React Native for Flutter Developers

Building mobile apps with React and JavaScript

What is React Native?

- Framework for building **native mobile apps** using React
- Write once, run on **iOS, Android, and Web**
- Uses **native components** (not WebView)
- JavaScript runtime + Native rendering

Flutter equivalent: Cross-platform mobile framework

JavaScript Engine + Native UI Rendering

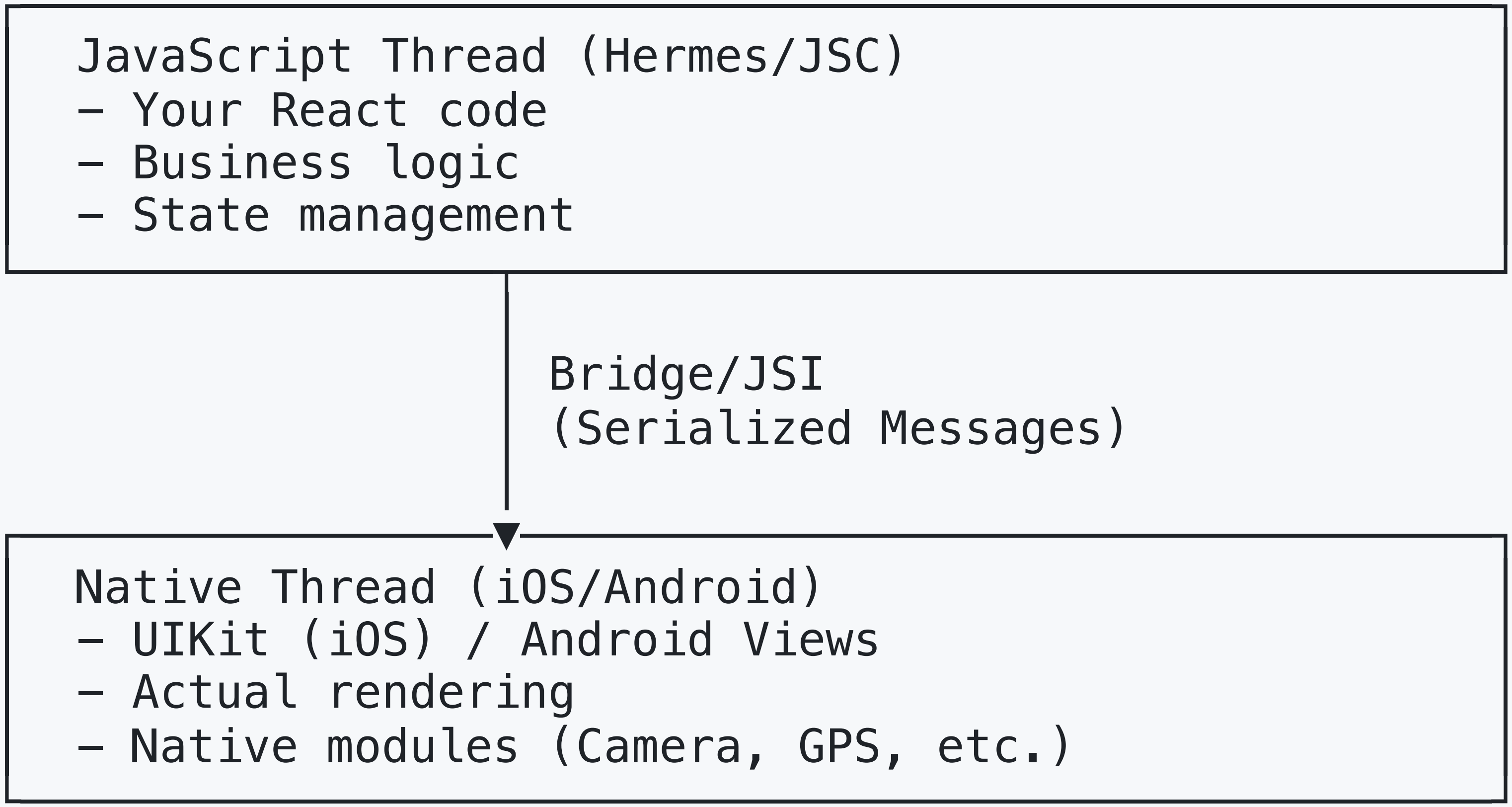
The JavaScript Engine

React Native uses **different JavaScript engines** depending on the platform:

```
iOS            → JavaScriptCore (JSC)
Android        → Hermes (optimized for RN) or JSC
                 (V8 is NOT used by default)
Web/Node       → V8
```

Hermes is now the recommended engine for Android - it's optimized specifically for React Native with faster startup and lower memory usage.

Architecture Diagram


JavaScript Thread (Hermes/JSC)
- Your React code
- Business logic
- State management
Bridge/JSI
(Serialized Messages)
Native Thread (iOS/Android)
- UIKit (iOS) / Android Views
- Actual rendering
- Native modules (Camera, GPS, etc.)

React vs React Native

React (Web)	React Native (Mobile)
`<div>`	`<View>`
`<span>`, `<p>`	`<Text>`
`<input>`	`<TextInput>`
`<button>`	`<TouchableOpacity>`
`<img>`	`<Image>`

All text must be in `<Text>` components!

Core Components

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

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

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

View Component

```
<View style={styles.container}>
  <Text>Content here</Text>
</View>
```

Flutter equivalent:

```
Container(
  child: Text('Content here'),
)
```

`View` is like Container - the basic building block

Text Component

```
// ✅ Correct
<Text>Hello World</Text>

// ❌ Wrong - text must be in Text component
<View>Hello World</View>
```

Flutter equivalent:

```
Text('Hello World')
```

Styling in React Native

No CSS! Use JavaScript objects

```
const styles = StyleSheet.create({
  container: {
    backgroundColor: 'lightblue',
    padding: 20,
    borderRadius: 10,
    marginTop: 50
  },
  text: {
    fontSize: 18,
    fontWeight: 'bold',
    color: '#333'
  }
});
```

Flutter equivalent: Widget properties or TextStyle

Flexbox Layout

```
<View style={{
  flex: 1,
  flexDirection: 'column',  // or 'row'
  justifyContent: 'center',
  alignItems: 'center'
}}>
  <Text>Centered!</Text>
</View>
```

Flutter equivalent: Column/Row with MainAxisAlignment and CrossAxisAlignment

Inline vs StyleSheet

```
// Inline (okay for quick tests)
<View style={{ padding: 10, backgroundColor: 'red' }}>

// StyleSheet (better performance, reusable)
const styles = StyleSheet.create({
  container: { padding: 10, backgroundColor: 'red' }
});

<View style={styles.container}>
```

Use `StyleSheet.create()` for production code!

Button and Touchables

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

function MyButton() {
  return (
    <TouchableOpacity
      onPress={() => console.log('Pressed')}
      style={styles.button}
    >
      <Text style={styles.buttonText}>Click me</Text>
    </TouchableOpacity>
  );
}
```

Flutter equivalent: TextButton, ElevatedButton, or InkWell

Touchable Components

- `TouchableOpacity` - Fades on press
- `TouchableHighlight` - Darkens on press
- `Pressable` - Modern, customizable (recommended)

```
<Pressable
  onPress={() => {}}
  style={({ pressed }) => [
    styles.button,
    pressed && styles.pressed
  ]}
>
  <Text>Press me</Text>
</Pressable>
```

TextInput Component

```
import { TextInput } from 'react-native';

function LoginForm() {
  const [email, setEmail] = useState('');

  return (
    <TextInput
      value={email}
      onChangeText={setEmail}
      placeholder="Enter email"
      keyboardType="email-address"
      style={styles.input}
    />
  );
}
```

Flutter equivalent: TextField

ScrollView

```
import { ScrollView } from 'react-native';

function LongList() {
  return (
    <ScrollView>
      <Text>Item 1</Text>
      <Text>Item 2</Text>
      {/* Many items... */}
    </ScrollView>
  );
}
```

Flutter equivalent: SingleChildScrollView

For long lists: Use `FlatList` instead!

FlatList: Efficient Lists

```
import { FlatList } from 'react-native';

function UserList({ users }) {
  return (
    <FlatList
      data={users}
      keyExtractor={item => item.id}
      renderItem={({ item }) => (
        <Text>{item.name}</Text>
      )}
    />
  );
}
```

Flutter equivalent: ListView.builder

Only renders visible items!

Image Component

```
import { Image } from 'react-native';

// Local image
<Image
  source={require('./assets/logo.png')}
  style={{ width: 100, height: 100 }}
/>

// Network image
<Image
  source={{ uri: 'https://example.com/image.jpg' }}
  style={{ width: 100, height: 100 }}
/>
```

Flutter equivalent: Image.asset, Image.network

Platform-Specific Code

```
import { Platform } from 'react-native';

const styles = StyleSheet.create({
  container: {
    marginTop: Platform.OS === 'ios' ? 20 : 0,
    ...Platform.select({
      ios: { shadowColor: 'black' },
      android: { elevation: 5 }
    })
  }
});
```

Flutter equivalent: Platform.isIOS, Platform.isAndroid

Navigation: React Navigation

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

const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator>
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}
```

Navigating Between Screens

```
function HomeScreen({ navigation }) {
  return (
    <View>
      <Button
        title="Go to Details"
        onPress={() => navigation.navigate('Details', {
          userId: 123
        })}
      />
    </View>
  );
}

function DetailsScreen({ route }) {
  const { userId } = route.params;
  return <Text>User ID: {userId}</Text>;
}
```

Flutter equivalent: Navigator.push

SafeAreaView

```
import { SafeAreaView } from 'react-native';

function App() {
  return (
    <SafeAreaView style={{ flex: 1 }}>
      <Text>This respects notches and status bars</Text>
    </SafeAreaView>
  );
}
```

Flutter equivalent: SafeArea widget

Important for iOS devices with notches!

StatusBar

```
import { StatusBar } from 'react-native';

function App() {
  return (
    <View>
      <StatusBar barStyle="dark-content" />
      <Text>Content</Text>
    </View>
  );
}
```

Controls the status bar appearance

Example: Counter App

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

export default function CounterApp() {
  const [count, setCount] = useState(0);

  return (
    <View style={styles.container}>
      <Text style={styles.count}>{count}</Text>
      <Button title="+" onPress={() => setCount(count + 1)} />
      <Button title="-" onPress={() => setCount(count - 1)} />
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, justifyContent: 'center', alignItems: 'center' },
  count: { fontSize: 48, marginBottom: 20 }
});
```

Example: Simple Todo App

```
function TodoApp() {
  const [todos, setTodos] = useState([]);
  const [text, setText] = useState('');

  const addTodo = () => {
    if (text.trim()) {
      setTodos([...todos, { id: Date.now(), text }]);
      setText('');
    }
  };

  return (
    <View style={styles.container}>
      <TextInput value={text} onChangeText={setText} />
      <Button title="Add" onPress={addTodo} />
      <FlatList
        data={todos}
        keyExtractor={item => item.id.toString()}
        renderItem={({ item }) => <Text>{item.text}</Text>}
      />
    </View>
  );
}
```

React Native vs Flutter

Feature	React Native	Flutter
Language	JavaScript/TypeScript	Dart
UI Components	Native	Custom rendered
Hot Reload	✅ Fast Refresh	✅ Hot Reload
Styling	StyleSheet objects	Widget properties
Navigation	React Navigation	Navigator
State Management	Hooks, Context, Redux	setState, Provider, Riverpod

Popular Libraries

- **Navigation:** React Navigation
- **State Management:** Redux, MobX, Zustand
- **UI Libraries:** React Native Paper, Native Base
- **Icons:** React Native Vector Icons
- **Forms:** Formik, React Hook Form
- **HTTP:** Axios, Fetch API

Development Tools

- **Expo:** Managed workflow (easier start)
- **React Native CLI:** Full control
- **Metro:** JavaScript bundler
- **Flipper:** Debugging tool
- **React DevTools:** Component inspector

Key Differences from Flutter

1. **JavaScript** instead of Dart
2. **Styling** via objects, not widget properties
3. **All text** must be in `<Text>` tags
4. **Flexbox** by default (similar to Column/Row)
5. **Third-party** navigation library
6. **Hot reload** works similarly

Common Patterns

```
// Conditional rendering
{isLoading ? <Loader /> : <Content />}

// List rendering
{items.map(item => <Item key={item.id} {...item} />)}

// Event handling
<Button onPress={handlePress} />

// Styling composition
<View style={[styles.base, isActive && styles.active]} />
```

Project Structure

```
my-app/
├── App.js                   # Entry point
├── package.json             # Dependencies
├── components/              # Reusable components
├── screens/                # Screen components
├── navigation/             # Navigation setup
├── styles/                 # Shared styles
└── assets/                 # Images, fonts
```

Getting Started

```
# Using Expo (recommended for beginners)
npx create-expo-app MyApp
cd MyApp
npm start

# Using React Native CLI
npx react-native init MyApp
cd MyApp
npm run android  # or npm run ios
```

Key Takeaways

1. React Native uses **native components**
2. Styling is done with **JavaScript objects**
3. **Flexbox** for layouts (like Flutter's Row/Column)
4. **FlatList** for efficient lists
5. **React Navigation** for routing
6. All concepts from React apply here
7. Platform-specific code when needed

Resources

- React Native Docs: https://reactnative.dev
- React Navigation: https://reactnavigation.org
- Expo Docs: https://docs.expo.dev
- Awesome React Native: https://github.com/jondot/awesome-react-native

Practice: Build small apps to solidify concepts!