Instituto Pío IX Casa Salesiana

Desarrollo de Aplicaciones Informáticas

Tema: Creación de `componentes compuestos` en React

Docente: Ing. Pablo Morandi
Ayudante: Matías Marchesi
Curso: 5to año
Especialidad: Informática

Redactado por: Matías Marchesi

---

1. ¿Qué son los Componentes Compuestos?

Los **componentes compuestos** son componentes que están formados por otros componentes más pequeños. Es como construir con bloques: tomamos piezas simples y las combinamos para crear algo más complejo.

Ventajas de los componentes compuestos:

- **Reutilización**: Los componentes pequeños se pueden usar en diferentes lugares
- **Organización**: Cada componente tiene una responsabilidad específica
- **Mantenimiento**: Es más fácil modificar partes específicas
- **Escalabilidad**: Podemos agregar más funcionalidades fácilmente

Ejemplo conceptual: Un formulario (componente compuesto) puede estar formado por:

- Títulos
- Campos de entrada
- Botones

---

2. Props - Propiedades de los Componentes

Las **props** (properties) son la forma en que pasamos información entre componentes. Son como los parámetros de una función, pero para componentes de React.

¿Para qué sirven?

- Personalizar el comportamiento de un componente
- Pasar datos de un componente padre a un componente hijo
- Hacer componentes más flexibles y reutilizables

Dos formas de usar props:

```
// 1. Desestructuración directa
export default function Button({ onClick, text }) {
    return (
        <button onClick={onClick}>
            {text}
        </button>
    );
}

// 2. Usando el objeto props completo
export default function Button(props) {
    return (
        <button onClick={props.onClick}>
            {props.text}
        </button>
    );
}
```

> **Nota**: La primera forma (desestructuración) es más común y limpia, pero ambas funcionan igual.

---

3. Creando Componentes Base

Antes de armar componentes compuestos, necesitamos crear los componentes básicos que vamos a combinar.

Componente Button

Creamos el archivo `src/app/components/Button.js`:

```
export default function Button({ onClick, text }) {
  return (
    <button onClick={onClick} className="btn">
      {text}
    </button>
  );
}
```

Componente Title

Creamos el archivo `src/app/components/Title.js`:

```
export default function Title({ text }) {
  return <h1>{text}</h1>;
}
```

Explicación del componente Title:

- Recibe la prop `text` y renderiza el tag `<h1>`

---

4. Armando un Componente Compuesto

Ahora creamos un formulario que combina nuestros componentes básicos.

Componente Form

Creamos el archivo `src/app/components/Form.js`:

```
import Button from "./Button";
import Title from "./Title";

export default function Form({ title, buttonText, onButtonClick }) {
  return (
    <div className="form-container">
      <Title text={title} />

      <div className="form-fields">
        <input
          type="text"
          placeholder="Ingresa tu nombre"
          className="input-field"
        />
        <input
          type="email"
          placeholder="Ingresa tu email"
          className="input-field"
        />
      </div>

      <Button text={buttonText} onClick={onButtonClick} />
    </div>
  );
}
```

¿Qué hace este componente?

- Combina `Title` y `Button` que creamos antes
- Agrega campos de entrada (inputs)
- Recibe props para personalizarlo
- Pasa las props correspondientes a cada componente hijo

5. Usando el Componente en las Páginas

Ahora podemos usar nuestro componente `Form` en cualquier página de nuestra aplicación.

Página de Login

Creamos `src/app/login/page.js`:

```
import Form from "../components/Form";

export default function LoginPage() {
  const handleLoginClick = () => {
    console.log("Usuario intentando hacer login");
    // Aquí iría la lógica de login
  };

  return (
    <div className="page-container">
      <Form
        title="Iniciar Sesión"
        buttonText="Entrar"
        onButtonClick={handleLoginClick}
      />
    </div>
  );
}
```

Página de Registro

Podemos reutilizar el mismo componente con diferentes props:

```
import Form from "../components/Form";

export default function RegisterPage() {
  const handleRegisterClick = () => {
    console.log("Usuario creando cuenta nueva");
    // Aquí iría la lógica de registro
  };

  return (
    <div className="page-container">
      <Form
        title="Crear Cuenta"
        buttonText="Registrarse"
        onButtonClick={handleRegisterClick}
        titleLevel={1}
      />
    </div>
```

```
  );
}
```

---

6. Funcionalidad con onClick

Los eventos en React se escriben en **camelCase** (diferente al HTML tradicional).

HTML tradicional:

```
<button onclick="miFuncion()">Clic aquí</button>
```

React/JSX:

```
<button onClick={miFuncion}>Clic aquí</button>
```

Ejemplo con Funciones Más Complejas

```
export default function LoginPage() {
  const handleLoginClick = () => {
    // Podemos hacer validaciones
  };

  const handleSpecialAction = () => {
    console.log("Acción especial ejecutada");
  };

  return (
    <div className="page-container">
      <Form
        title="Bienvenido al Sistema"
        buttonText="Ingresar al Sistema"
        onButtonClick={handleLoginClick}
        titleLevel={1}
      />

      {/* Podemos agregar más elementos */}
      <button onClick={handleSpecialAction}>Acción Especial</button>
    </div>
  );
}
```

Flujo Completo de Props

```
// 1. La página define la función
const handleLoginClick = () => { /* lógica */ };

// 2. Se la pasa al componente Form
<Form onButtonClick={handleLoginClick} />

// 3. Form la recibe y se la pasa a Button
<Button onClick={onButtonClick} />

// 4. Button la ejecuta cuando se hace clic
<button onClick={onClick}>
```
