ThePower business school

🇪🇸 Props ReactJS

Introducción

Los `props` de React consisten en delegar lo que un componente va a renderizar a otro componente, es decir, pasar información de un componente padre a un componente hijo.

La forma más adecuada de enviar datos es unidireccional y top-down, es decir, de componentes padres a componentes hijos. Además de esto debemos tener en cuenta que estos datos serán inmutables, por lo que el componente que recibe las `props` solamente podrá leer la información y no sobrescribirla.

¿Cómo se envía un prop?

Como ejemplo vamos a crear un componente llamado `Greeting` y haremos que diga "Hola" y nuestro nombre. Este componente lo utilizaremos directamente desde `App` siguiendo la siguiente estructura.

```
import React from "react";
import { Greeting } from "./components";
import "./App.scss";

const App = () => {
  return (
    <div className="App">
      <Greeting name="Siegmeyer of Catarina"/>
    </div>
  );
};

export default App;
```

En este caso, al componente le estamos pasando una `prop` llamada `name` con el valor `"Siegmeyer of Catarina"`. Otro ejemplo para hacer exactamente lo mismo y entender que tipo de valores se pueden pasar por props es declarando una variable del mismo valor justo antes del return, de tal forma que el valor que le daremos a la prop será dicha variable a través de las conocidas llaves.

```
import React from "react";
import { Greeting } from "./components";
import "./App.scss";

const App = () => {

  const name = "Siegmeyer of Catarina";

  return (
    <div className="App">
     <Greeting name={name}/>
    </div>
  );
};

export default App;
```

Una vez inyectado el prop desde nuestro componente padre solo nos queda invocarlo en el componente hijo, en este caso en `Greeting`

```
import React from "react";

const Greeting = (props) => {
  return (
    <div>
      <p>Hello {props.name}!</p>
    </div>
  );
};

export default Greeting;
```

Como podemos ver en nuestro código, al componente `Greeting` le hemos pasado como argumento las `props`, llamando a través de las llaves a todas las props que estemos

utilizando e indicando que necesitamos el `.name` declarado en el componente padre.

Una forma más limpia de invocar nuestros `props` es haciendo destructuring justo antes del return.

```
import React from "react";

const Greeting = (props) => {
  const { name } = props;
  return (
    <div>
      <p>Hello {name}!</p>
    </div>
  );
};

export default Greeting;
```

Así solo tendremos que llamar a `name` en vez de `props.name`, teniendo un código más limpio y legible. Esto es muy útil si pasamos más de un prop a un componente.

Y de esta forma combinamos los dos ejemplos con ES6.

```
import React from "react";

const Greeting = ({name}) => {
  return (
    <div>
      <p>Hello {name}!</p>
    </div>
  );
};

export default Greeting;
```

De esta forma, y concatenando un texto con el valor de nuestro prop `name` tendremos el siguiente resultado en nuestra aplicación.

```
Hello Siegmeyer of Catarina!
```

Mini-ejercicio

Ha llegado el momento de ponerse a trabajar con ReactJS, para ello os proponemos una pequeña práctica que os ayude afianzar el concepto de props.

1. Crea una aplicación de ReactJS con vite → name: components-props—basics.
2. Crea tu carpeta de `components` dentro de `src`.
3. Realizamos algunos componentes de ReactJS:
    a. Componente Title ⇒ Crea un componente Title que retorne un `<h1>` con un texto recibido por props.
    b. Componente SubTitle ⇒ Crea un componente que retorne un `<h2>` con un texto recibido por props.
    c. Componente Image ⇒ Crea un componente que retorne un `<image>` con un src y alt recibido por props || además también recibirá el with y el height.
    d. Componente Paragraph ⇒Crea un componente que retorne un `<p>` con un texto recibido por props.
4. Estila cada uno de ellos haciendo uso de CSS Modules → hoja de estilo asociada al componente.
5. Exporta los componentes en un `index.js` e importalos en `App.jsx`.
6. Puedes usar los componentes tantas veces como quieras pero podéis escribir un pequeño blog repitiendo SubTitle, Image y Paragraph.
7. Comprueba que la visualización es correcta.