Componentes inmutables (práctica)

Tiempo estimado: 25min

El objeto de esta práctica es afianzar, reforzar y consolidar los conocimientos teóricos presentados en la lección.

Al finalizarla, el estudiante:

- Habrá creado componentes simples y compuestos.
- Habrá definido componentes mediante funciones y clases.

Objetivos

El objetivo de la práctica es crear una tabla HTML donde tanto la tabla como cada una de sus filas se implementarán mediante componentes. La tabla mostrará nombres de bandas musicales. La tabla se implementará mediante el componente Bands y las filas mediante Band.

Preparación del entorno

Para comenzar, vamos a crear un proyecto de aplicación React mediante el generador de Justo:

1. Abrir una consola.
2. Crear el directorio de la práctica e ir a él.
3. Invocar el generador:

   ```
   > justo -g react
   ```
4. Responder a las preguntas realizadas por el generador. A la hora de responder:
   - Indicar N cuando el generador pregunte si utilizar React Router.
5. Instalar las dependencias del proyecto:

   ```
   > npm install
   ```
6. Invocar la tarea build del catálogo del proyecto para construir la aplicación:

   ```
   > justo build
   ```
7. Abrir el archivo dist/index.html con el navegador, por ejemplo, mediante la tarea catalogada chrome del proyecto:

   ```
   > justo chrome
   ```
   O bien directamente:

   ```
   > google-chrome dist/index.html
   ```
   Debe aparecer el mensaje Hello World!

Componente simple mediante función

Para comenzar, vamos a crear un componente simple que muestre, mediante un elemento HTML <tr>, la información de una banda:

1. Ir a la consola.
2. Crear el componente simple Band mediante el generador:

   ```
   > justo -g react component
   ```
   Para responder, tenga en cuenta lo siguiente:
   - Indicar <none> como subcarpeta, dentro de app/components, donde crear el componente.
   - Indicar Band como nombre de componente.

- Indicar una descripción de componente. Aparecerá como comentario de documentación del componente.
- Indicar Immutable como tipo de componente.
- Indicar Simple como estructura de componente.
- Indicar Function para la implementación del componente.

3. Editar el archivo app/components/Band.jsx y echar un vistazo a la plantilla de partida creada por el generador.
4. Modificar la función para que el componente se represente en HTML como sigue:

```
<tr>
  <td>Nombre de la banda</td>
  <td>Año de formación</td>
  <td>Origen de la banda</td>
</tr>
```

Por ejemplo:

```
export default function Band(props) {
  return (
    <tr>
      <td>{props.name}</td>
      <td>{props.year}</td>
      <td>{props.origin}</td>
    </tr>
  );
}
```

5. Guardar cambios.

Componente compuesto mediante función

Ahora, vamos a crear un componente compuesto que represente una tabla de bandas.

1. Ir a la consola.
2. Crear el componente Bands con el generador:

```
> justo -g react component
```

A la hora de responder:

- Indicar <none> como subcarpeta donde crear el componente.
- Indicar Bands como nombre de componente.
- Indicar Immutable como tipo de componente.
- Indicar Composite como estructura de componente.
- Indicar Function para la implementación del componente.

3. Editar el archivo app/components/Bands.jsx y echar un vistazo a la plantilla de partida creada por el generador.
4. Modificar la función para que el componente se represente en HTML como sigue:

```
<table>
  <thead>
    <tr>
      <th>Banda</th>
      <th>Año de formación</th>
      <th>Orígen</th>
    </tr>
  </thead>

  <tbody>
    <tr>
      <td>Nombre de la banda</td>
      <td>Año de formación</td>
      <td>Origen de la banda</td>
    </tr>
```



```
    <!-- ... -->
  </tbody>
</table>
```

Para esta parte, supondremos que las filas se deben especificar como componentes dentro del contenido de la tabla. Ejemplo:

```
export default function Bands(props) {
  return (
    <table>
      <thead>
        <tr>
          <th>Banda</th>
          <th>Año de formación</th>
          <th>Origen de la banda</th>
        </tr>
      </thead>

      <tbody>
        {props.children}
      </tbody>
    </table>
  );
}
```

Observe el uso de la expresión {props.children}. Lo que se está diciendo es: *en este punto, reproduce el contenido indicado en el elemento instanciador*.

5. Guardar cambios.

Registro del componente raíz

A continuación, vamos a indicar como componente raíz o principal de la aplicación una instancia del componente Bands:

1. Editar el archivo app/index.jsx.

   Por convenio y buenas prácticas, el componente raíz se indica en este archivo.

2. Importar los componentes:

```
//imports
import React from "react";
import ReactDOM from "react-dom";
import App from "./views/App";
import Band from "./components/Band";
import Bands from "./components/Bands";
```

3. Crear la constante con los datos:

```
//data
const data = [
  {
    name: "Close Lobsters",
    year: 1985,
    origin: "Scotland"
  },
  {
    name: "Wild Swans",
    year: 1980,
    origin: "England"
  },
  {
    name: "The Smiths",
    year: 1982,
    origin: "England"
  }
];
```

4. Registrar como componente raíz o principal lo siguiente:

```
//root component
ReactDOM.render(
```

```
  <Bands>
    <Band {...data[0]} />
    <Band {...data[1]} />
    <Band {...data[2]} />
  </Bands>,
  document.getElementById("react-app")
);
```

Observe que el componente Bands se instancia mediante un elemento compuesto, el cual contiene las tres filas de datos que deseamos muestre.

5. Guardar cambios.
6. Construir la aplicación:

```
> justo build
```

7. Abrir la aplicación en el navegador.

   Debe aparecer lo siguiente:

Banda	Año de formación	Origen de la banda
Close Lobsters	1985	Scotland
Wild Swans	1980	England
The Smiths	1982	England

Componente compuesto mediante clase

Ahora, vamos a reimplementar el componente Bands mediante una clase:

1. Suprimir el archivo app/components/Bands.jsx.
2. Ir a la consola.
3. Crear el componente Bands con el generador. A la hora de responder, tenga en cuenta lo siguiente:
   - Indicar <none> como subcarpeta donde crear el componente.
   - Indicar Bands como nombre de componente.
   - Indicar Immutable como tipo de componente.
   - Indicar Composite como estructura del componente.
   - Indicar Class como medio de implementación.
4. Editar el archivo app/components/Bands.jsx y echar un vistazo a la plantilla de partida creada por el generador. Es muy diferente de la que generamos cuando implementamos el componente mediante una función.
5. Redefinir el método render():

```
render() {
  return (
    <table>
      <thead>
        <tr>
          <th>Banda</th>
          <th>Año de formación</th>
          <th>Origen de la banda</th>
        </tr>
      </thead>

      <tbody>
        {this.props.children}
      </tbody>
    </table>
  );
}
```

Observe que la definición es idéntica a la que vimos anteriormente, pero ahora hay que usar la propiedad props.children del objeto this, en vez del parámetro de la función.

6. Guardar cambios.
7. Editar el archivo app/index.jsx.
8. Modificar el componente raíz como sigue:

```
ReactDOM.render(
  <Bands>
    {
      data.map(function(band) {
        return <Band key={band.name} {...band} />;
      })
    }
  </Bands>,
  document.getElementById("react-app")
);
```

Este cambio no es necesario para el uso del componente. Lo hacemos para que el estudiante pueda ver otra manera de indicar el contenido de un componente compuesto.

No olvide la añadidura del atributo key. Indispensable cuando se añade contenido mediante el método map() de los *arrays*.

9. Guardar cambios.
10. Reconstruir la aplicación:

```
> justo build
```

11. Abrir la aplicación en el navegador.

Uso de las propiedades estáticas defaultProps y propTypes

Veamos ahora cómo trabajar con las propiedades defaultProps y propTypes. Recordemos que defaultProps tiene como objeto indicar los valores predeterminados de las propiedades del componente. Mientras que propTypes las restricciones de tipo y obligatoriedad. Por otra parte, no olvidemos que estas propiedades sólo se pueden definir cuando se utiliza una clase componente.

1. Generar el componente Band: inmutable, simple y mediante clase.

   Cuando el generador solicite si deseamos crear las propiedades, responder que sí.

2. Editar el archivo app/components/Band.js.

   Observe que se han definido las propiedades estáticas propTypes y defaultProps.

3. Modificar la definición de la propiedad propTypes por:

```
static get propTypes() {
  return {
    name: React.PropTypes.string.isRequired,
    year: React.PropTypes.number.isRequired,
    origin: React.PropTypes.string
  };
}
```

name y year son obligatorios; origin, opcional. Los valores de name y origin deben ser cadenas de texto, mientras que el de year, un número.

4. Modificar la definición de la propiedad defaultProps por:

```
static get defaultProps() {
  return {
    origin: "Desconocido"
  };
}
```

Si no se indica origin, React asignará el valor Desconocido.

5. Modificar el método de representación a:

```
render() {
  return (
    <tr>
      <td>{this.props.name}</td>
      <td>{this.props.year}</td>
```



```
        <td>{this.props.origin}</td>
      </tr>
    );
  }
```

6. Guardar cambios.
7. Editar el archivo app/index.jsx.
8. Modificar la definición de la constante de datos a:

```
const data = [
  {
    name: "Close Lobsters",
    year: 1985,
    origin: "Scotland"
  },
  {
    name: "Wild Swans",
    year: 1980,
    origin: "England"
  },
  {
    name: "The Smiths"
  }
];
```

   Observe que de The Smiths no se indica el campo obligatorio year, ni el opcional origin.
9. Guardar cambios.
10. Reconstruir la aplicación.
11. Abrir la aplicación en el navegador.

    Se espera:

Banda	Año de formación	Origen de la banda
Close Lobsters	1985	Scotland
Wild Swans	1980	England
The Smiths		Desconocido

    El valor de la propiedad origin de The Smiths la ha fijado React a partir de la información de la propiedad defaultProps.
12. Abrir la consola JavaScript del navegador.

    Debe aparecer un mensaje de error similar al siguiente:

    Este mensaje lo ha generado React a partir de la información indicada en la propiedad propTypes.

