Componentes en React

Las aplicaciones en React se construyen mediante componentes.

El potencial de este funcionamiento consiste en que podemos crear aplicaciones completas de una manera modular y de fácil mantenimiento,a pesar de su complejidad.

Pero en sí, para javascript, un componente no es más que una función que recibe un objeto, al que llamaremos “props”, y retorna un elemento de React que describe que va a aparecer en la pantalla.

React admite 2 tipos de componentes, los funcionales y los de clase, nosotros trabajaremos con componentes del tipo funcional, ya que en la actualidad se trabaja de esa forma.

Componente de clase:

```jsx
import React from 'react';

class Clase extends React.Component {
    constructor() {
      super();
      this.state = {saludo: "Hola! Soy un componente de clase"};
    }
    render() {
      return <h1>{this.state.saludo}</h1>;
    }
}

export default Clase;
```

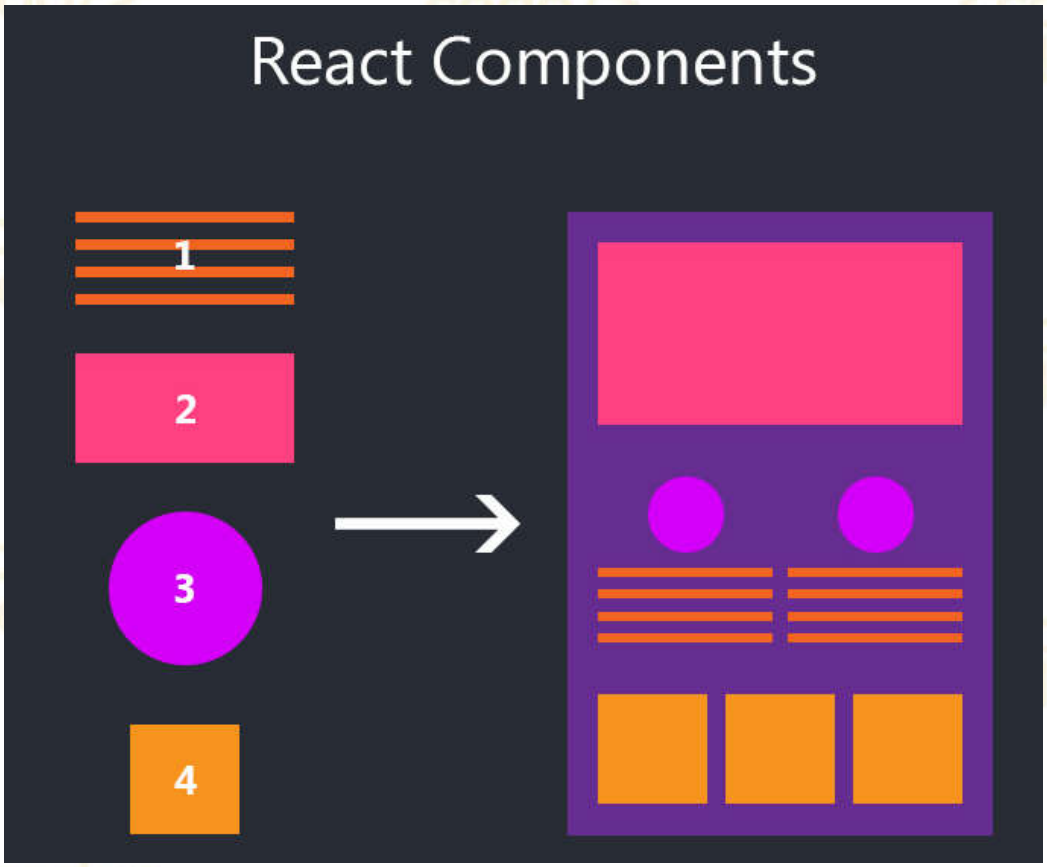
Componente funcional:

```jsx
import React from 'react';

function Funcional() {
  const saludo = 'Hola! Soy un componente funcional';

  return <h1>{saludo}</h1>;
}

export default Funcional;
```

Diseño Modular

Los componentes permiten separar la interfaz de usuario en piezas independientes,reutilizables y pensar en cada pieza de forma totalmente aislada.

Al desarrollar crearemos componentes para resolver pequeños distintos problemas, que son fáciles de visualizar y comprender durante el proceso de desarrollo, por lo cualnos ahorrar mucho tiempo.



Los distintos componentes se apoyarán en otros para solucionar problemas mayores yal final la aplicación será un conjunto de componentes que trabajan entre sí.

Este modelo de trabajo tiene varias ventajas, como la facilidad de mantenimiento,depuración, escalabilidad, etc.

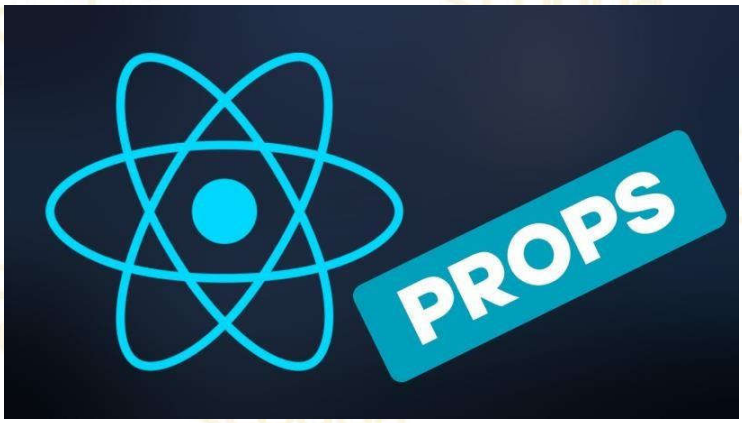
A menudo, las actualizaciones se convierten en un problema porque la aplicación tieneuna lógica compleja y los cambios en un componente pueden afectar a otros. Para resolver el problema, Facebook ha complementado React con la capacidad de reutilizar componentes del sistema, y los desarrolladores lo definen como una de las mejores características de React js.

La reutilización de activos es bien conocida entre los diseñadores, que suelen volver a emplear los mismos objetos digitales. Puede comenzar con los componentes más finos(casilla de verificación, botón, etc.), luego pasar a los componentes de envoltura compuestos por estos pequeños elementos y avanzar hasta el componente raíz principal. Todos los componentes tienen su lógica interna, lo que facilita su manipulación y definición. Este enfoque garantiza un aspecto uniforme de la aplicacióny facilita el mantenimiento y el crecimiento de la base de código.

Ventajas

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

Propiedades en React



En React, los **Props** se refieren a las propiedades las cuales cumplen un rol importanteen el proceso de desarrollo de una aplicación o página web. Los componentes son los bloques de construcción de React y estos componentes usan Props para mejorar su funcionalidad y poder reusar el código.

¿Cuál es el propósito de los Props de React?

Como ya sabemos, React es uno de los frameworks JavaScript más usados en los últimos años. Los Props tiene una importante función: ellos pasan los datos de un componente a otro, ofreciendo así un canal por el medio del cual los componentes sepuede comunicar.

Hay una regla que debes aprender antes de comenzar a usar los Props de React: todoslos componentes deben funcionar de la misma manera que una función pura (con respecto a los props).

Cómo usar los Props de React para hacer un encabezado

Primero, debes pasar el Prop como un atributo al componente funcional, luego de estopodrás tener acceso al Prop y a los datos dentro del componente.

Usar un Prop en el componente de encabezado

```
function Header(props) {
return (
<div>
<h1> {props.title} </h1>
</div>
);
}
export default Header;
```

El componente de encabezado anterior toma un atributo prop y lo usa para acceder alos datos sobre el título de la aplicación. Para que este componente se vea en la interfaz del usuario, debes insertarlo en el archivo App.js de React.

El archivo App.js

```
import Header from
'./components/Header';function App() {
return (
<div>
<Header title='administrador' />
```

);

}

export default App;

Este código muestra el archivo App.js en React, el cual se procesa en la interfaz de usuario. El componente de la aplicación muestra el Header o Encabezado en la interfazusando el tag "Header" y este tag contiene un Prop y un valor de Prop, por lo que ahora tiene acceso a una propiedad de título que se puede usar en esta sección.

Usar Props predeterminados

El archivo App.js actualizado debe verse

así: import Header from

'./components/Header';function App() {

return (

<Header/>

);

}

export default App;

Al hacer esto, la interfaz de usuario mostrará un encabezado en blanco pero hay unasimple manera de solucionar este problema. Añadiendo un Prop

predeterminado al componente que utiliza el Prop, puedes efectivamente resolver el problema. Inclusoasí el componente no reciba un Prop, siempre tendrá uno con el que pueda trabajar.

Ejemplo de uso de un Prop predeterminado

```
function Header(props) {
return (
<div>
<h1> {props.title} </h1>
</div>
);
}
Header.defaultProps ={
title: 'Nombre de App'
}
export default Header;
```

Con este código, el encabezado ahora podrá utilizar un Prop predeterminado para mostrar el encabezado en la interfaz de usuario, sin necesidad de insertar un Prop solopara esta función en el archivo App.js.

Tipos de componentes en React

Cuando creamos un proyecto en React mediante la instrucción create-react-app partiremos de un componente base en el cual se agrupan diferentes

funcionalidades ycaracterísticas como la función render , las props , su propio state o los lifecycle eventso su propio contexto:

Sin embargo, los componentes que habitualmente emplearemos tan solo emplearánuna parte de todas esas API's; de hecho, podemos separarlos en dos grandes grupos:

Aquellos componentes que emplean la función render junto con las props y el contextasociado

Los componentes que emplean la función render, el state y los lifecycle events

Es a partir de este punto en el que surgen los dos primeros tipos de componentes.

Componentes de presentación

Son aquellos que simplemente se limitan a mostrar datos y tienen poca o nula lógicaasociada a manipulación del estado (por eso son también llamados stateless components).

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

Este tipo de componentes, probablemente el más básico que podemos definir, y quetambién es conocido como stateless components, se encarga tan solo de pintar los elementos en base a las props recibidas por los otros componentes donde aparezcanembebidos. De ahí que también reciban el sobrenombre de stateless ya que ellos mismos no almacenan ningún tipo de estado, limitándose a presentar los datos por pantalla. De hecho, a menudo los definiremos directamente como funciones que devuelven el HTML correspondiente sin necesidad de extender la clase React.Component

La ventaja más evidente de separar estos componentes del resto es la posibilidad de reutilizarlos siempre que queramos sin tener que recurrir a escribir el mismo código una y otra vez

Componente contenedor

Tienen como propósito encapsular a otros componentes y proporcionarles las propiedades que necesitan.

Además, se encargan de modificar el estado de la aplicación para que el usuario vea el cambio en los datos (por eso son también llamados state components).

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

Por otra parte, los container components son aquellos componentes que sí emplean la API state para establecer la lógica y el funcionamiento de nuestra aplicación, de ahí que también sean conocidos como state components.

Este tipo de componentes, por tanto, serán los encargados de realizar llamadas a las API's externas, conectarse a Redux o establecer la lógica a realizar en función de las acciones que realice el usuario sobre la interfaz.

Render props

Finalmente, el último tipo de componente o patrón (yo prefiero en este caso emplear este último término) que podemos emplear a la hora de definir la estructura de nuestra aplicación es el que se conoce como render props.

La característica de este tipo de componentes es que reciben una función a través de las props (la cual devuelve un elemento de React) y que será la que empleen dentro de