

Fundamentos de React

Andre Rodriguez - 28 de Enero, 2020

Requisitos

- Tener conocimientos básicos de programación. Ojalá en Javascript.
- Tener conocimientos básicos de GIT.
- Tener NodeJs instalado.

Anuncios

- En este curso vamos a fallar y arreglar errores.
- Hasta al mejor programador@ se le olvidan las cosas. Lo importante son los conceptos.
- Queremos aprender a aprender por medio de la “repetición espaciada”
- Si tienen alguna duda, compartan la pregunta.
- El código de este curso se encuentra en <https://github.com/andrralv/react-fundamentals>

Programa

- El proyecto: Administrador de vacaciones para empleados
- Qué es React
- Instalación de React
- Historial de versiones de componentes en React
- JSX
- Creando nuestro primer componente
- Aplicando estilos a nuestro componente (CSS vs CSS in JS)
- Manejo del estado en componentes (Hooks)
- Principios de programación funcional
- Comunicación de datos con una API
- Manejo del ciclo del componente con el hook useEffect
- Manejo de colecciones de datos
- Aplicando el resto de funcionalidades a nuestra aplicación

El proyecto: Portal de vacaciones para empleados

Prototipos

Página Solicitud.	Página Empleados.
<p>Ada Lovelace Ingeniero@ Dras Disp: 8.</p> <div>Solicitudes < tabla ></div> <div>Crear Nueva Solicitud</div> <div><div>fecha inicio</div><div>fecha fin.</div><div>Dras a solicitar: <input type="text"/></div><div>Enviar solicitud.</div></div>	<p>➤ Alan Turing.</p> <div>< Solicitudes ></div> <p>➤ Ada Lovelace.</p> <p>➤ Brendan Eich.</p>

Qué es React

- Es una librería creada por Facebook que nos permite crear interfaces y pantallas para la web en Desktop y dispositivos móviles.
- Es parte de Web 2.0, lo cual por medio de REST, permite hacer transferencias de datos sin refrescar la pantalla.
- Existen otras plataformas como Angular, Vue y Svelte que hacen lo mismo de diferente manera.
- La primera versión de React fue lanzada el 29 de Mayo del 2013.

React tiene 2 principales beneficios:

- Permite un manejo del estado de la aplicación de manera asincrónica. Esto permite crear interfaces dinámicas que no requieren refrescar la pantalla.
- La sintaxis de JSX es mucho más descriptiva que el HTML, por lo que podemos crear markup más estructurado, modular, y re utilizable.

Particularidades de React

- React crea un DOM virtual y lo renderiza todo en un solo elemento del HTML.
- Remplaza jQuery por completo (y se recomienda no utilizarlo).
- Se complementa con otras librerías como Redux, Lodash, Axios, Immutable, etc.
- Es la interfaz de grandes aplicaciones como Facebook, Instagram, Airbnb, entre otras.

Instalación

- `npx create-react-app vacs-manager` o bajar el branch init del repositorio
- Continuar explicación del Scaffold en el video
- Referencia: `package.json`

Branches del repositorio

- **init:** Solo tiene la instalación de create-react-app
- **scaffold:** Tiene el primer componente y el scaffold del API agregados
- **finished:** Es toda la aplicación terminada
- **progress:** La que vamos a crear en este momento

Historial de versiones de componentes en React

Maneras de Crear un Componente

Código	Paradigma de Programación	Versión de React
<pre>var MyComponent = React.createClass({ render() { return <div> <h1>Hello World!</h1> <p>This is my first React Component.</p> </div> } }) ReactDOM.render(<MyComponent />, document.getElementById('react-component'))</pre>	Programación Imperativa	React 15 <
<pre>class MyComponent extends React.Component{ render() { return <div> <h1>Hello World!</h1> <p>This is my first React Component.</p> </div> } }</pre>	Programación Orientada a Objetos	React 15 a React 16.8
<pre>const MyComponent = () => { return <div> <h1>Hello World!</h1> <p>This is my first React Component.</p> </div> }</pre>	Programación Funcional	React 16.8 >

¿Qué es JSX?

- JSX es una abstracción de la sintaxis de HTML con Javascript anidada.
- Es similar a los lenguajes de templating como PUG, Jade, handlebars, PHP anidado, etc
- Facilita la integración de la lógica del programa en Javascript con lo que se muestra en el HTML.

Código JSX

Sin JSX

```
class Hello extends React.Component {
  render() {
    return React.createElement('div', null, `Hello ${this.props.toWhat}`);
  }
}

ReactDOM.render(
  React.createElement(Hello, {toWhat: 'World'}, null),
  document.getElementById('root')
);
```

Con JSX

```
const MyComponent = () => {
  return <div>
    <h1>Hello World!</h1>
    <p>This is my first React Component.</p>
  </div>
}
```

Creando Nuestro Primer Componente

Código

```
import React from 'react';
import './CartaEmpleado.css';

const CartaEmpleado = ({ datos }) => {
  return (
    <div className="carta-empleado">
      <div className="nombre">
        Nombre: { datos.nombre }
      </div>
      <div className="columna-datos">
        <div className="puesto">
          Puesto: { datos.puesto }
        </div>
        <div className="dias-disponibles">
          Dias disponibles: { datos.dias }
        </div>
      </div>
    </div>
  )
}

export default CartaEmpleado;
```

Aplicando Estilos a Nuestro Componente

CSS vs CSS in JS

CSS	<pre>import React from 'react'; import './CartaEmpleado.css'; const CartaEmpleado = ({ datos }) => { return (<div className="carta-empleado"> <div className="nombre"> Nombre: { datos.nombre } </div> </div>) }</pre>
CSS in JS	<pre>import React from 'react'; import { createUseStyles } from 'react-jss'; const useStyles = createUseStyles({ seccion: { width: '800px', fontWeight: 'bold', }, }); const CartaEmpleado = ({ datos }) => { const classes = useStyles(); return (<div className={classes.seccion}> </div>) }</pre>

Código

```
import React, { useState } from 'react';
import './CartaEmpleado.css';

const CartaEmpleado = ({ datos }) => {

  Const [nombre, setNombre] = useState('Joaquin');

  return (
    <div className="carta-empleado">
      <div className="nombre" onClick={() => setNombre('Marcos')}>
        Nombre: { nombre }
      </div>
      <div className="columna-datos">
        <div className="puesto">
          Puesto: { datos.puesto }
        </div>
        <div className="dias-disponibles">
          Dias disponibles: { datos.dias }
        </div>
      </div>
    </div>
  )
}

export default CartaEmpleado;
```


Características

- Es un paradigma de programación utilizado en lenguajes como Haskell y Smalltalk (del cual Javascript toma inspiración)
- Dado que Javascript es un lenguaje Multi-paradigma, se puede implementar programación funcional.
- Se basa en el principio de que todas las aplicaciones tienen un estado de datos al cual se le pueden aplicar transformaciones.

Fundamentos

- El estado inicial es inmutable.
- Cada función debe de hacer 1 sola cosa. (A esto se le llama pure function)
- Las funciones que hacen más de una cosa se les llama side-effect. La idea es minimizar esto al máximo.
- Las funciones pueden retornar otras funciones.
- El encadenamiento de funciones aplica las transformaciones.

Principios de Programación Funcional

Beneficios

- Más fácil de probar.
- Más fácil de refactorizar.
- Se puede aplicar progresivamente.
- Código más reducido.

Principios de Programación Funcional

Perjuicios

- La curva de aprendizaje es más abrupta.
- Sintáxis más compleja/matemática.
- No siempre se pueden evitar los side-effects.

Características

- React se puede comunicar con un servidor por medio del protocolo REST.
- Se puede utilizar `fetch()` nativo, u otra librería como Axios.
- La naturaleza de Javascript nos permite hacer peticiones asincrónicas.
- El hook `useEffect()` nos ayuda a llamar a una función cuando el componente se renderiza o recibe props.

Código

```
await axios({
  method: 'patch',
  url: `http://localhost:3001/empleados/${empleadoId}`,
  data: merge(datos, solicitud)
});

const response = await axios({
  method: 'get',
  url: `http://localhost:3001/empleados/${empleadoId}`,
});
console.log(response);
```

Código

```
import React, { useState, useEffect } from 'react';
import './CartaEmpleado.css';

const CartaEmpleado = ({ datos }) => {

  Const [nombre, setNombre] = useState('Joaquin');

  useEffect() => {
    llamarNombre();
  }, []);

  Const llamarNombre = () => {
    const respuesta = // hacer llamado al API con arios y guardar datos en el estado
    setNombre(respuesta.nombre);
  }

  return (
    <div className="carta-empleado">
      <div className="nombre" onClick={() => setNombre('Marcos')}>
        Nombre: { nombre }
      </div>
    </div>
  )
}
```

Comparación

<div>Paradigma Imperativo:</div> <div><div>- Se modifica el estado inicial</div><div>- Se pueden aplicar operaciones sobre otros datos dentro del for</div></div>	<pre>const estado = [1, 2, 3] for (var i=0; I< estado.length; I++) { estado[i] = i + 1; }</pre> <div>> 2, 3, 4</div>
<div>Paradigma Funcional:</div> <div><div>- Se retorna una copia del estado transformada</div><div>- Se aplican los cambios únicamente sobre el estado</div></div>	<pre>const estado = [1, 2, 3] const estadoMasUno = estado.map(item => item + 1);</pre> <div>> 2, 3, 4</div>

Recursos

- React DatePicker: [npmjs.com/package/react-datepicker](https://www.npmjs.com/package/react-datepicker)
- Issue con DatePicker: <https://github.com/HackerOx01/react-datepicker/issues/2677>
- Moment: <https://www.npmjs.com/package/moment>
- Lodash: <https://lodash.com/docs/4.17.15>
- Json Server: <https://www.npmjs.com/package/json-server>
- Documentación oficial: <https://reactjs.org/docs/>
- React-Router: <https://reactrouter.com/web/guides/quick-start>
- CSS in JS: <https://cssinjs.org/?v=v10.5.1>
- Axios: <https://www.npmjs.com/package/axios>