Components, Props, and State (React Native)

Lesson 2

Installing

```
npx create-expo-app@latest YourAppName
```

```
cd YourAppName
```

```
npx expo start
```

Or

```
Npm run web
```

```
Npm run android/ios
```

```
npx expo start -c
```

Lesson Objectives

- Understand what components are
- Pass data using props
- Manage dynamic data using state
- Build a simple interactive screen

What is a Component?

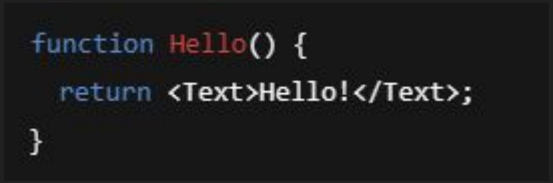
A component is a reusable piece of UI.

- Think of components as functions that return UI

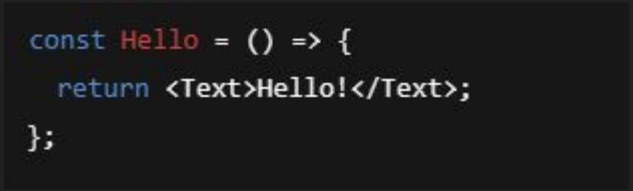
Examples:

- Button
- Header
- Card
- Screen

Functional Components (Modern, Recommended)



```
function Hello() {
  return <Text>Hello!</Text>;
}
```



```
const Hello = () => {
  return <Text>Hello!</Text>;
};
```

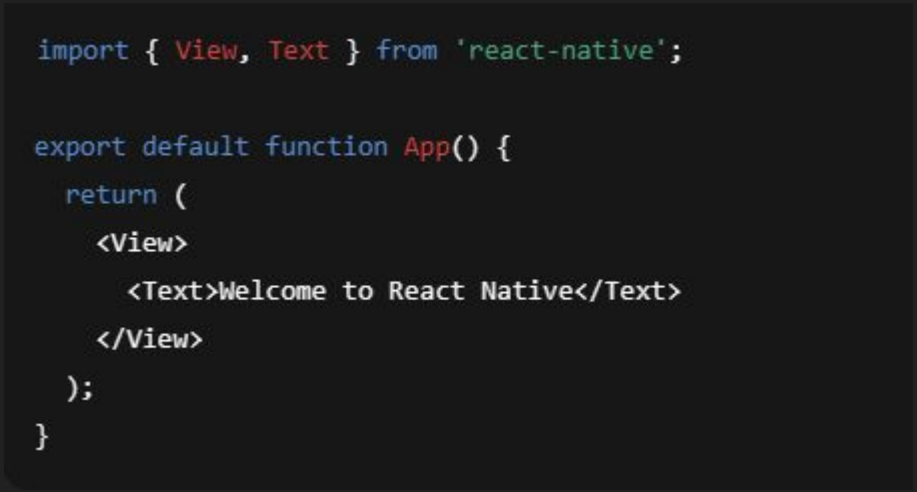
Component Rules (Important!)

- Component name must start with a capital letter

- Must return one root element

- Can be reused multiple times

Example: Simple Component



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

export default function App() {
  return (
    <View>
      <Text>Welcome to React Native</Text>
    </View>
  );
}
```

Why Reusable Components?

- Less duplicated code
- Cleaner structure
- Easier to maintain

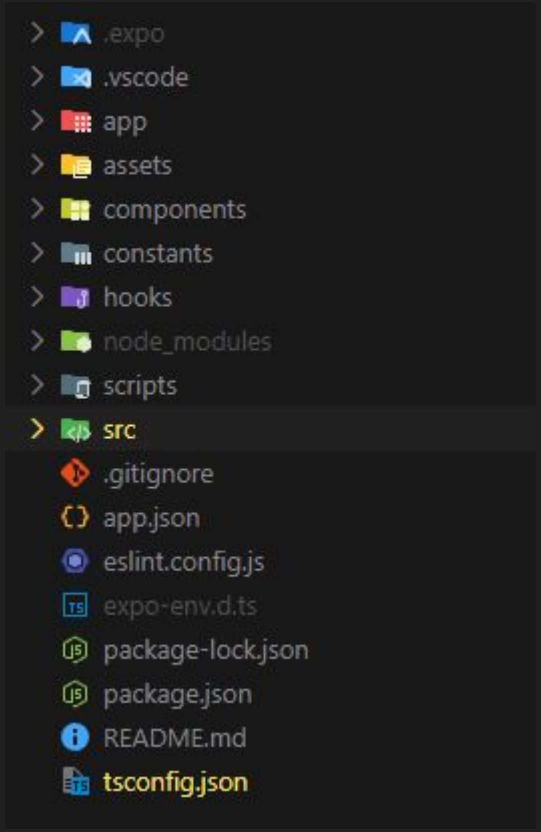
Recommended Folder Structure (Real Projects)

```
src/
    ├── components/
    ├── screens/
    ├── navigation/
    ├── services/
    ├── hooks/
    ├── utils/                              (md folder1)
```

Question

The required folder are already created

Whats the point of moving them to “SRC” folder?

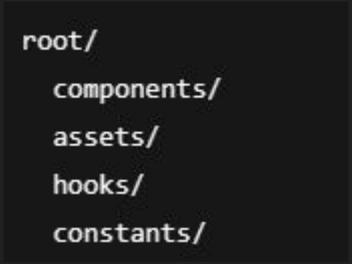


```
> .expo
> .vscode
> app
> assets
> components
> constants
> hooks
> node_modules
> scripts
> src
  .gitignore
  app.json
  eslint.config.js
  expo-env.d.ts
  package-lock.json
  package.json
  README.md
  tsconfig.json
```

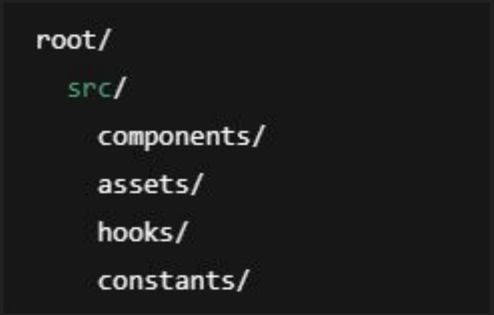
The real reason src/ exists

`src/` is meant to say:

> “Everything that is application code lives here.”



```
root/
  components/
  assets/
  hooks/
  constants/
```



```
root/
  src/
    components/
    assets/
    hooks/
    constants/
```

Why keeping app code in root becomes messy later

Now app code is mixed with:

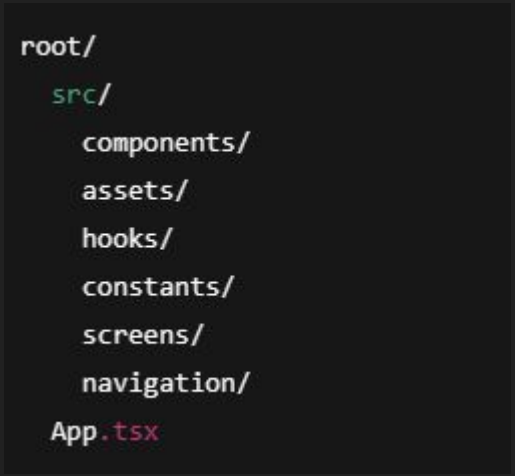
build configs

CI scripts

platform files

That’s where src/ saves your sanity.

Move everything into src/ (recommended)

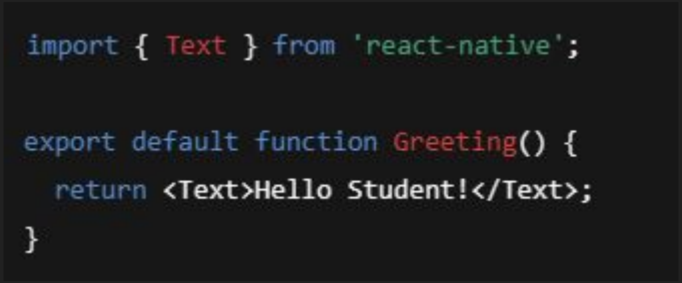


```
root/
  src/
    components/
    assets/
    hooks/
    constants/
    screens/
    navigation/
  App.tsx
```

How to Move it?

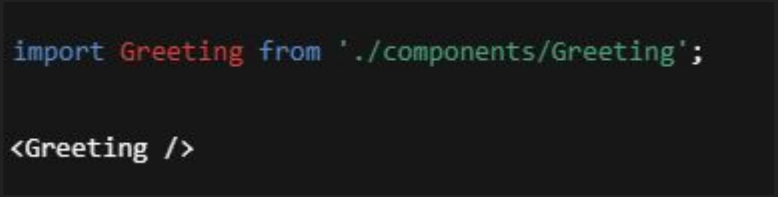
Custom Component

components/Greeting.js



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

export default function Greeting() {
  return <Text>Hello Student!</Text>;
}
```



```
import Greeting from './components/Greeting';

<Greeting />
```

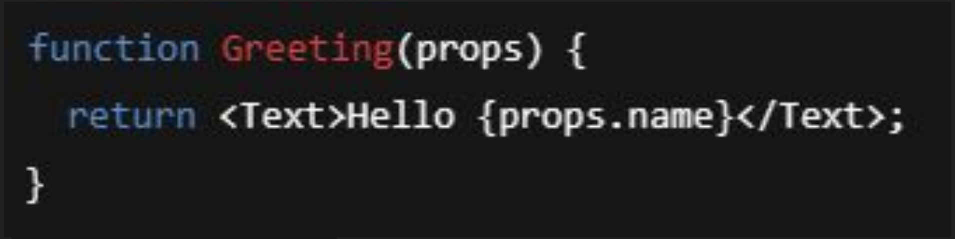
Props (Passing Data to Components)

Props = properties

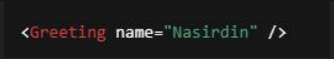
They allow you to pass data from parent to child components

Props are read-only

Props Example



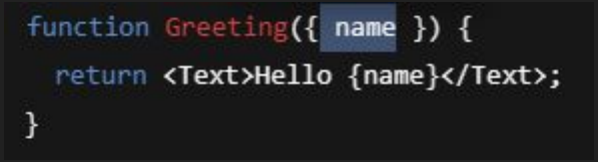
```
function Greeting(props) {
  return <Text>Hello {props.name}</Text>;
}
```



```
<Greeting name="Nasirdin" />
```

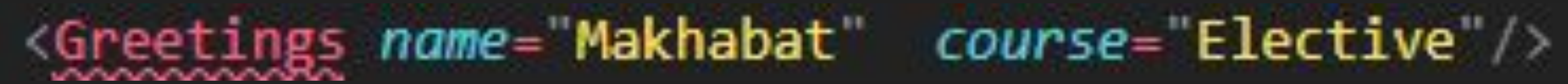
Props with Destructuring (Recommended)

Props destructuring in React is an ES6 technique that extracts specific properties directly from the props object within component function parameters, reducing props.

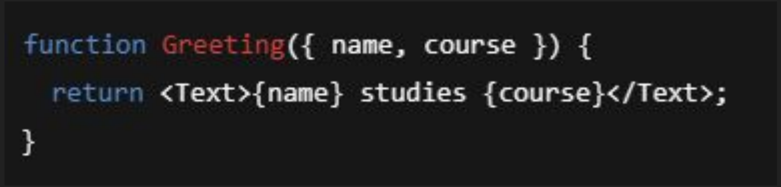


```
function Greeting({ name }) {
  return <Text>Hello {name}</Text>;
}
```

Passing Multiple Props



```
<Greetings name="Makhabat"  course="Elective"/>
```



```
function Greeting({ name, course }) {
  return <Text>{name} studies {course}</Text>;
}
```

State (Dynamic Data)

State is data that:

- Changes over time
- Affects what is shown on the screen

Examples:

- Counter value
- Input text
- Logged-in user