

What is React Native?



- React Native is a framework that allows you to build mobile applications using React (JavaScript/TypeScript).
- Cross-platform development means writing code once that can run on multiple platforms.
- React Native is a cross-platform framework because you can write one codebase in JavaScript and deploy it to both iOS and Android.

Back in the old days?



- Macbook required
- XCode as your IDE
- Had to learn Android SDK and its frameworks
- Android Studio/Eclipse as the IDE

React Native by itself is not enough!

(screenshot from docs) 🙌



The screenshot shows the React Native documentation website. The header includes the React Native logo, version 0.78, and navigation links for Development, Contributing, Community, Showcase, and Blog. A search bar is also present. The left sidebar lists the table of contents, with 'Get Started with React Native' highlighted under the 'Environment setup' section. The main content area is titled 'Get Started with React Native' and contains the following text:

React Native allows developers who know React to create native apps. At the same time, native developers can use React Native to gain parity between native platforms by writing common features once.

We believe that the best way to experience React Native is through a **Framework**, a toolbox with all the necessary APIs to let you build production ready apps.

You can also use React Native without a Framework, however we've found that most developers benefit from using a React Native Framework like **Expo**. Expo provides features like file-based routing, high-quality universal libraries, and the ability to write plugins that modify native code without having to manage native files.

Go-to Framework for React Native?

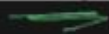


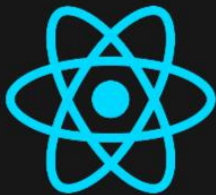
Expo!

They're not the same thing!



React Native





VS



React Native

1- Component Structure

```
import React from 'react';

const WebComponent = () => {
  return (
    <div className="container">
      <h1>Welcome</h1>
      <p>This is a web component</p>
    </div>
  );
};
```

```
// React Native
import React from 'react';
import { View, Text } from 'react-native';

const NativeComponent = () => {
  return (
    <View style={styles.container}>
      <Text style={styles.title}>Welcome</Text>
      <Text>This is a native component</Text>
    </View>
  );
};
```

2- Styling

```
// React Web - Using CSS
const WebStyledComponent = () => {
  return (
    <div className="card">
      
      <div className="content">
        <h2>John Doe</h2>
        <p>Web Developer</p>
      </div>
    </div>
  );
};
```

```
/* CSS file */
.card {
  padding: 16px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}

.profile-image {
  width: 100px;
  height: 100px;
  border-radius: 50%;
}
```

```
// React Native - Using StyleSheet
import { StyleSheet } from 'react-native';

const NativeStyledComponent = () => {
  return (
    <View style={styles.card}>
      <Image
        source={require('./profile.jpg')}
        style={styles.profileImage}
      />
      <View style={styles.content}>
        <Text style={styles.name}>John Doe</Text>
        <Text style={styles.role}>Mobile Developer</Text>
      </View>
    </View>
  );
};
```

```
const styles = StyleSheet.create({
  card: {
    padding: 16,
    borderRadius: 8,
    shadowColor: 'black',
    shadowOffset: { width: 0, height: 2 },
    shadowOpacity: 0.1,
    shadowRadius: 4,
  },
  profileImage: {
    width: 100,
    height: 100,
    borderRadius: 50,
  },
});
```

3- Event handling

```
// React Web
const WebButton = () => {
  const handleClick = (event) => {
    console.log('Button clicked:', event);
  };

  return (
    <button
      onClick={handleClick}
      onMouseOver={() => console.log('Mouse over')}
    >
      Click me
    </button>
  );
};
```

```
// React Native
const NativeButton = () => {
  const handlePress = (event) => {
    console.log('Button pressed:', event);
  };

  return (
    <TouchableOpacity
      onPress={handlePress}
      onLongPress={() => console.log('Long press')}
    >
      <Text>Press me</Text>
    </TouchableOpacity>
  );
};
```

4- Lists

```
// React Web
const WebList = ({ items }) => {
  return (
    <div className="list-container">
      {items.map((item) => (
        <div key={item.id} className="list-item">
          <h3>{item.title}</h3>
          <p>{item.description}</p>
        </div>
      ))}
    </div>
  );
};
```

```
// React Native
const NativeList = ({ items }) => {
  const renderItem = ({ item }) => (
    <View style={styles.listItem}>
      <Text style={styles.itemTitle}>{item.title}</Text>
      <Text>{item.description}</Text>
    </View>
  );

  return (
    <FlatList
      data={items}
      renderItem={renderItem}
      keyExtractor={(item) => item.id}
      onEndReached={() => console.log("End reached")}
      onEndReachedThreshold={0.5}
    />
  );
};
```

5- Forms

```
// React Web
const WebForm = () => {
  const [formData, setFormData] = useState({
    username: "",
    email: "",
  });

  return (
    <form onSubmit={(e) => e.preventDefault()}>
      <input
        type="text"
        value={formData.username}
        onChange={(e) =>
          setFormData({
            ...formData,
            username: e.target.value,
          })
        }
        placeholder="Username"
      />
      <button type="submit">Submit</button>
    </form>
  );
};
```

```
// React Native
const NativeForm = () => {
  const [formData, setFormData] = useState({
    username: "",
    email: "",
  });

  return (
    <View style={styles.form}>
      <TextInput
        style={styles.input}
        value={formData.username}
        onChangeText={(text) =>
          setFormData({
            ...formData,
            username: text,
          })
        }
        placeholder="Username"
      />
      <TouchableOpacity onPress={() => console.log("Submit")}>
        <Text>Submit</Text>
      </TouchableOpacity>
    </View>
  );
};
```