Mobile App Development

Mobile development: React Native

Case study 1

- **Your boss**: *I need you to make a counter app for me right now.*
- **You**: *Sure, give me .... minutes.*

Expo

The fastest way to make "counter" app materialize on your phone.

Assume

- You know React.
- You and your boss have Android phones.

Setup

- Create Expo account.
- Install required CLI tool / authenticated
  - `npm install -g eas-cli`
  - `eas login`

Steps

- `npx create-expo-app -t expo-template-blank-typescript`
- `cd <project-name>`
- `npx expo install expo-updates` (For OTA update)
- `npm start`
- Scan QR code using `Expo Go`

App.tsx

```tsx
import { StatusBar } from "expo-status-bar";
import { StyleSheet, Text, View, Button } from "react-native";
import { useState } from "react";
export default function App() {
  const [count, setCount] = useState(0);
  return (
    <View style={styles.container}>
      <StatusBar backgroundColor="blue" />
      <Text style={{ fontSize: 50 }}>Coutns: {count}</Text>
      <Button onPress={() => setCount((c) => c + 1)} title="Add" />
      <Button onPress={() => setCount(0)} title="Reset" color="red" />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
    gap: 10,
  },
});
```

Build

- `eas init`
- `eas update:configure`
- `eas build:configure`
- `eas build --platform android --profile preview`

Update

- `eas update --branch preview --message "Fix typo"`

Case study 2

- **Your boss**: *I need you to make a mirror app for me right now.*
- **You**: *Sure, give me .... minutes.*

Initialize a project

- `npx create-expo-app -t expo-template-blank-typescript`
- `npx expo install expo-updates expo-camera`
- `npm install nativewind react-native-reanimated react-native-safe-area-context`
- `npm install --dev tailwindcss@^3.4.17 prettier-plugin-tailwindcss@^0.5.11 babel-preset-expo`
- `npx tailwindcss init`

Configuring TailwindCSS

./tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  // NOTE: Update this to include the paths to all files that contain Nativewind classes.
  content: ["./App.tsx", "./components/**/*.{js,jsx,ts,tsx}"],
  presets: [require("nativewind/preset")],
  theme: {
    extend: {},
  },
  plugins: [],
};
```

Global styles

`./global.css`

```
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Configuring TailwindCSS

./babel.config.js

```
module.exports = function (api) {
  api.cache(true);
  return {
    presets: [
      ["babel-preset-expo", { jsxImportSource: "nativewind" }],
      "nativewind/babel",
    ],
  };
};
```

Metro Config

`./metro.config.js`

```
const { getDefaultConfig } = require("expo/metro-config");
const { withNativeWind } = require("nativewind/metro");

const config = getDefaultConfig(__dirname);

module.exports = withNativeWind(config, { input: "./global.css" });
```

Metro is a JavaScript bundler that is part of React Native. It compiles and bundles JavaScript code for mobile apps. Source

Metro Config

To enable Metro to process CSS files, add the following configuration to your `app.json`:

```
{
  "expo": {
    "web": {
      "bundler": "metro"
    }
  }
}
```

Add types

`./nativewind-env.d.ts`

```
/// <reference types="nativewind/types" />
```

Permission

./app.json

```
{
  "expo": {
    ...
    "plugins": [
      [
        "expo-camera",
        {
          "cameraPermission": "Allow $(PRODUCT_NAME) to access your camera."
        }
      ]
    ]
    ...
  }
}
```

Main application

`./App.tsx`

https://github.com/mobileapp-68/rn-camera-68/blob/main/App.tsx

Build and deploy

- `eas init`
- `eas update:configure`
- `eas build:configure`
- `eas build --platform android --profile preview`

Overview of React Native

React Native

- React Native is an open-source UI software framework created by Meta (formerly Facebook).
- RN enables developers to build native mobile applications for iOS and Android using JavaScript and the `React` library.

React Native vs Flutter

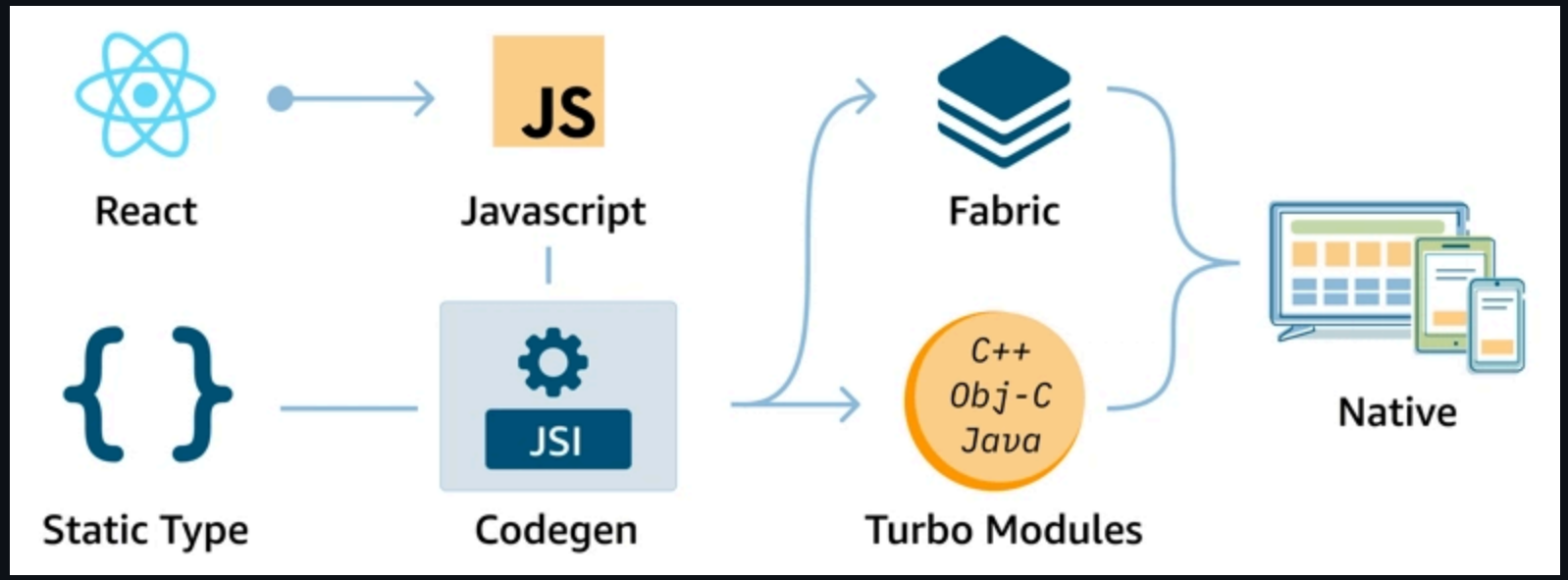
- Popularity

	React Native	**Flutter**
Language	JavaScript / TypeScript	Dart
UI	Wraps **native** iOS/Android components (plus some custom)	Custom **widget** tree rendered via Skia/Impeller engine
Dev API	Slim core + large 3rd-party ecosystem (Meta + community)	Rich core SDK, many things “batteries-included”, smaller but growing ecosystem

	React Native	**Flutter**
Dev option	More versatile for JS shops, easy web integration, many libraries and devs available	More streamlined, consistent UI across platforms, great tooling and hot reload
Performance	Improved a lot with new Fabric/TurboModules, but still usually behind Flutter in raw FPS/startup under heavy UI	Typically faster and more consistent (startup, animations, heavy graphics) thanks to AOT + Skia/Impeller

React Native

React Native architecture



Source

React Native New Architecture

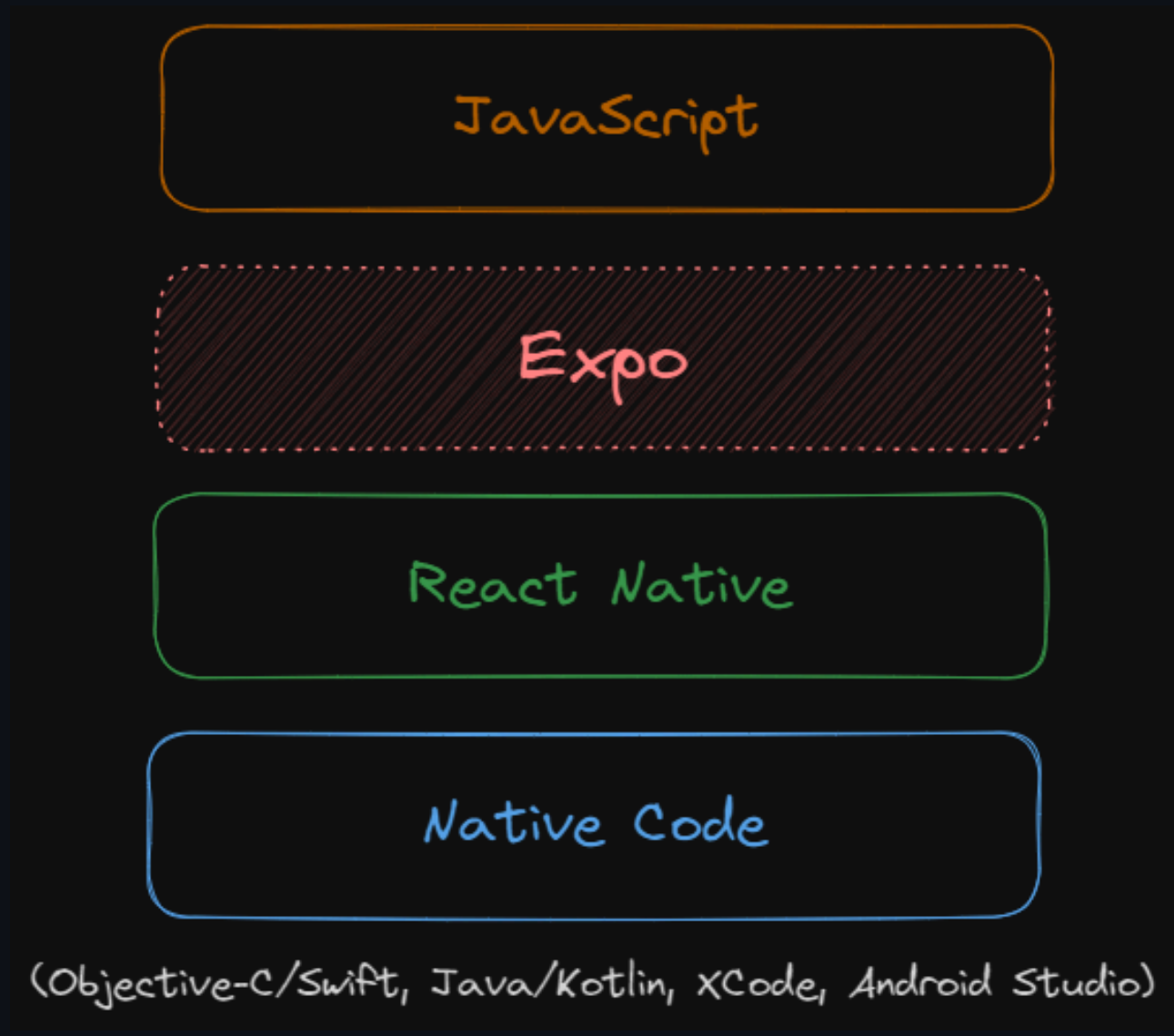
- Uses React on top, Hermes JS engine, and a "New Architecture" core.
- **JSI**: Direct JS <-> C++ interface, no JSON bridge.
- **Fabric**: New C++ renderer + Yoga v3, concurrent and sometimes synchronous rendering on UI thread.
- **TurboModules**: Lazy-loaded, JSI-based native modules with batched, low-overhead calls.
- **Codegen**: Typed contracts + generated glue across JS/native.

But what is Expo?

- Expo is a set of tools and services built around React Native.
- From React Native official doc:

> If you are new to mobile development, the easiest way to get started is with Expo Go.

Expo position



Expo ecosystem

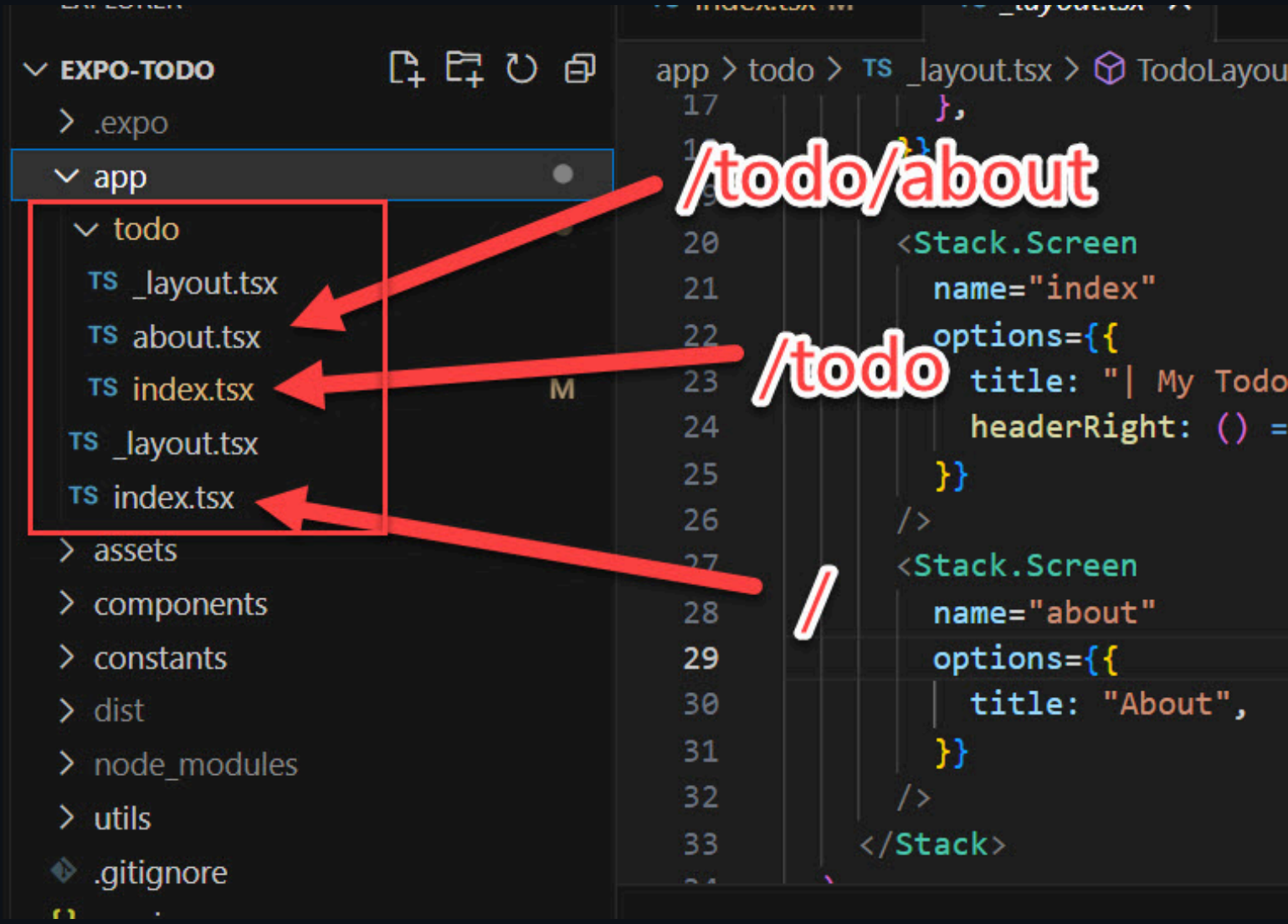
- **Expo SDK**
  - Framework for building React Native apps.
- **Expo Go**
  - App that makes testing apps easy via a scannable QR code.
- **Expo Dev Clients**
  - A framework to extend Expo Go.
- **Expo Application Services (EAS)**
  - Freemium services for building and submission.

Todo app

```
git clone https://github.com/mobileapp-68/rn-todo-68
```

Expo router

- File-base routing.
  - `/app` folder
- `_layout.tsx` for layout



/app/todo/_layout.tsx

```tsx
const AboutMenu = () => {
  return (
    // 👇👇👇👇👇👇👇👇👇
    <Link href="/todo/about">
      <TouchableOpacity onPress={() => {}} style={{ paddingRight: 10 }}>
        <Ionicons
          name="help-circle-outline"
          size={32}
          color={COLORS.lightWhite}
        />
      </TouchableOpacity>
    </Link>
  );
};
```

Main navigation (tab)

`/app/_layout.tsx`

```
export default function AppLayout() {
  return (
    <View ...>
      <Tabs screenOptions={...}>
        <Tabs.Screen name="index" options={...}/>
        <Tabs.Screen name="todo" options={...}/>
      </Tabs>
    </View>
  );
}
```

Secondary navigation (stack)

`/app/todo/_layout.tsx`

```tsx
export default function TodoLayout() {
  return (
    <Stack screenOptions={...} >
      <Stack.Screen name="index" options={...} />
      <Stack.Screen name="about" options={} />
    </Stack>
  );
}
```

Styling

- Cannot use CSS.
- All of the core components accept a prop named `style`.
- The style names and values usually match how CSS works on the web.
- Default behavior is `flex-column`

Styling

./app/index.tsx

```tsx
import { StyleSheet } from "react-native";
// ...
export default function Home() {
  return <View style={styles.container}>...</View>;
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
    fontFamily: "Prompt",
  },
});
```

Style library

- Native Base
- Native Wind

I just want to press something...

- `Button`
- `TouchableOpacity`
- `TouchableHighlight`
- `TouchableWithoutFeedback`
- `TouchableNativeFeedback`
- `Pressable`

TouchableOpacity

./components/TodoForm.tsx

```
import { TouchableOpacity } from "react-native";

const TodoForm: FC<Props> = ({ txt, setTxt, addTodo }) => {
  return (
    // ...
      <TouchableOpacity style={...} onPress={...}>
        <Ionicons ... />
      </TouchableOpacity>
  );
};

export default TodoForm;
```

I just want to see a list.

- `ScrollView`
- `FlatList`
- `SectionList`
- `VirtualizedList`
- `VirtualizedSectionList`

./component/TodoList.tsx

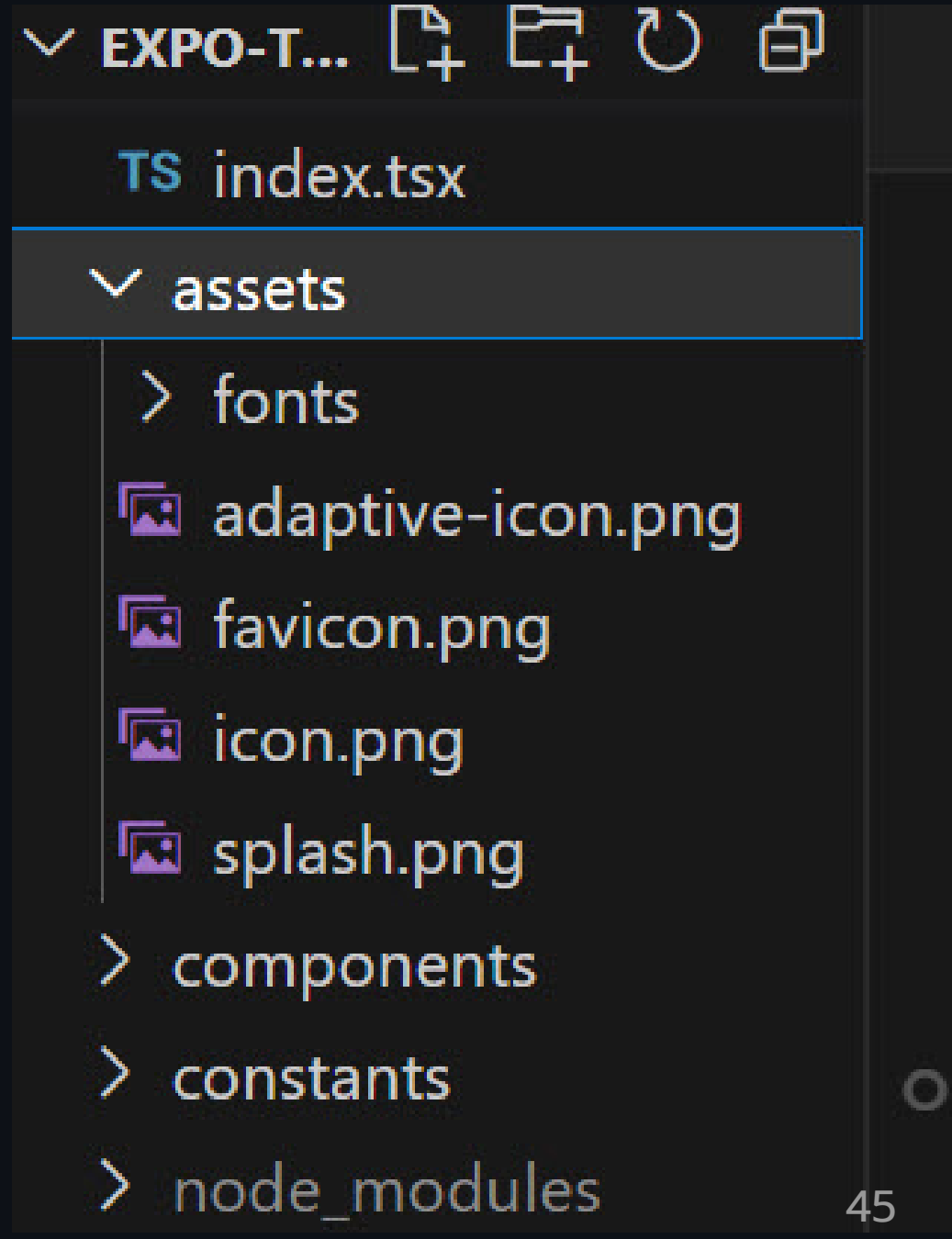
```tsx
import {  ListRenderItemInfo, FlatList }

const TodoList: FC<Props> = (props) => {
  const renderTodo = ({ item }: ListRenderItemInfo<Todo>) => (
    <TodoItem todo={item} deleteTodo={props.deleteTodo} />
  );

  return (
    <View style={styles.container}>
      <FlatList
        data={props.todos}
        renderItem={renderTodo}
        keyExtractor={(todo: Todo) => todo.id.toString()}
        ItemSeparatorComponent={Separator}
      />
    </View>
  );
};
```

Icon and splash screen

- Template



Business logic

Plain old React

./app/todo/index.tsx

```tsx
import { useState, useEffect } from "react";
import axios from "axios";
export default function Todo() {
  const [todos, setTodos] = useState<Todo[]>([]);

  function deleteTodo(id: number) {...}
  function addTodo(txt: string) {...}

  useEffect(() => {
    axios
      .get("https://jsonplaceholder.typicode.com/todos")
      .then((res) => {
        setTodos(res.data.slice(0, 10));
      })
  }, []);

  return (...);
}
```

./components/TodoForm.tsx

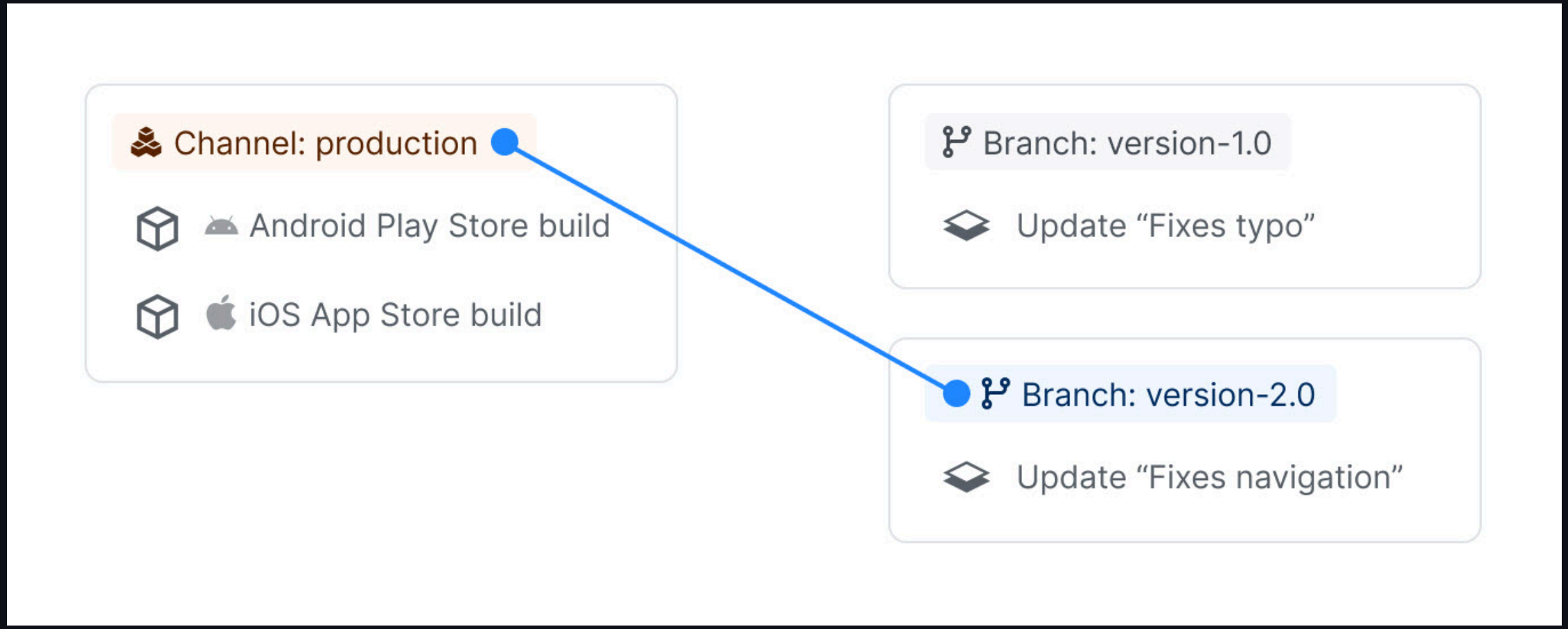
```tsx
interface Props {
  txt: string;
  setTxt: (txt: string) => void;
  addTodo: (txt: string) => void;
}

const TodoForm: FC<Props> = ({ txt, setTxt, addTodo }) => {
  return (
    //...
    <TextInput onChangeText={(t) => setTxt(t)} value={txt} />
    <TouchableOpacity onPress={() => {addTodo(txt);}}>
    //...
    </TouchableOpacity>
    //...
  );
};
```

Deployment

- `Profile`
- `Channel`
- `Branch`

Channel and branch



Build

./eas.json

```
{
  "build": {
    "development": {
      "developmentClient": true,
      "distribution": "internal",
      "channel": "development"
    },
    "preview": {
      "channel": "preview",
      "distribution": "internal"
    },
    "production": {
      "channel": "production"
    }
  }
}
```

Build

- `eas build --platform android --profile preview`

Inspect

- `eas channel:list`
- `eas branch:list`

Update

- `eas update --branch preview --message "Fix typo"`

🦀 Takeaway

Choose React Native if

- You and your team know React.
- Your business **differentiator** is not mobile applications.