

Desarrollo en React JS

Introducción e instalación React JS

Desarrollador Web con React Js

Introducción

ReactJS es una librería JavaScript de código abierto enfocada a la visualización que nos permite el desarrollo de interfaces de usuario de forma sencilla mediante componentes interactivos y reutilizables. Está basado en un paradigma llamado **programación orientada a componentes** en el que cada componente es una pieza con la que el usuario puede interactuar.

Estas piezas se crean usando una sintaxis llamada **JSX** permitiendo escribir HTML (y opcionalmente CSS) dentro de objetos JavaScript. Estos componentes son reutilizables y se combinan para crear componentes mayores hasta configurar una web completa.

Esta es la forma de tener HTML con toda la funcionalidad de JavaScript y el estilo gráfico de CSS centralizado y listo para ser abstraído y usado en cualquier otro proyecto.

¿Cómo funciona el virtual DOM?



Imagina que tienes un objeto que es un modelo de una persona. Tienes todas las propiedades relevantes de una persona que podría tener, y refleja el estado actual de la persona. **Esto es básicamente lo que React hace con el DOM.**

Si tomamos ese objeto y le hacemos algunos cambios, agregamos un bigote, unos bíceps. En React, cuando aplicamos estos cambios, dos cosas ocurren:

- En primer lugar, React ejecuta un algoritmo de “diffing”, que identifica lo que ha cambiado.
- El segundo paso es la reconciliación, donde se actualiza el DOM con los resultados de diff.

Lo que hace React, ante estos cambios, en lugar de tomar a la persona real y reconstruirla desde cero, sólo cambiaría la cara y los brazos.

JSX es una extensión de sintaxis de Javascript, que se parece a HTML.

- Mejora la legibilidad del código, mejora la experiencia del desarrollador, y reduce la cantidad de errores de sintaxis, ya que no es necesario repetir tantas veces el mismo código.
- Otro beneficio es el de poder integrar a otros miembros de tu equipo que no sean programadores en el desarrollo. Es “fácil” leer el código si ellos están acostumbrados a leer HTML.

React no requiere usar JSX, pero la mayoría de la gente lo encuentra útil como ayuda visual cuando trabajan con interfaz de usuario dentro del código Javascript. Esto también permite que React muestre mensajes de error o advertencia más útiles.

Especificando atributos con JSX

Podemos utilizar comillas para especificar strings literales como atributos.

```
1 const element = <div tabIndex="0"></div>;
```

También puedes usar llaves para insertar una expresión JavaScript en un atributo:

```
1 const element = <img src={user.avatarUrl}></img>;
```

Dado que JSX es más cercano a JavaScript que a HTML, React DOM usa la convención de nomenclatura camelCase en vez de nombres de atributos HTML.

Por ejemplo, **class** se vuelve **className** en JSX, y **tabindex** se vuelve **tabIndex**.

create-react-app



create-react-app configura tu ambiente de desarrollo de forma que puedas usar las últimas características de Javascript, brindando una buena experiencia de desarrollo, y optimizando tu aplicación para producción.

Necesitarás tener Node mayor o igual a la versión 10.16 y npm mayor o igual a la versión 5.6 instalados en tu máquina.

Para crear un proyecto ejecuta:

```
create-react-app nombredemiproyecto cd nombredemiproyecto npm start
```

create-react-app no se encarga de la lógica de backend o de bases de datos; tan solo crea un flujo de construcción para frontend, de manera que lo puedes usar con cualquier backend

Creación de componentes

Podemos definir un componente usando una simple función, y su valor de retorno será el contenido HTML que se muestre.

Así mismo podemos incluir todos los componentes que necesitemos dentro del valor de retorno de otros componentes. Este concepto se llama composición.

Por lo general, las aplicaciones de React nuevas tienen un único componente App en lo más alto.

Es importante que en cada archivo en el que vayamos a definir un componente (ya sea de clase o funcional recordemos importar React al principio de nuestro archivo js y exportarlo al final.

Componentes de presentación vs 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.
- 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 para despachar alguna acción y que el usuario vea el cambio en los datos.

Presentación:

- Orientados al aspecto visual
- No tienen dependencia con fuentes de datos.
- Reciben callbacks por medio de props
- Pueden ser descritos como componentes funcionales.


Contenedores:

- Orientados al funcionamiento de la aplicación
- Contienen componentes de presentación y/o otros contenedores
- Se comunican con las fuentes de datos
- Usualmente tienen estado para representar el cambio en los datos

Componentes de clase vs. componentes de función


Los componentes de React pueden definirse de dos formas distintas:

- como funciones
- como clases



```
1 function Saludo(props) {  
2   return <h1>Hola, {props.name}</h1>;  
3 }
```

Ejemplo de componente de función



```
1 class Saludo extends React.Component {  
2   render() {  
3     return <h1>Hola, {this.props.name}</h1>;  
4   }  
5 }
```

Ejemplo de componente de clase

Ambos componentes son equivalentes, sin embargo la forma más sencilla, y la más utilizada en las últimas actualizaciones es de función.

Render

Todo componente de clase en React, 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.

```
import React from 'react'

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

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

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

React DOM compara el elemento y sus hijos con el elemento anterior, y solo aplica las actualizaciones del DOM que son necesarias para que el DOM esté en el estado deseado.

Gracias!