Answer:

1. React Native is cross-platform, meaning that you can create one application that can be used on various platforms (Android and iOS).

2. 7

3. 4 Types of Lifecycle React Native:

a. Initialization is a React Native cycle to set State and Props before the application starts

b. Mounting is a cycle when the application has just been opened. There are 2 types, namely componentDidMount() when loading the application before rendering is done. componentWillMount is the cycle after the rendering process is done.

c. Updating is when you change data that has been mounted.

d. Unmount is the process of destroying or destroying the previously defined components.

4. React.PureComponent is similar to React.Component. The difference between them is that React.Component doesn’t implement shouldComponentUpdate(), but React.PureComponent implements it with a shallow prop and state comparison.

Scroll view It does not provide any memory management, It loads all the content at once, It results in slow rendering and increased memory usage.

Flatlist It provides automatic memory management, It loads content as the window scrolled, It does not affect the rendering speed.

5. yes! Class components are JavaScript ES2015 classes that extend a base class from React called Component.

Functional components are simpler. They don’t manage their own state or have access to the lifecycle methods provided by React Native. They are literally plain old JavaScript functions, and are sometimes called stateless components.

6. yes! yes! Redux is a standalone state management library, which can be used with any library or framework whether it's React, React Native or any other view library

7. yes!

1. Redux
2. Reduxjs toolkit
3. Async storage
4. React native firebase
5. React navigation
6. Sweet alert
7. axios

8. no

9. yes

Bonus Answer:

1. The main difference between the == and === operator in javascript is that the == operator does the type conversion of the operands before comparison, whereas the === operator compares the values as well as the data types of the operands.
2. The first and most obvious difference between arrow functions and regular functions is their syntax. Not only do they look different, but arrow functions also provide an implicit return shorthand and allow parenthesis around a single argument to be omitted.
3. a. let array = oldArray, if you change the value in the array, then oldArray will also change

b. b. let array = […oldArray], if you change the value in the array, then oldArray will not change

1. function call API

const [isLoading, setLoading] = useState(true);

const [data, setData] = useState([]);

const getMovies = async () => {

try {

const response = await fetch(https://my-json-server.typicode.com/horizon-code-academy/fake-movies-api/movies');

const json = await response.json();

setData(json);

} catch (error) {

console.error(error);

} finally {

setLoading(false);

}

}

useEffect(() => {

getMovies();

}, []);