Argentina programa 4.0

UNIVERSIDAD TECNOLOGICA NACIONAL

Desarrollo en React JS

Introducción a Componentes

Desarrollador Web con React Js

Componentes

Argentina programa 4.0

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

Conceptualmente, los componentes son como las funciones de JavaScript.

Aceptan entradas arbitrarias (llamadas “props”) y retornan elementos de React que describen lo que debe aparecer en la pantalla.

Los componentes pueden referirse a otros componentes en su retorno. Esto nos permite utilizar la misma abstracción de componente para cualquier nivel de detalle. Un botón, un cuadro de diálogo, un formulario, una pantalla: en las aplicaciones de React, todos ellos son expresados comúnmente como componentes.

Propiedades

Argentina programa 4.0

Un componente en React puede recibir propiedades como parámetros desde un componente padre para poder insertar valores y eventos en su HTML.

Imagina que tienes un componente que representa un menú con varias opciones, y éstas opciones las pasamos por parámetros como una propiedad llamada **options**:

```
import React from 'react'

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

  render () {
    let menuOptions = ['Opción 1', 'Opción 2', 'Opción 3']
    return <Menu options={menuOptions} />
  }
}
```

¿Cómo accedemos a estas propiedades en el componente hijo a la hora de renderizarlo? **Por medio de las props.**

Argentina programa 4.0

Veamos como con el código del componente **<Menu />**:

```
import React from 'react'

class Menu extends React.Component {
  constructor (props) {
    super(props)
  }

  render () {
    let options = this.props.options
    return (
      <ul>
       {options.map(option => <li>{option}</li>)}
      </ul>
    )
  }
}
```

En el método render creamos una variable **options** con el valor que tenga **this.props.options**. Éste **options** dentro de **props** es el mismo atributo options que tiene el componente **<Menu />** y es a través de props como le pasamos el valor desde el padre al componente hijo.

Argentina programa 4.0

```
import React from 'react'
import ReactDOM from 'react-dom'

class App extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {
    let menuOptions = ['Opción 1', 'Opción 2', 'Opción 3']
    return <Menu options={menuOptions} />
  }
}
```

```
class Menu extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {
    let options = this.props.options
    return (
      <ul>
        {options.map(option => <li>{option}</li>)}
      </ul>
    )
  }
}

ReactDOM.render(<App />, document.getElementById('app')
```

En el caso de ser un componente de tipo función las propiedades las recibimos como parámetro de la misma:

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
```

Estados

Argentina programa 4.0

Además de las props, los componentes en React pueden tener estado. Lo característico del estado es que si éste cambia, el componente se renderiza automáticamente.

Si tomamos el código anterior y en el componente <App /> guardamos en su estado las opciones de menú, el código de App sería así:

```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      menuOptions: ['Opción 1', 'Opción 2', 'Opción 3']
    }
  }

  render() {
    return <Menu options={this.state.menuOptions} />
  }
}
```

De ésta manera, las "opciones" pertenecen al estado de la aplicación (App) y se pueden pasar a componentes hijos (Menú) a través de las props.
Ahora, si queremos cambiar el estado, añadiendo una nueva opción al menú tenemos a nuestra disposición la función **setState**, que nos permite modificarlo.

Eventos

Argentina programa 4.0

Si las propiedades pasan de padres a hijos, es decir hacia abajo, los eventos se disparan hacia arriba, es decir de hijos a padres.

El componente <Menu /> va a tener una nueva propiedad llamada **onAddOption**: va a llamar a la función **handleAddOption** en <App /> para poder modificar el estado.

```
render() {
  return (
    <Menu
      options={this.state.menuOptions}
      onAddOption={this.handleAddOption.bind(this)}
    />
)
}
```

```
handleAddOption () {
  this.setState({
    menuOptions: this.state.menuOptions.concat(['Nueva Opción'])
  })
}
```

Cada vez que se llame a la función, añadirá al estado el ítem Nueva Opción.

Cada vez que se modifique el estado, se "re-renderizará" el componente y veremos en el navegador la nueva opción añadida.

Para poder llamar a esa función, necesitamos disparar un evento desde el hijo. Añadiremos un elemento <button> y utilizaremos el evento onClick de JSX, que simula al listener de click del ratón y ahí llamaré a la función "propiedad" onAddOption del padre.


```
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      menuOptions: ['Opción 1', 'Opción 2', 'Opción 3']
    }
  }

  render() {
    return (
      <Menu
        options={this.state.menuOptions}
        onAddOption={this.handleAddOption.bind(this)}
      />
    )
  }

  handleAddOption () {
    this.setState({
      menuOptions: this.state.menuOptions.concat(['Nueva Opción'])
    })
  }
}
```

Cada vez que hagamos click en el botón, llamará a la función que le llega por props.

Esta función, llama en el componente padre (App) a la función **handleAddOption** y la bindeamos con this, para que pueda llamar a **this.setState** dentro de la función.

Éste **setState** modifica el estado y llama internamente a la función render lo que provoca que se vuelva a "pintar" todo de nuevo.

Argentina programa 4.0

```
class Menu extends React.Component {
  constructor(props) {
    super(props)
  }

  render() {
    let options = this.props.options
    return (
      <div>
        <ul>
          {options.map(option => <li>{option}</li>)}
        </ul>
        <button onClick={this.props.onAddOption}>Nueva Opción</button>
      </div>
    )
  }
}

ReactDOM.render(<App />, document.getElementById('app'))
```

Equivalencias entre Hooks de efectos y Ciclos de Vida

Argentina programa 4.0

componentDidMount

```
useEffect(() => {
  /* ComponentDidMount code */
}, []);
```

componentDidUpdate

```
useEffect(() => {
  /* componentDidUpdate code */
}, [var1, var2]);
```

componentWillUnmount

```
useEffect(() => {
  return () => {
    /* componentWillUnmount code */
  }
}, []);
```

Gracias!