UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Programador Web Avanzado

Clase N° 13: React JS - Componentes

Profesor: Ing. Leandro Rodolfo Gil Carrano
Email: leangilutn@gmail.com

Componentes

Componentes

En React JS existen dos categorías recomendadas para los componentes: los componentes de presentación y los componentes contenedores.

Los componentes de presentación son aquellos que simplemente se limitan a mostrar datos y tienen poca o nula lógica asociada a manipulación del estado, es preferible que la mayoría de los componentes de una aplicación sean de este tipo porque son más fáciles de entender y analizar.

Los componentes contenedores tienen como propósito encapsular a otros componentes y proporcionarles las propiedades que necesitan, además se encargan de modificar el estado de la aplicación por ejemplo usando Flux o Redux para despachar alguna acción y que el usuario vea el cambio en los datos.

Componentes - Ventajas

- Favorece la separación de responsabilidades, cada componente debe tener una única tarea.
- Al tener la lógica de estado y los elementos visuales por separado es más fácil reutilizar los componentes.
- Se simplifica la tarea de hacer pruebas unitarias.
- Puede mejorar el rendimiento de la aplicación.
- La aplicación es más fácil de entender.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes De Presentación

Componentes de presentación

- Orientados al aspecto visual
- No tienen dependencia con fuentes de datos (e.g. Flux)
- Reciben callbacks por medio de props
- Pueden ser descritos como componentes funcionales.
- Normalmente no tienen estado

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes de presentación

Ejemplo:

```
// Componentes de presentación
class Item extends React.Component {
  render () {
      return (
          <li><a href='#'>{ this.props.valor }</a></li>
      );
  }
}

class Input extends React.Component {
  render () {
      return (
          <input type='text' placeholder={ this.props.placeholder } />
      );
  }
}

class Titulo extends React.Component {
  render () {
      return (
          <h1>{ this.props.nombre }</h1>
      );
  }
}
```

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes Contenedores

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes contenedores

```
// Contenedor
class AppContainer extends React.Component {
  constructor (props) {
    super(props);
    this.state = {
      temas: ['JavaScript', 'React JS', 'Componentes']
    };
  }

  render () {
    const items = this.state.temas.map(t => (
      <Item valor={ t } />
    ));
    return (
      <div>
       <Titulo nombre='List Items'/>
       <ul>{ items }</ul>
       <Titulo nombre='Inputs'/>
       <div>
        <Input placeholder='Nombre' /><br/>
        <Input placeholder='Apellido' />
       </div>
      </div>
    );
  }
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes Funcionales

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes funcionales

Son componentes de react declarados como funciones, por defecto el componente app viene declarado como un componente funcional

```
import React from 'react';
import logo from './logo.svg';
import './App.css';
import Home from './Home'
import Contacto from './Contacto'
import Menu from './Menu'
function App() {

  return (
    <div className="App">
      Hola Mundo
    </div>
  );
}

export default App;
```

Componentes De clases

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Componentes de clases

Son componentes conformados por clases.
Estos componentes deben heredar de **Component**, ejemplo:

```
import React, { Component } from 'react';


class Contacto extends Component {
    constructor(props){
        super(props)
        console.log(this.props)
    }
  render() {
    let prueba ="dsadas"
    return (
      <div className="App">
        {this.props.data.map(d=><div>{d}</div>)}
        Contacto
      </div>
    );

  }
}

export default Contacto;
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Render

Render

Todo componente de React (tipo clase), tiene un método Render que es el que se encarga de renderizar en el navegador el HTML correspondiente al componente.

Este método se llama automáticamente cuando se crea un componente y cuando el estado del componente se actualiza.

En este método es donde usamos JSX para facilitar el desarrollo y creación de elementos HTML. Veamos un ejemplo:

Render

```
import React from 'react'

class MyComponent extends React.Component {
  constructor () {
    super()
  }

  render () {
    return (
      <div>
        <span>Hola!, soy un componente</span>
      </div>
    )
  }
}
```

Propiedades

Propiedades

Un componente en React puede recibir propiedades como parámetros desde un componente padre para poder insertar valores y eventos en su HTML.

Imagina que tienes un componente que representa un menú con varias opciones, y éstas opciones las pasamos por parámetros como una propiedad llamada options:

Propiedades

```
import React from 'react'

class App extends React.Component {
  constructor () {
    super()
  }

  render () {
    let menuOptions = ['Opción 1', 'Opción 2', 'Opción 3']
    return <Menu options={menuOptions} />
  }
}
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Propiedades

¿Cómo accedemos a estas propiedades en el componente hijo a la hora de renderizarlo? Por medio de las props. Veamos como con el código del componente <Menu />:

```
import React from 'react'

class Menu extends React.Component {
  constructor (props) {
    super(props)
  }

  render () {
    let options = this.props.options
    return (
      <ul>
       {options.map(option => <li>{option}</li>)}
      </ul>
    )
  }
}
```

Propiedades

El código final de este ejemplo sería:

```
import React from 'react'
import ReactDOM from 'react-dom'

class App extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {
    let menuOptions = ['Opción 1', 'Opción 2', 'Opción 3']
    return <Menu options={menuOptions} />
  }
}

class Menu extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {
    let options = this.props.options
    return (
      <ul>
        {options.map(option => <li>{option}</li>)}
      </ul>
    )
  }
}

ReactDOM.render(<App />, document.getElementById('app')
```

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Propiedades

En el caso de que el componente este declarado como función, las propiedades se reciben como argumento de la función, ejemplo:

```
import React from 'react';

function Home(props) {

  return (
    <div >
      {props.productos.map(producto=><div>{producto.nombre}</div>)}
    </div>
  );
}

export default Home;
```

Propiedades

Como se ve en el ejemplo anterior las propiedades se acceden por el argumento **props.**

Este argumento es recibido como parámetro en el constructor y se invoca al método **super(props)**

El método super (en este caso) hace un llamado al constructor de la clase padre pasando el argumento props como parámetro, al hacerlo se inicializa la variable de clase **props** que se hereda de **Component.**

El llamado al componente y pasaje de propiedades desde el componente padre en ambos casos es igual

Crear un componente

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Crear componente

Para crear un componente como función debemos crear un nuevo archivo dentro de la carpeta src (puede estar dentro de un subdirectorio).

Importar la clase **React,** declarar una función con el mismo nombre del archivo generado **(nombre_componente).**

Por ultimo realizar un **export default nombre_component;**, el nombre de la función, archivo y modulo deben ser iguales.

Crear componente

Ejemplo de componente como función:

```
import React from 'react';

function Home() {

  return (
    <div >
      Hola Mundo
    </div>
  );
}

export default Home;
```

Crear componente

Para crear un componente como clase debemos crear un nuevo archivo dentro de la carpeta src (puede estar dentro de un subdirectorio).

Importar la clase **React y Component,** declarar una clase con el mismo nombre del archivo generado **(nombre_componente**) que herede de **Component**.

Por ultimo realizar un **export default nombre_component;**, el nombre de la clase, archivo y modulo deben ser iguales.

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Crear componente

Ejemplo de componente como clase:

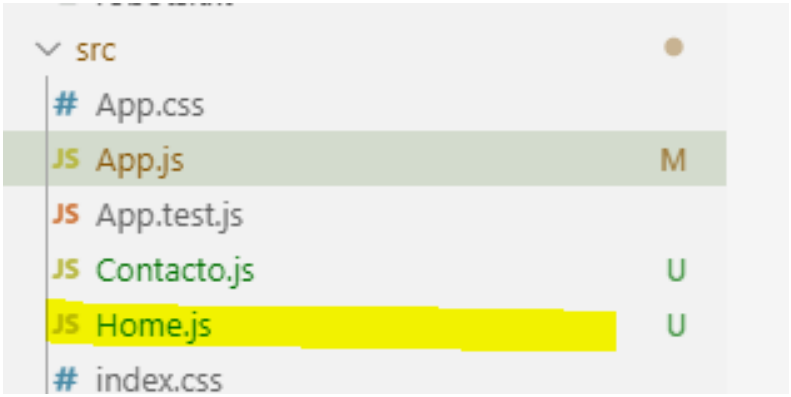
```
import React, { Component } from 'react';


class Contacto extends Component {

  render() {
    return (
      <div className="App">

        Contacto
      </div>
    );

  }
}

export default Contacto;
```

Llamar a un componente desde otro

Para llamar a un componente desde otro debemos importar el componente al cual queremos llamar (colocando el path del mismo) y luego en el JSX del componente padre crear la etiqueta **<nombre_componente />**

Nombre_componente: es el nombre de la clase o función, según como este declarado el mismo.

Ejemplo:



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Llamar a un componente desde otro

```
import React from 'react';
import Home from './Home'
function App() {

  return (
    <div className="App">
      <Home />
    </div>
  );
}

export default App;
```