**1. What is the InteractionManager and how is it used in React Native? Why is it important?**

**Answer**: The InteractionManager is the native module responsible for deferring the execution of a function until an “interaction” has finished. We can call InteractionManager.runAfterInteractions(() => {...}) to handle this deferral.

We can also register our own interactions. InteractionManager is very important because React Native has two threads. There is a JavaScript UI thread which handles drawing updates to the screen, and another thread used for all tasks not on the UI thread. Since there is only one thread for making UI updates, it can get overloaded and drop frames, especially during things

like navigation screen animations. We use the InteractionManager to ensure that our function is executed after these animations occur so that we do not drop frames on the UI thread. Trying to draw a new screen while it is being animated is often too much for the thread to handle.

**2. Explain some of the fundamental tradeoffs between building with React Native and building a “true” native app?**

**Answer**: React Native has exploded in popularity because the tradeoffs it provides make sense to many companies and teams. But building an app in React Native is not always the right choice.

React Native makes sense when a team is building a product that does not need extremely high performance.

The limitations of the asynchronous bridge are a bottleneck for things like 3D games, and games with lots of particle updates. Apps that rely on deep interactions with low-level APIs, or need large amounts of native code, might be easier to build as native apps.

React Native makes sense when an existing team is already proficient in JavaScript and/or has an existing React application built on the web or another platform. The “learn once, write everywhere” ethos advocated for by Facebook is very useful when diversifying a product across platforms. Hiring becomes easier, since JavaScript developers are plentiful, and you don’t need to seek out native specialists.

React Native is partially open source, partially closed source. Facebook maintains a private repository of React Native code that is used in their apps. When code from the private repo can be split off so that it contains nothing proprietary, it is often merged into the open source codebase. This leaves users of React Native with the classic tradeoff of open source software: There are often bugs—React Native is still in alpha form—and improvements can be spotty. On the other hand, motivated teams can make contributions to the source code and implement fixes and features that they need. Depending on a team’s resources and product roadmap,

relying on open source may be the right choice.

**3. What is the relationship between React Native and React?**

**Answer**: React Native is built using React. React, at its core, is a library for “diffing” a virtual DOM and rendering this DOM to a screen with minimal operations. React, by default, does not have an opinion about which nodes are in its virtual DOM tree. Instead, it simply has algorithms that can determine changes in the tree and rerender. React on the web provides its own node primitives (<div>, <span>, etc), which are the building blocks for web applications. But new node primitives can be defined, as React Native has done.

React Native defines its own primitives (<View>, <Image>, etc) which do not render HTML elements but instead map to native views, like UIView and UIImageView. It implements a bridge that allows the JavaScript runtime to communicate asynchronously with the native runtime. React itself provides the tree diffing and rendering infrastructure that allows React Native to work.

**4. What is FlexBox in React Native?**

**Answer**: Flexbox in React Native works in the same way as it does in CSS on the web, with a few exceptions. It is used to provide a consistent layout on different screen sizes.

**5. What are props and state in React Native?**

**Answer**: In a React native, different props of various component instances will issue the value of the props to render the views. The other meaning of the props is properties. A parent component passes a prop to the child component. The values defined in the props are utilized by the child components. It becomes very easy to program a reusable code by the implementation of the props.

To handle any data that is changeable, a component named state is used. A state is aspects of the React Native that may change or differ in a component. Each component contains different state values. Based on these values of the state, the UI is also changed.

**6. Explain what is use of webView in React Native?**

**Answer**: WebView is a React Native is a component that is implemented to load a web page or web content. It is imported from the core of the react-native library. The WebView is replaced from the inbuilt core react native and is then placed inside the react-native webview library. In other words, WebView is a bridge that connects web platforms with the react native and gives the users different options to create connections to an application that is running on the web.

WebView can also control the navigation and load an application into the web page. Webview is an element of the React Native that needs no installation or configuration.

**7. How are Hot Reloading and Live Reloading in React Native different?**

**Answer**: Live reloading reloads or refreshes the entire app when a file changes. For example, if you were four links deep into your navigation and saved a change, live reloading would restart the app and load the app back to the initial route.

Hot reloading only refreshes the files that were changed without losing the state of the app. For example, if you were four links deep into your navigation and saved a change to some styling, the state would not change, but the new styles would appear on the page without having to navigate back to the page you are on because you would still be on the same page.

**8. What are React Hooks?**

**Answer**: Hooks are a new addition in React 16.8. They let you use state and other React features without writing a class. With Hooks, you can extract stateful logic from a component so it can be tested independently and reused. Hooks allow you to reuse stateful logic without changing your component hierarchy. This makes it easy to share Hooks among many components or with the community.

**9. What is the difference between ShadowDOM and VirtualDOM?**

**Answer**: The Shadow DOM is a browser technology designed primarily for scoping variables and CSS in web components. The virtual DOM is a concept implemented by libraries in JavaScript on top of browser APIs.

**10. How React Native deals with a variety of screen sizes?**

**Answer**: React Native deals efficiently with different screen sizes using the following:

Flexbox provides uniform layout on screen with different sizes using its three properties: justify-content, flex-direction, alignItems. Dimensions can style the page accurately on screen with different sizes. Pixel ratio class can be utilized to access the pixel density of the device and render the image accordingly. The aspect ratio sets the height. Scrollview, a scrolling container that contains multiple components and can be scrolled both vertically and horizontally.