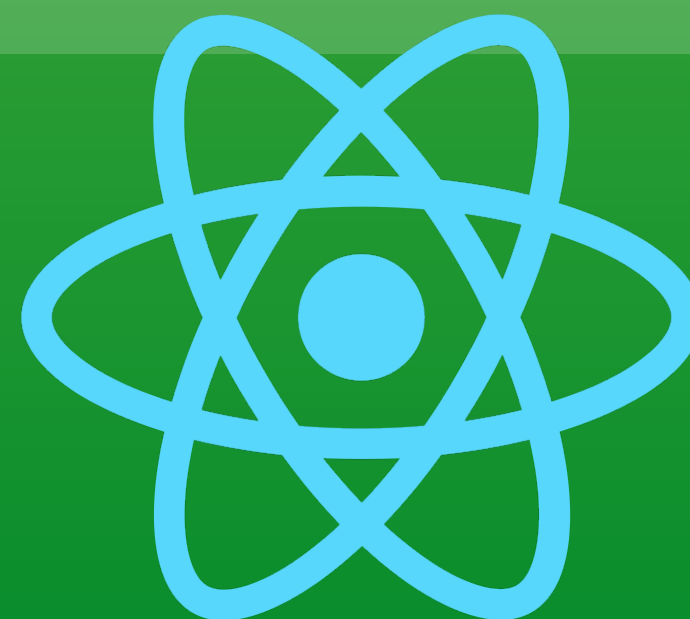




JESUÏTES
educació

DAWM14UF2
PROGRAMACIÓ D'APLICACIONS MÒBILS

UF2.6.2 REACT NATIVE COMPONENTS BÀSICS



CFGS Desenvolupament d'Aplicacions Web

M14. Desenvolupament d'aplicacions en entorns mòbils

Fundació Jesuïtes Educació - Escola del Clot

Sergi Grau sergi.grau@fje.edu

- + Treballar amb els components bàsics de React native
- + Creació de components bàsics.
- + Creació de components amb estat

- + La majoria d'aplicacions utilitzen un d'aquests components bàsics. Es tradueixen a components nadius de cada plataforma

View

The most fundamental component for building a UI.

Text

A component for displaying text.

Image

A component for displaying images.

TextInput

A component for inputting text into the app via a keyboard.

ScrollView

Provides a scrolling container that can host multiple components and views.

StyleSheet

Provides an abstraction layer similar to CSS stylesheets.

- + El component View és el component més fonamental per crear una UI, View és un contenidor que admet la disposició amb flexbox, estil, maneig tàctil i controls d'accessibilitat.
- + View es tradueix directament a l'equivalent a la vista nativa de qualsevol plataforma on funcioni React Native, tant si es tracta d'un UIView, <div>, android.view, etc.
- + La vista està dissenyada per niar dins d'altres vistes i pot tenir entre 0 i molts fills de qualsevol tipus.

```
return (  
  <View style={styles.container}>  
    <Text>DAW2</Text>  
  </View>  
);
```

```
<View  
  style={{  
    flexDirection: 'row',  
    height: 100,  
    padding: 20,  
  }}>  
  <View style={{backgroundColor: 'blue', flex: 0.3}} />  
  <View style={{backgroundColor: 'red', flex: 0.5}} />  
  <Text>Hello World!</Text>  
</View>
```

- + Component de reacció per mostrar text. El text admet la nidificació, el disseny i el maneig tàctil.
- + Tant Android com iOS permeten mostrar text formatat anotant intervals d'una cadena amb format específic com el text en negreta o color (NSAttributedString a iOS, SpannableString a Android). A la pràctica, això és molt tediós. Per a React Native, vam decidir utilitzar un paradigma web on podeu niar text per aconseguir el mateix efecte.

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

export default class BoldAndBeautiful extends Component {
  render() {
    return (
      <Text style={{fontWeight: 'bold'}}>
        I am bold
      <Text style={{color: 'red'}}>
        and red
      </Text>
    </Text>
  );
}
```

- + L'element `<Text>` és únic en relació amb la disposició: tot el que hi ha a dins ja no s'utilitza la disposició de flexbox, sinó que s'utilitza la disposició de text.
- + Això vol dir que els elements que hi ha dins d'un `<Text>` ja no són rectangles, sinó que s'emboliquen quan veuen el final de la línia.
- + A React Native, som més estrictes al web al fet de posar text directament en un contenidor: heu d'embolicar tots els nodes de text dins d'un component `<Text>`. No podeu tenir un node de text directament sota una `<View>`.

Un component senzill

Els components de React implementen el mètode `render()` que rep dades d'entrada i retorna allò a mostrar. En aquest exemple s'usa una sintaxi similar a XML anomenada JSX. El mètode `render()` pot accedir a les dades d'entrada del component mitjançant `this.props`.

JSX no és obligatori per utilitzar React.

Proveu [Babel REPL](#) per veure el codi JavaScript produït per la fase de compilació del format JSX.

EDITOR JSX INTERACTIU

☒ JSX?

RESULTAT

```
class HelloMessage extends React.Component {  
  render() {  
    return (  
      <div>  
        Hola {this.props.name}  
      </div>  
    );  
  }  
}  
  
ReactDOM.render(  
  <HelloMessage name="Júlia" />,  
  document.getElementById('hello-example')  
);
```

Hola Júlia

Un component senzill

Els components de React implementen el mètode `render()` que rep dades d'entrada i retorna allò a mostrar. En aquest exemple s'usa una sintaxi similar a XML anomenada JSX. El mètode `render()` pot accedir a les dades d'entrada del component mitjançant `this.props`.

JSX no és obligatori per utilitzar React.

Proveu [Babel REPL](#) per veure el codi JavaScript produït per la fase de compilació del format JSX.

EDITOR JSX INTERACTIU

☒ JSX?

RESULTAT

```
class HelloMessage extends React.Component {  
  render() {  
    return (  
      <div>  
        Hola {this.props.name}  
      </div>  
    );  
  }  
}  
  
ReactDOM.render(  
  <HelloMessage name="Júlia" />,  
  document.getElementById('hello-example')  
);
```

Hola Júlia

Un component amb estat

A més a més de rebre dades d'entrada (accessibles mitjançant `this.props`), un component pot mantenir dades del seu estat local (accessibles mitjançant `this.state`). Quan les dades d'estat d'un component canvien, es torna a cridar a `render()` i s'actualitza el marcat renderitzat.

EDITOR JSX INTERACTIU

☒ JSX?

RESULTAT

```
class Timer extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = { seconds: 0 };  
  }  
  
  tick() {  
    this.setState(state => ({  
      seconds: state.seconds + 1  
    }));  
  }  
  
  componentDidMount() {  
    this.interval = setInterval(() => this.tick(), 1000);  
  }  
}
```

Segons: 77

```
class Timer extends React.Component {
  constructor(props) {
    super(props);
    this.state = { seconds: 0 };
  }

  tick() {
    this.setState(state => ({
      seconds: state.seconds + 1
    }));
  }

  componentDidMount() {
    this.interval = setInterval(() => this.tick(), 1000);
  }

  componentWillUnmount() {
    clearInterval(this.interval);
  }

  render() {
    return (
      <div>
        Segons: {this.state.seconds}
      </div>
    );
  }
}

ReactDOM.render(
  <Timer />,
  document.getElementById('timer-example')
);
```

Una aplicació

Emprant `props` i `state`, podeu crear una petita aplicació per gestionar tasques pendents. Aquest exemple utilitza `state` per mantenir el llistat de tasques i el text que l'usuari ha introduït. Tot i que els gestors d'esdeveniments semblen renderitzats en línia, s'apleguen i s'implementen utilitzant els principis de delegació d'esdeveniments.

EDITOR JSX INTERACTIU

✓ JSX?

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { items: [], text: '' };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  render() {
    return (
      <div>
        <h3>Per fer</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <label htmlFor="new-todo">
            Què cal fer?
          </label>
        </form>
      </div>
    );
  }
}
```

RESULTAT

Per fer

Què cal fer?

Afegeix #1

```
class MarkdownEditor extends React.Component {
  constructor(props) {
    super(props);
    this.handleChange = this.handleChange.bind(this);
    this.state = { value: 'Hola **món**!' };
  }

  handleChange(e) {
    this.setState({ value: e.target.value });
  }

  getRawMarkup() {
    const md = new Remarkable();
    return { __html: md.render(this.state.value) };
  }

  render() {
    return (
      <div className="MarkdownEditor">
        <h3>Entrada</h3>
        <label htmlFor="markdown-content">
          Introduïu text en format markdown
        </label>
        <textarea
          id="markdown-content"
          onChange={this.handleChange}
          defaultValue={this.state.value}
        />
        <h3>Sortida</h3>
        <div
          className="content"
          dangerouslySetInnerHTML={this.getRawMarkup()}
        />
      </div>
    );
  }
}
```



```
    );  
  }  
}  
  
ReactDOM.render(  
  <MarkdownEditor />,  
  document.getElementById('markdown-example')  
);
```

UN COMPONENT UTILIZANT CONNECTORS EXTERNS



JESUÏTES
educació

DAWM14UF2 sergi.grau@fje.edu

Un component utilitzant connectors externs

React us permet interactuar amb altres biblioteques i frameworks. Aquest exemple utilitza **remarkable**, una biblioteca de Markdown externa, per convertir en temps real el valor de l'etiqueta `<textarea>`.

EDITOR JSX INTERACTIU

✓ JSX?

```
class MarkdownEditor extends React.Component {  
  constructor(props) {  
    super(props);  
    this.handleChange = this.handleChange.bind(this);  
    this.state = { value: 'Hola **món**!' };  
  }  
  
  handleChange(e) {  
    this.setState({ value: e.target.value });  
  }  
  
  getRawMarkup() {  
    const md = new Remarkable();  
    return { __html: md.render(this.state.value) };  
  }  
}
```

RESULTAT

Entrada

Introduïu text en format markdown

Hola **món**!

Sortida

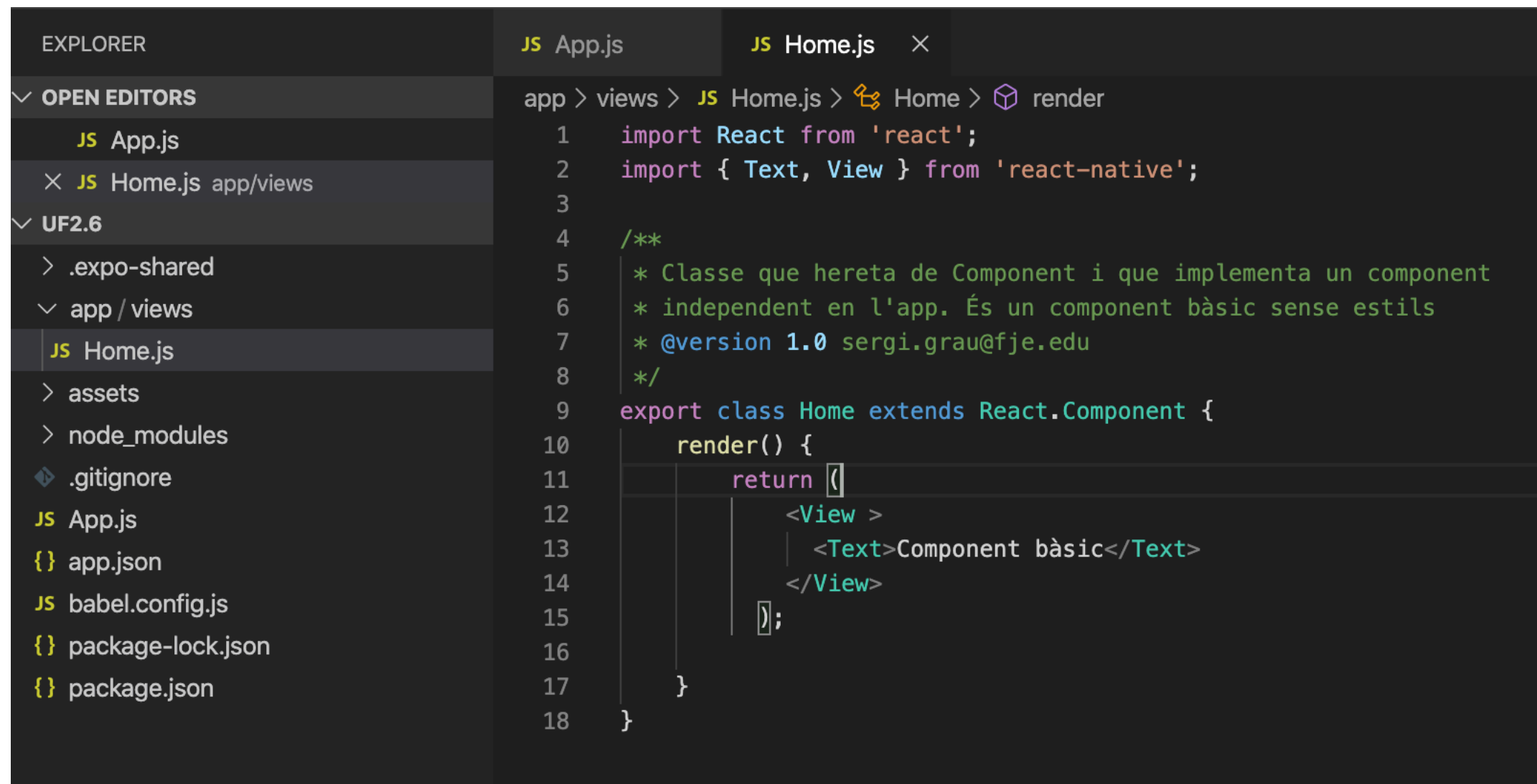
Hola món!

```
class TodoApp extends React.Component {
  constructor(props) {
    super(props);
    this.state = { items: [], text: '' };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }

  render() {
    return (
      <div>
        <h3>Per fer</h3>
        <TodoList items={this.state.items} />
        <form onSubmit={this.handleSubmit}>
          <label htmlFor="new-todo">
            Què cal fer?
          </label>
          <input
            id="new-todo"
            onChange={this.handleChange}
            value={this.state.text}
          />
          <button>
            Afegeix #{this.state.items.length + 1}
          </button>
        </form>
      </div>
    );
  }

  handleChange(e) {
    this.setState({ text: e.target.value });
  }
}
```

- + creeu una carpeta app/view, i un fitxer Home.JS que contindrà el nostre component. Aquest component no tindrà estils i per tant no farà una càrrega del mòdul



The screenshot shows the VS Code interface with the Explorer on the left and the Editor on the right. The Explorer shows the project structure with the 'app / views' directory selected. The Editor shows the 'Home.js' file being created, with the following code:

```
app > views > JS Home.js > Home > render
1  import React from 'react';
2  import { Text, View } from 'react-native';
3
4  /**
5   * Classe que hereta de Component i que implementa un component
6   * independent en l'app. És un component bàsic sense estils
7   * @version 1.0 sergi.grau@fje.edu
8   */
9  export class Home extends React.Component {
10    render() {
11      return (
12        <View >
13          <Text>Component bàsic</Text>
14        </View>
15      );
16    }
17  }
18 }
```


CREACIÓ D'UN COMPONENT PROPI



JESUÏTES
educació

DAWM14UF2 sergi.grau@fje.edu

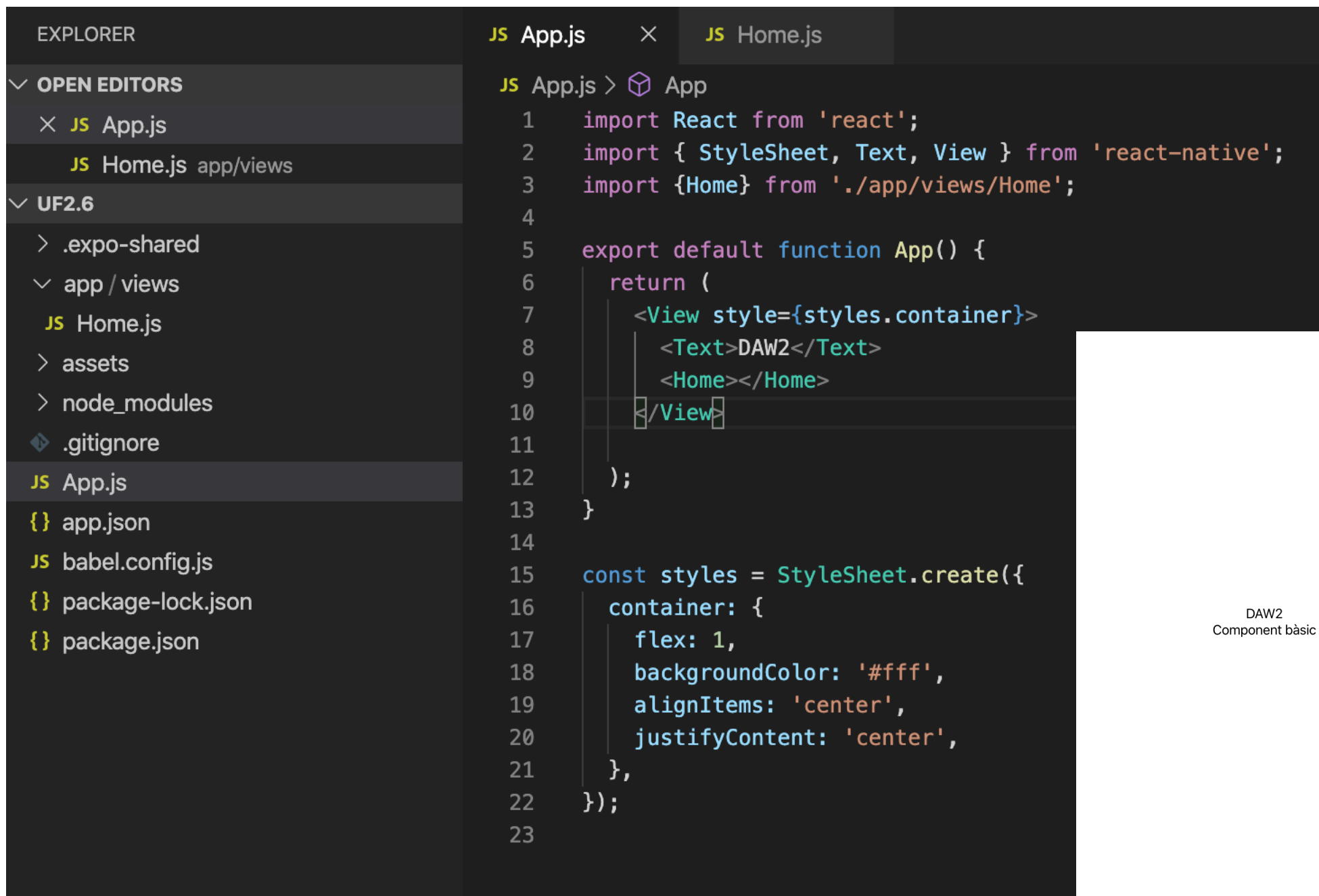
- + Ja el podeu aplicar al vostre JSX de l'aplicació al fitxer App.js

```
EXPLORER
├── OPEN EDITORS
│   ├── JS App.js
│   └── JS Home.js app/views
├── UF2.6
│   ├── .expo-shared
│   ├── app / views
│   │   ├── JS Home.js
│   │   ├── assets
│   │   ├── node_modules
│   │   ├── .gitignore
│   │   ├── JS App.js
│   │   ├── {} app.json
│   │   ├── JS babel.config.js
│   │   ├── {} package-lock.json
│   │   └── {} package.json
└──
```

```
JS App.js
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3  import {Home} from './app/views/Home';
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>DAW2</Text>
9        <Home></Home>
10     </View>
11   );
12 }
13
14
15 const styles = StyleSheet.create({
16   container: {
17     flex: 1,
18     backgroundColor: '#fff',
19     alignItems: 'center',
20     justifyContent: 'center',
21   },
22 });
23
```

DAW2
Component bàsic

- + Ja el podeu aplicar al vostre JSX de l'aplicació al fitxer App.js



```
EXPLORER
├── OPEN EDITORS
│   ├── JS App.js
│   └── JS Home.js app/views
├── UF2.6
│   ├── .expo-shared
│   ├── app / views
│   │   ├── JS Home.js
│   │   ├── assets
│   │   ├── node_modules
│   │   ├── .gitignore
│   │   ├── JS App.js
│   │   ├── {} app.json
│   │   ├── JS babel.config.js
│   │   ├── {} package-lock.json
│   │   └── {} package.json
└──
```

```
JS App.js
1  import React from 'react';
2  import { StyleSheet, Text, View } from 'react-native';
3  import { Home } from './app/views/Home';
4
5  export default function App() {
6    return (
7      <View style={styles.container}>
8        <Text>DAW2</Text>
9        <Home></Home>
10     </View>
11   );
12 }
13
14
15 const styles = StyleSheet.create({
16   container: {
17     flex: 1,
18     backgroundColor: '#fff',
19     alignItems: 'center',
20     justifyContent: 'center',
21   },
22 });
23
```

DAW2
Component bàsic

- + Aquests són els components propis d'UI en Apps

Button

A basic button component for handling touches that should render nicely on any platform.

Picker

Renders the native picker component on Android and iOS.

Slider

A component used to select a single value from a range of values.

Switch

Renders a boolean input.

+ <https://reactnative.dev/>

+ <https://ca.reactjs.org>