Front End III

Estructura de una clase en React

Llegó el momento de estudiar la estructura de una clase en React. La clase más básica (y que funcione) que podemos escribir en React es esta:

```
import React from "react";

class App extends React.Component {
   render() {
       return (null);
   }
}
```

Una clase en React sigue las mismas reglas de las clases en ECMAScript, pero tiene una estructura propia y tiene algunas reglas que conviene entender y recordar. Veamos cuáles son:

1. La biblioteca React debe estar en Scope. Esto lo logramos empezando por importar React.
2. El nombre de la clase debe empezar con mayúscula y extender la clase base **React.Component**.

3. Se debe implementar la función de clase render retornando algo, aunque sea un null.

Esto es lo mínimo que React espera en una clase, pero así no nos sirve de mucho. La idea de escribir una clase es crear un componente que maneje su estado. De hecho, si el componente no va a manejar su propio estado, usar clases para definir el componente es prácticamente innecesario. Incluso, si el componente está pensado para manejar su propio estado, la forma preferida es usar componentes funcionales, pero por ahora concentrémonos en analizar la estructura de una clase en React.

Si nuestro componente recibirá propiedades, la manera de indicárselo a la clase es escribir un constructor que tome esas propiedades como argumento. Al hacer esto, es imprescindible que invoquemos al constructor de la clase base React.Component y le pasemos las propiedades. Esto hace que nuestra clase luzca así:

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

   render() {
       return (null);
   }
}
```

Es importante recordar que antes de usar this dentro del constructor debemos haber llamado a super que es el constructor de la clase base. Por eso, lo escribimos primero que todo dentro del constructor.

Dentro del constructor podemos hacer varias cosas, de hecho, hay dos tipos de código que encontraremos dentro del constructor: uno es para crear el estado; el otro es para declarar las funciones que serán llamadas desde otros componentes, es decir, funciones que si bien serán definidas dentro de la clase del componente, están pensadas para ser invocadas por otros componentes que serán creados desde nuestra clase.

Veamos ambos casos con un ejemplo. Supongamos que nuestra clase es un componente llamado **Pelicula**. Empecemos por el estado. Cada componente de clase **Pelicula** tendrá algunos datos que nos interesa manejar internamente: imagen, título, año, director, actores, género, puntaje y su dirección en IMDB (llamaremos a esta variable: imdb_URL). Estos datos conforman el estado del componente, y deben declararse y definirse dentro del constructor. La forma de hacerlo es por medio de la variable de clase state, a la cual accedemos por medio de this de esta manera: **this.state**.

La variable this.state es provista por la clase base de la cual heredamos y espera un objeto literal de JavaScript con cada una de las variables de estado. A las variables de estado debemos asignarles valores por defecto según el tipo de dato que manejarán:

```
import React from "react";

class Pelicula extends React.Component {
   constructor(props) {
       super(props);

       this.state = {
           imagen: "",
           titulo: "",
           año: 0,
           director: "",
           actores: [],
           genero: "",
           puntaje: 0,
           imdb_URL: ""
       };
   }

   render() {
       return (null);
   }
}
```

Ya tenemos las variables de estado asignadas. Ahora cabe preguntarnos: ¿qué queremos hacer con este estado? Esto nos lleva al otro tipo de código que encontraremos en el constructor: código para declarar funciones que serán llamadas desde otros componentes.

Habíamos hablado de dos funciones en nuestro ejemplo de plantillas. Para nuestro ejemplo real veremos qué significa esto en términos de cómo hay que escribir la clase, pero no escribiremos el código real por razones de simplicidad.

Primero veamos qué debemos meter en el constructor y por qué. Digamos que una función se llamará **seleccionarPelicula**, y la otra **verEnIMDB**. Es siempre buena práctica nombrar las funciones usando verbos que reflejen lo mejor posible lo que la función hace, sin importar si es muy largo el nombre. Por supuesto, siempre hay excepciones y hay que evitar nombres innecesariamente largos.

Estas dos funciones se deben declarar dentro del constructor utilizando **bind**. JavaScript es un lenguaje muy distinto a los demás lenguajes (como C++, JAVA, PHP, etc.), por esta razón debemos tener en cuenta que el contexto al que nos referimos cuando invocamos una función sea el correcto.

Para asignar el contexto correcto, una forma es usar **Function.prototype.bind()**. Hay otro mecanismo basado en arrow functions, pero bind es muy conveniente porque permite asignar el contexto al objeto al cual le queremos pasar la invocación de la función y será el mismo contexto sin importar cuántas veces se renderice el componente, es decir, no se destruirá y volverá a recrearse con cada renderizado. Veamos cómo queda el constructor:

```
constructor(props) {
    super(props);

    this.state = {
        imagen: "",
        titulo: "",
        año: 0,
        director: "",
        actores: [],
        genero: "",
        puntaje: 0,
        imdb_URL: ""
    };

    this.seleccionarPelicula = this.seleccionarPelicula.bind(this);

    this.verEnIMDB = this.verEnIMDB.bind(this);
}
```

Gracias a bind, se garantiza que estas dos funciones podrán ser invocadas por cualquier elemento o componente al que se le pasen y ejecutadas correctamente sin que salte un error.
Ahora debemos definir las funciones dentro de la clase con la funcionalidad que deseamos. Como dijimos, no hace falta el código, solo estamos interesados en la estructura de la clase. La clase ahora luce así:

```
import React from "react";

class Pelicula extends React.Component {

   constructor(props) {
       super(props);

       this.state = {
           imagen: "",
           titulo: "",
           año: 0,
           director: "",
           actores: [],
           genero: "",
           puntaje: 0,
           imdb_URL: ""
       };

       this.seleccionarPelicula = this.seleccionarPelicula.bind(this);

       this.verEnIMDB = this.verEnIMDB.bind(this);
   }

   seleccionarPelicula() {}

   verEnIMDB() {}

   render() {
       return (null);
   }
}
```

Las funciones que no están pensadas para que sean ejecutadas por otros componentes o elementos no necesitan ser vinculadas con bind. Simplemente basta con definirlas dentro de la clase sin la respectiva línea en el constructor.

Veamos ahora la función **render**. Por el momento estamos retornando null, pero la idea es retornar un elemento JSX con la funcionalidad que hemos escrito en la clase. En nuestro ejemplo, queremos mostrar la información de cada película y queremos cierta funcionalidad al hacer clic sobre el componente. Entonces, nuestro render podría retornar elementos HTML y usar las funciones de la siguiente manera:

```
render() {
    return (
        <li onClick={this.seleccionarPelicula}>
            <img src={this.state.image}
            alt={this.state.titulo + " image"}
            width="auto" height="200"
        />
            <div onClick={this.verEnIMDB}>
                <p> <b>Título:</b> {this.state.titulo} </p>
                <p> <b>Año:</b> {this.state.año} </p>
                <p> <b>Director:</b> {this.state.director} </p>
                <p> <b>Actores:</b> {this.state.actores.join(', ')} </p>
                <p> <b>Género:</b> {this.state.genero} </p>
                <p> <b>Puntaje:</b> {this.state.puntaje} </p>
            </div>
        </li>
    );
}
```

Al hacer clic en la película, invocaremos a **seleccionarPelicula**, y al hacer clic en cualquier parte de los datos, iremos al sitio de la película en IMDB.com.

Existen otras funciones que forman parte de la estructura de la clase y se usan para actualizar el estado del componente. Estas funciones forman parte del llamado ciclo de vida del componente. Este ciclo lo estudiaremos en detalle más adelante, por ahora basta mencionar que forma parte de la estructura de una clase React. Las dos funciones que se utilizan para actualizar el estado del componente son: **componentDidMount** y **componentDidUpdate**.

Por último, y no menos importante, debemos exportar nuestra clase para que sea accesible como un módulo, por otros módulos de la aplicación. La exportación de la clase puede tener dos formas. Con nombre:

```
export { Pelicula };
```

Sin nombre:

```
export default Pelicula;
```

En el primer caso, debe ser importada con los corchetes angulares y con el nombre exacto. Por ejemplo:

```
import { Pelicula } from "./Pelicula.jsx";
```

En el segundo caso, debe ser importada desde otros módulos sin los corchetes angulares y puede tener el nombre original o cualquier otro nombre. Con el nombre original sería así:

```
import Pelicula from "./Pelicula.jsx";
```

Con cualquier otro nombre, por ejemplo Movie, sería así:

```
import Movie from "./Pelicula.jsx";
```

Finalmente, nuestra clase luce de esta manera:

```
import React from "react";

class Pelicula extends React.Component {

   constructor(props) {
       super(props);

       this.state = {
           imagen: "",
           titulo: "",

           año: 0,
           director: "",
           actores: [],
           genero: "",
           puntaje: 0,
           imdb_URL: ""
       };

       this.seleccionarPelicula = this.seleccionarPelicula.bind(this);

       this.verEnIMDB = this.verEnIMDB.bind(this);
   }

   seleccionarPelicula() { }
```

```
  verEnIMDB() { }

  componentDidMount() { }

  componentDidUpdate() { }

  render() {
      return (
          <li onClick={this.seleccionarPelicula}>
              <img src={this.state.image}
              alt={this.state.titulo + " image"}
              width="auto" height="200"
          />
              <div onClick={this.verEnIMDB}>
                  <p> <b>Título:</b> {this.state.titulo} </p>
                  <p> <b>Año:</b> {this.state.año} </p>
                  <p> <b>Director:</b> {this.state.director} </p>
                  <p> <b>Actores:</b> {this.state.actores.join(', ')} </p>
                  <p> <b>Género:</b> {this.state.genero} </p>
                  <p> <b>Puntaje:</b> {this.state.puntaje} </p>
              </div>
          </li>
      );
  }
}

export { Pelicula };
```

Este es un ejemplo simple de la estructura más común de un componente de clase en React. Encontraremos al constructor con funciones que usan bind y donde se definen las variables de estado; usaremos las funciones de clase componentDidMount y componentDidUpdate; retornaremos elementos JSX en el render; y siempre habrá al menos una sentencia export para exportar la clase.

¡Hasta la próxima!