React Native

Since you already know **React and Next.js**, you don't need to relearn **state management, JSX, hooks (`useState`, `useEffect`), props, or general component structure**. Below is an optimized **React Native Roadmap (2025)** focusing **only on what's different** from React/Next.js.

🔷 Phase 1: React Native Core Differences (1-2 Weeks)

✅ Objective: Learn what makes React Native different from React.

🚀 1. React Native Setup & Environment

- No browser! Apps run on mobile devices.
- **Choose your setup**:
  - **Expo CLI** (Easier, faster, but has some limitations)
  - **React Native CLI** (More control, closer to native development)
- Install dependencies:

  OR (if using React Native CLI)

```
npm install -g expo-cli
expo init MyProject
```

```
npx react-native init MyProject
```

🚀 2. No HTML & No CSS

- Instead of `div`, `span`, `p`, use:
  - `<View>` (like `<div>`)
  - `<Text>` (like `<p>`, must be used for all text)
  - `<Image>` (like `<img>`)

- No `className`! Instead, use **StyleSheet API**:

```
import { StyleSheet, View, Text } from 'react-native';

const App = () => (
  <View style={styles.container}>
    <Text style={styles.text}>Hello, React Native!</Text>
  </View>
);

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', alignItems: 'center' },
  text: { fontSize: 20, color: 'blue' },
});

export default App;
```

🚀 3. Flexbox is Default for Layout

- Uses **Flexbox** for layout (like in web CSS).
- `flexDirection` defaults to **column** (instead of row like in web CSS).
- No `px`, everything is unitless.

🔷 Phase 2: Navigation & API Differences (2-3 Weeks)

✅ Objective: Learn how navigation and API handling differ from web apps.

🚀 4. Navigation (No React Router)

- Instead of `react-router-dom`, use **React Navigation**:

```
npm install @react-navigation/native
```

- **Types of navigation:**
  - **Stack Navigation** (like browser history for mobile)

- **Tab Navigation** (bottom tabs)
- **Drawer Navigation** (side menu)

Example of **Stack Navigation**:

```
import { createStackNavigator } from '@react-navigation/stack';
import { NavigationContainer } from '@react-navigation/native';

const Stack = createStackNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Profile" component={ProfileScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

🚀 5. API Calls Work the Same, But Need Async Handling for UI

- `fetch` and **Axios** work the same.
- Loading states are more critical because slow responses can make the app feel frozen.
- Example:

```
import { useState, useEffect } from 'react';
import { View, Text, ActivityIndicator } from 'react-native';

const MyComponent = () => {
  const [data, setData] = useState(null);
  const [loading, setLoading] = useState(true);

  useEffect(() => {
    fetch('https://api.example.com/data')
      .then(res => res.json())
```

```
      .then(data => {
        setData(data);
        setLoading(false);
      });
  }, []);

  if (loading) return <ActivityIndicator size="large" color="blue" />;

  return <Text>{data.title}</Text>;
};
```

🔷 Phase 3: Native Features (3-4 Weeks)

✅ Objective: Learn how to use mobile-specific features.

🚀 6. Handling Touch & Gestures

- No `<button>`, use **Touchable components**:
  - `<TouchableOpacity>` (common for buttons)
  - `<Pressable>` (more flexible)
  - `<TouchableWithoutFeedback>` (dismiss keyboard)

Example:

```
<TouchableOpacity onPress={() => console.log('Pressed!')}>
  <Text>Click Me</Text>
</TouchableOpacity>
```

🚀 7. Accessing Native Device Features

- **Camera & Photos:** `expo-camera`, `expo-image-picker`
- **Location Services:** `expo-location`
- **Push Notifications:** `expo-notifications`
- Example (Getting user location):

```
import * as Location from 'expo-location';

const getLocation = async () => {
  const { status } = await Location.requestForegroundPermissionsAsync
();
  if (status !== 'granted') return console.log('Permission denied');

  const location = await Location.getCurrentPositionAsync({});
  console.log(location);
};
```

🚀 8. AsyncStorage (Local Storage)

- No `localStorage` or `sessionStorage`
- Use `AsyncStorage` instead:

```
npm install @react-native-async-storage/async-storage
```

- Example:

```
import AsyncStorage from '@react-native-async-storage/async-storag
e';

const storeData = async () => {
  await AsyncStorage.setItem('userToken', '12345');
};

const getData = async () => {
  const token = await AsyncStorage.getItem('userToken');
  console.log(token);
};
```

🔷 Phase 4: Performance & Deployment (3-4 Weeks)

✅ Objective: Optimize performance & deploy your app.

🚀 9. Performance Optimization

- **FlatList** for large lists (instead of `.map()`):

```
<FlatList
  data={myData}
  keyExtractor={item => item.id}
  renderItem={({ item }) => <Text>{item.name}</Text>}
/>
```

- Memoization (`useMemo`, `useCallback`) to avoid re-renders.

🚀 10. Debugging & Testing

- Debugging with **React Native Debugger**
- **Testing Libraries:**
  - Unit Tests → Jest
  - UI Tests → React Native Testing Library

🚀 11. Deployment

- **Android:**

```
expo build:android
```

- **iOS:**

```
expo build:ios
```

- Use **Expo EAS** for Over-the-Air (OTA) updates.

🔥 Final Thoughts

What's Different from React/Next.js?

✅ **No HTML elements** (use `View`, `Text`, etc.)

✅ **No CSS, No Tailwind (by default)** (use `StyleSheet API`)

✅ **Different Navigation** (use `react-navigation` instead of React Router)

✅ **Native APIs** (camera, location, storage)

✅ **Touchable interactions** instead of `<button>`

✅ **Different debugging & deployment process**

🚀 Final Step: Build a Real App!

Once you've mastered the above, build a **real-world project** like:

- **E-commerce App** (Product list, cart, checkout)
- **Chat App** (Firebase authentication, real-time messages)
- **Fitness Tracker** (Tracks steps, stores workout data)

Since you already know React/Next.js, you can **become proficient in React Native in 6-8 weeks** by focusing only on these differences.

Would you like recommendations for **specific courses** or **boilerplate projects**? 🚀