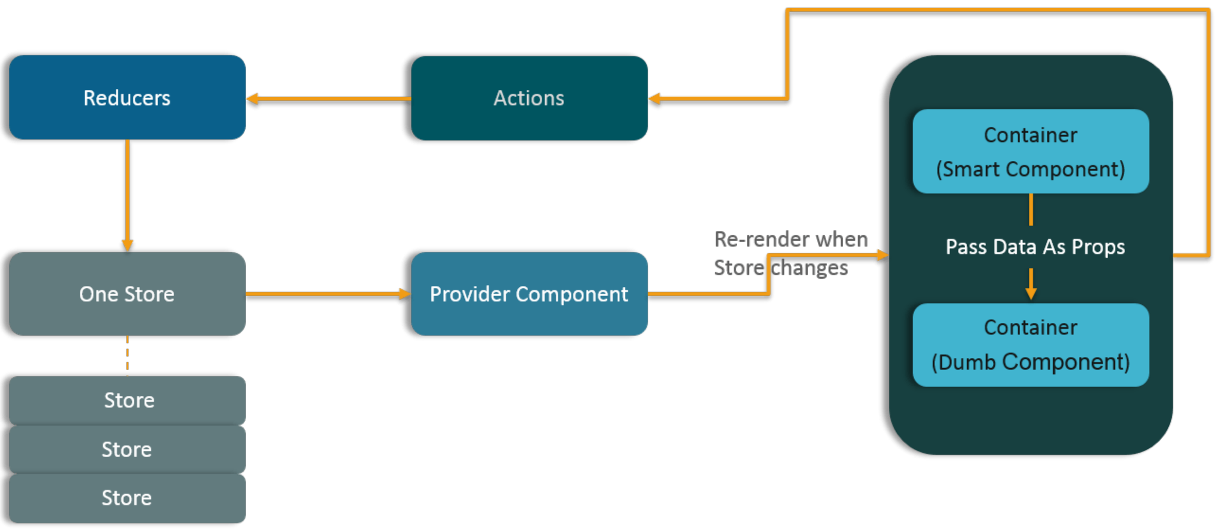
Reducers are pure functions which specify how the application's state changes in response to an ACTION. Reducers work by taking in the previous state and action, and then it returns a new state. It determines what sort of update needs to be done based on the type of the action, and then returns new values. It returns the previous state as it is, if no work needs to be done.

How are Actions defined in Redux?

Actions in React must have a type property that indicates the type of ACTION being performed. They must be defined as a String constant and you can add more properties to it as well. In Redux, actions are created using the functions called Action Creators. Below is an example of Action and Action Creator:

```
function addTodo(text) {
  return {
    type: ADD_TODO,
    text
  }
}
```

Show how the data flows through Redux?



What is the significance of Store in Redux?

A store is a JavaScript object which can hold the application's state and provide a few helper methods to access the state, dispatch actions and register listeners. The entire state/ object tree of an application is saved in a single store. As a result of this, Redux is very simple and predictable. We can pass middleware to the store to handle the processing of data as well as to keep a log of various actions that change the state of stores. All the actions return a new state via reducers.

How is Redux different from Flux?

Flux vs Redux	
Flux	**Redux**
1. The Store contains state and change logic	1. Store and change logic are separate
2. There are multiple stores	2. There is only one store
3. All the stores are disconnected and flat	3. Single store with hierarchical reducers
4. Has singleton dispatcher	4. No concept of dispatcher

5. React components subscribe to the store	5. Container components utilize connect
6. State is mutable	6. State is immutable

What are the advantages of Redux?

Advantages of Redux are listed below:

- **Predictability of outcome –** Since there is always one source of truth, i.e. the store, there is no confusion about how to sync the current state with actions and other parts of the application.
- **Maintainability –** The code becomes easier to maintain with a predictable outcome and strict structure.
- **Server-side rendering –** You just need to pass the store created on the server, to the client side. This is very useful for initial render and provides a better user experience as it optimizes the application performance.
- **Developer tools –** From actions to state changes, developers can track everything going on in the application in real time.
- **Community and ecosystem –** Redux has a huge community behind it which makes it even more captivating to use. A large community of talented individuals contribute to the betterment of the library and develop various applications with it.
- **Ease of testing –** Redux's code is mostly functions which are small, pure and isolated. This makes the code testable and independent.
- **Organization –** Redux is precise about how code should be organized, this makes the code more consistent and easier when a team works with it.

What is React Router?

React Router is a powerful routing library built on top of React, which helps in adding new screens and flows to the application. This keeps the URL in sync with data that's being displayed on the web page. It maintains a standardized structure and behavior and is used for developing single page web applications. React Router has a simple API.

Why is switch keyword used in React Router v4?

Although a **<div>** is used to encapsulate multiple routes inside the Router. The 'switch' keyword is used when you want to display only a single route to be rendered amongst the several defined routes. The **<switch>** tag when in use matches the typed URL with the defined routes in sequential order. When the first match is found, it renders the specified route. Thereby bypassing the remaining routes.

Why do we need a Router in React?

A Router is used to define multiple routes and when a user types a specific URL, if this URL matches the path of any 'route' defined inside the router, then the user is redirected to that particular route. So basically, we need to add a Router library to our app that allows creating multiple routes with each leading to us a unique view.

List down the advantages of React Router.
Few advantages are:

i. Just like how React is based on components, in React Router v4, the API is *'All About Components'*. A Router can be visualized as a single root component (**<BrowserRouter>**) in which we enclose the specific child routes (**<route>**).
ii. No need to manually set History value: In React Router v4, all we need to do is wrap our routes within the **<BrowserRouter>** component.

iii. The packages are split: Three packages one each for Web, Native and Core. This supports the compact size of our application. It is easy to switch over based on a similar coding style.

What are stateless components?
Stateless components (a flavor of “reusable” components) are nothing more than pure functions that render DOM based solely on the properties provided to them.
Below given component has no need for any internal state — let alone a constructor or lifecycle handlers. The output of the component is purely a function of the properties provided to it.

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

ReactDOM.render(
  <StatelessCmp name="Someone" birthday="23/10/1985" />,
  document.getElementById('main')
);
```

How is React different from angular and VUE js?
Angular – developed by Google, angular is a typescript-based JavaScript application framework. It is also known as Super-heroic JavaScript MVW Framework. It was developed with the motive to encounter the challenges of creating single page applications. There are several versions of angular such as Angular 2+, Angular 2 or ng2. Angular is the rewritten, mostly incompatible successor to AngularJS which means AngularJS is the oldest framework.

React– React was developed by Facebook in March 2013. It is a JavaScript library that is used for building user interfaces. React creates large web applications and also provides speed, scalability, and simplicity.

Vue Js- Launched in February 2014, Vue is the most famous and rapidly growing framework in JS field. Vue is an intuitive, fast and composable MVVM for building interactive interfaces. It is extremely adaptable, and several JavaScript libraries make use of this. Vue is also a web application framework that helps in making advanced single page applications.

Why to use React js?

- Reusable components, using mixins (that allow overlap only on lifecycle methods, and have a predictable execution order).
- Hierarchical components (Components that appear within another component in the mock should appear as a child in the hierarchy).
- Uses inheritance
- High Cohesion, Loose coupling
- Virtual, shadow DOM
- It's SEO friendly - React can run on the server and virtual DOM render data and return the regular web page.
- Easy but Powerful JavaScript library
- Tools & Extensions

What is the difference between controlled and uncontrolled components?
With controlled components, data are handled by React while in uncontrolled components the data is in the regular DOM and we can access it using refs.

What is a container component?
Containers are components which usually provide data and logic and are not concerned with rendering the view. Containers are in most cases connected components and get the data from the state and pass down actions bound to dispatch.

What are selectors? Why would you use reselect or a memoization library?
Selectors are functions which accept the state and return a portion of it while applying calculations, transformations, mappings, filtering etc. This way the logic of how to retrieve data for a specific view is encapsulated in a selector. Since many of the mentioned operations are expensive, when calling the selector again without state change, you want to skip the expensive operations as they will return the same results and hence the usage of reselect. Reselect will return the results from the Cache in case arguments didn't change.

What are approaches of handling side effects in redux?
Side effects are mostly async calls and are handled by redux thunks or redux sagas.

What are actions and action creators?
An action is an object with a mandatory type field, it usually has also a payload which are the data related to the action. An action creator is a function which returns an action.

How would you optimise the performance of a React application?
The most expensive task in a React app is the update of the DOM. The basic optimisation is to reduce how many times a component re-renders. This can be achieved by using componentShouldUpdate, using PureComponent or memoization libraries like reselect. Reducing the size of the final JS file also helps improving performance and we can use dynamic imports and chunks for this.

Do you know what the reconciliation algorithm is?
It is the algorithm responsible for figuring out what changed between re-renders and how to update the actual DOM. It is basically a diffing algorithm. The latest addition of improvements on the core algorithm is called React Fiber.

Do you know the Context API?
The context api is a feature which allows sharing data without passing it down in the component tree. With the latest API, in some cases it is a nice replacement for state management libraries. Many libraries use the api including Redux, styled components theme provider and some other UI libraries.

How do you handle method binding in components?
There are more ways, we can bind the method in render, in componentDidMount or use the preferred way, which are arrow functions.

What are new features in React 16+?
React introduced Fiber, Portals, Error boundaries, the new Context API and deprecated for instance componentWillUpdate. The latest addition will be Hooks, which are soon to come.