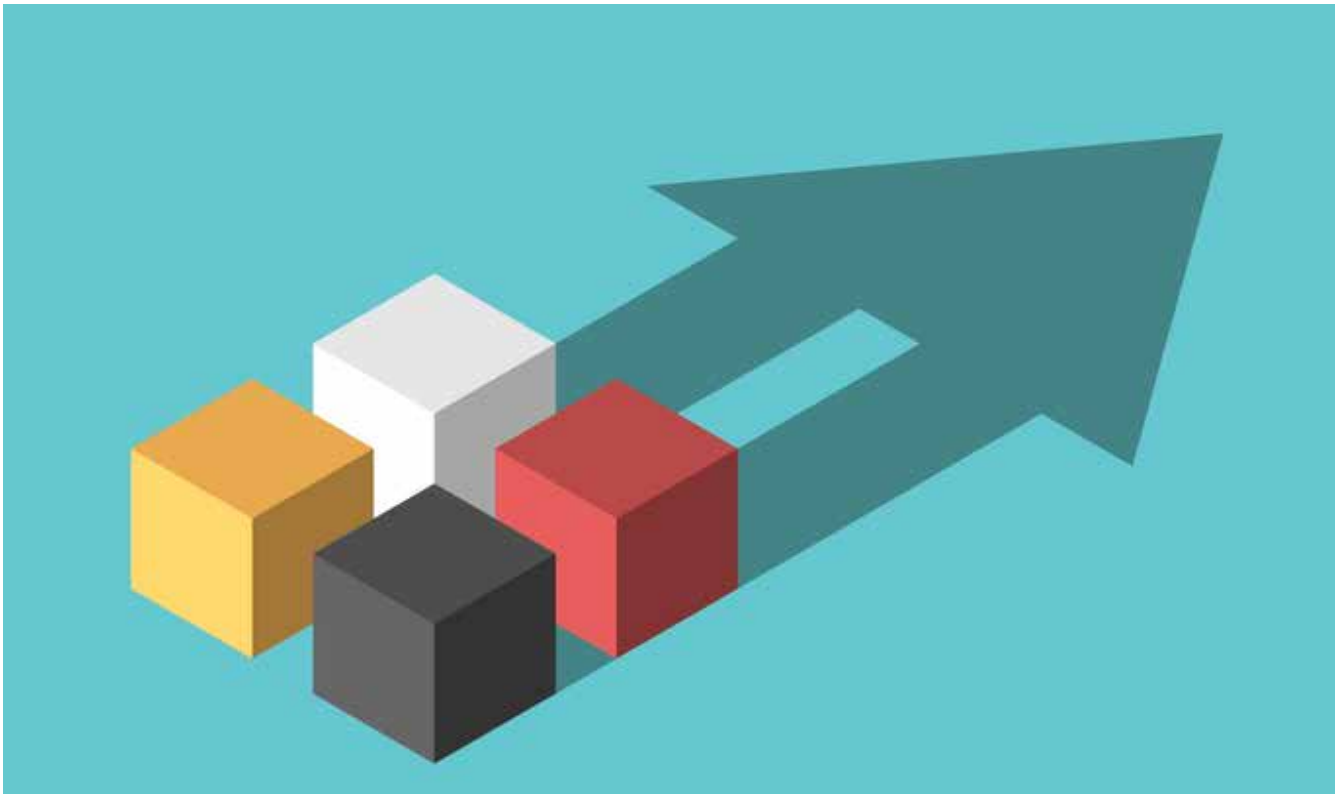
React Nivel Avanzado

Eventos, estados y contexto

Introducción



En React, el uso de componentes que permite reutilizar códigos, de tal forma que se puedan personalizar los elementos de la interfaz de acuerdo a las necesidades del usuario, para lograr esto, los componentes deben ser capaces de personalizarse.

Esto se logra mediante las propiedades (props), las propiedades se declaran como parámetros dentro de la declaración del componente (usualmente se le llama props).

Para trabajar los datos de manera interna, se hará uso de los estados o *states*, que permiten reaccionar a los cambios dentro de un componente.

Si se desea compartir información a diferentes niveles se usarán contextos para evitar diferentes complicaciones que se pueden presentar al tratar de solucionar lo mismo mediante el uso de propiedades y que los estados no permiten de la misma forma.

Se puede notar, como cada característica de React permite realizar algo de forma más sencilla, y conocer estas herramientas permitirá optimizar de mejor manera la aplicación y evitar caer en malas prácticas al abusar de una sola herramienta que posiblemente no esté optimizada para realizar una función que puede hacer otra creada para el mismo objetivo.

Explicación

Se puede compartir información entre componentes mediante el uso de propiedades, manejo de estados, o usando contextos, cada uno de estos tiene su propia utilidad, por lo que un uso combinado de estas herramientas proporciona mayor flexibilidad al trabajar con los componentes.

```
Los ejemplos a mostrar están hechos en ReactJS.
Para seguir los pasos se recomienda crear un proyecto vacío
usando el comando

npx create-react-app Hello
```

Props

Los componentes usan las propiedades para comunicarse unos con otros. Cada componente padre puede pasar información a un componente hijo mediante el uso de propiedades. Las propiedades se usan de forma similar a como se hace en los atributos HTML, a diferencia de que en React se puede enviar cualquier tipo de dato, como objetos, arreglos y funciones. (React Docs, 2023)

Un ejemplo sencillo sería el siguiente:
Se debe crear un archivo Greeting.jsx donde se desarrolle el componente Greeting.

```
function Greeting ({ name }) {
    return <div> Hello {name}! </div>
}
```

En el código anterior se definió un componente para mostrar un saludo, este componente recibirá el nombre de la persona a quien se le saludará mediante la propiedad **name**. Este mismo valor se usará para imprimir dentro del return del componente.

Ahora se denominará al componente **Greeting** desde **App.js** se desea mostrar y se le pasa el valor dentro de la propiedad **name** que ya se definió previamente.

```
<Greeting name="World" />
```

Explicación

Eventos

Al igual que en el ejemplo anterior, se puede hacer lo mismo para los eventos de los elementos HTML, enviando métodos de Javascript a través de las propiedades.

Un ejemplo de esto es el evento onClick, el cual se puede definir de la siguiente manera.

Primero, se debe crear el componente MyButton, el cual recibirá dos propiedades: **label y onClick.**

```
function MyButton({ label, onClick }) {
    return <button onClick={onClick}>{label}</button>
}
```

Del lado donde se invoca quedaría de la siguiente manera:

```
function App() {
  const handleClick = () => {
    alert('clicked')
  }

  return (
    <div className="App">
      <MyButton label="Click me" onClick={handleClick} />
      <Greeting name="World" />
    </div>
  );
}
```

Aquí, el evento **handleClick**, es el método que se enviará a través de la propiedad **onClick** del componente **MyButton** y se recibirá y asignará al atributo onClick del elemento button HTML.

De la misma manera se puede enviar cualquier otro evento que los elementos HTML soporten, pueden ser onClick, onBlur, onChange, onFocus, entre otros.

Explicación

Estados

React proporciona una variedad de funciones especiales que permiten brindar nuevas capacidades a los componentes, como el estado. Estas funciones se llaman ganchos, y el gancho useState, como su nombre lo indica, es precisamente el que se requiere para darle algún estado al componente. (Mozilla Developer Network, 2023)

Los estados dan a los componentes la interacción que necesitan para lograr que la app pueda modificar ciertos valores al momento, mientras se está utilizando la aplicación y esto facilita que se puedan realizar ciertas acciones al estar observando los cambios en tiempo de ejecución.

La forma en que se crea un componente con estado es la siguiente:

```
function TextField() {
    const [inputValue, setInputValue] = useState('');

    const handleInputChange = ({ target }) => {
        setInputValue(target.value)
    }

    return (<input type="text" value={inputValue} onChange={han-
dleInputChange} />)
}
```

Del código anterior se destaca lo siguiente:

- Usar el Hook **useState** para declarar inputValue y setInputValue.
- **inputValue** es la variable de estado.
- **inputValue** se inicializa dentro del hook como una cadena vacía.

El método **setInputValue** servirá para actualizar el valor de la variable **inputValue**, lo cual permitirá que React reconozca el cambio al estar observando los cambios de estado en este componente.

Además de lo anterior también se nota que nuestro input tiene un método onChange, este método al dispararse toma el valor del input HTML y lo asigna a la variable de estado, para que se pueda tener el mismo valor en el estado.

Explicación

Contexto

Como se mencionó anteriormente, es posible pasar valores de un componente padre a un componente hijo mediante el paso de propiedades, pero en ocasiones es necesario enviarle información a componentes que se encuentran dentro de niveles más abajo en el árbol del virtual DOM, dando como resultado enviar las propiedades a varios elementos hasta llegar al que necesitamos enviar la información, esto se conoce como *Prop drilling* (debido a que se pasa a través de varias capas de componentes).

Para solucionar esto, se creó el contexto.

Para crear un contexto, primero se debe crear un archivo que se pueda importar en los componentes para poder usarlo.

El siguiente es el archivo **UserContext.jsx**

```
import { createContext } from 'react';

export const UserContext = createContext();
```

En este ejemplo el contexto será usuario.

Greeting.jsx

```
import React, { useContext } from 'react';
import { UserContext } from './UserContext';

function Greeting({ name }) {
    const usercontext = useContext(UserContext)

    return <div>
        Hello {name}! <br />
        Context: <b>{usercontext}</b>
    </div>
}
export default Greeting
```

Ahora el código crece al aumentar los imports, y ahora incluir los hooks de useContext, donde se toma el valor del contexto **UserContext**, en este ejemplo el valor no se actualiza, pero podría enviar un objecto con el método set en lugar de sólo una cadena de texto.

Explicación

El archivo **App.jsx** quedaría de la siguiente forma:

```
import Greeting from './Greeting';
import MyButton from './MyButton';
import { UserContext } from './UserContext';

function App() {
  const handleClick = () => {
    alert('clicked')
  }
return <div className="App">
      <UserContext.Provider value="World wide">
        <MyButton label="Click me" onClick={handleClick} />
        <Greeting name="World" />
      </UserContext.Provider>
    </div>
}

export default App;
```

Como ya se mencionó, se pueden enviar objetos mediante el atributo value del provider, de la siguiente manera:

```
<UserContext.Provider value={{value, setValue}}>
```

Esto permite enviar métodos para poder realizar otras acciones desde cualquier otro componente donde se requiera usar estos métodos y valores.

Cierre

El conocer las herramientas y características con las que contamos al desarrollar en React, facilitará la resolución de problemas al permitir realizar lo que se desea de forma más eficiente, para que en lugar de tener que dedicar tiempo a solventar carencias de la librería, mejor se puedan realizar actividades que realmente requieran esfuerzo.

Se debe elegir la herramienta adecuada de acuerdo a la situación que se requiera solucionar, esto para optimizar de mejor manera los recursos disponibles.

Referencias bibliográficas

- Mozilla Developer Network. (2023). *React interactivity: Events and state*. Recuperado de https://developer.mozilla.org/en-US/docs/Learn/Tools_and_testing/Client-side_JavaScript_frameworks/React_interactivity_events_state
- React Docs. (2023). *Passing Props a Componente*. Recuperado de https://react.dev/learn/passing-props-to-a-component

Para saber más

Videos
Para conocer acerca de los efectos secundarios de los cambios de estados te recomendamos:

- Agustin Navarro Galdon. (2020, 2 de abril). *React useEffect en 5 minutos* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=7qnRm9F2x50

- Appdelante. (2022, 29 de junio). *useEffect explicado a fondo* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=0_D8ruGVp20

Checkpoint

Asegúrate de:

- Comprender el paso de valores mediante *props*.
- Conocer el uso de los *states*.
- Entender el uso de los hooks useState.

Requerimientos técnicos

- Android Studio
- React
- Xcode

Prework

- Deberás revisar lo siguiente:
  - **Tema 1. Fundamentos de React.**
  - **Tema 2. Programando con React Native.**
  - **Tema 3. Componentes Core y Nativos**
  - **Tema 4. Maquetando nuestra App**
  - **Tema 5. Eventos, Estados y Contexto**

La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.