React.JS

Introducción

Módulo 01

Ciclo de vida

Ciclo de vida

Los ciclos de vida de los componentes se dividen en 3 grupos :

- **MONTADO**
- **ACTUALIZACIÓN**
- **DESMONTADO**

A su vez cada uno tiene varios métodos que lo subdividen.

Veamos cada uno de esos métodos.

Montado

La primera fase ocurre solo una vez por componente cuando este se crea y monta en la UI. Esta fase se divide en 4 funciones.

constructor(props)

Este método se ejecuta cuando se instancia un componente. Nos permite definir el estado inicial del componente, hacer bind de métodos y definir propiedades internas en las que podemos guardar muchos datos diferentes, por ejemplo la instancia de una clase (un parser, un validador, etc.).

componentWillMount()

Este método se ejecuta cuando el componente se está por renderizar. En este punto es posible modificar el estado del componente sin causar una actualización (y por lo tanto no renderizar dos veces el componente).

Es importante sin embargo evitar causar cualquier clase de efecto secundario (petición HTTP por ejemplo) ya que este método se ejecuta en el servidor y hacer esto puede causar problemas de memoria.

render()

En este momento de la fase de montado se van a tomar las propiedades, el estado y el contexto y se va a generar la UI inicial de este componente.

Esta función debe ser pura (no puede tener efectos secundarios) y no debe modificar nunca el estado del componente.

componentDidMount()

Este último método de la fase de montaje se ejecuta una vez el componente se renderiza en el navegador (este no se ejecuta al renderizar en el servidor) y nos permite interactuar con el DOM o las otras APIs del navegador (geolocation, navigator, notificaciones, etc.).

También es el mejor lugar para realizar peticiones HTTP o suscribirse a diferentes fuentes de datos (un Store o un WebSocket) y al recibir una respuesta, actualizar el estado. Cambiar el estado en este método causa que se vuelva a renderizar el componente.

Esa fase puede ocurrir múltiples veces (o incluso ninguna), sucede cuando algún dato del componente (ya sea una propiedad, un estado o el contexto) se modifica y por lo tanto requiere que la UI se vuelva a generar para representar ese cambio de datos.

componentWillReceiveProps(nextProps)

Este método se ejecuta inmediatamente después que el componente reciba nuevas propiedades. En este punto es posible actualizar el estado para que refleje el cambio de propiedades, ya sea reiniciando su valor inicial o cambiándolo por uno nuevo. Hay que tener en cuenta que React puede llegar a ejecutar este método incluso si las propiedades no cambiaron. Por eso, es importante validar que las nuevas propiedades (*nextProps*) sean diferentes de las anteriores (*this.props)*.

Actualización

shouldComponentUpdate(nextProps, nextState)

Este método (el cual debe ser puro) se ejecuta antes de empezar a actualizar un componente, cuando llegan las nuevas propiedades (*nextProps*) y el nuevo estado (*nextState*).

Acá es posible validar que estos datos sean diferentes de los anteriores (*this.props* y *this.state*) y devolver true o false dependiendo de si queremos volver a renderizar o no el componente.

componentWillUpdate(nextProps, nextState)

Una vez el método anterior devuelve true se ejecuta este método, acá es posible realizar cualquier tipo de preparación antes de que se actualice la UI.

Es importante tener en cuenta que acá no se puede ejecutar *this.setState* para actualizar el estado. Si queremos actualizar el estado con base a un cambio de propiedades debemos hacerlo en *componentWillReceiveProps.*

render()

Al igual que en el montado acá se va a generar la UI, esta vez con los datos que hayan cambiado. Como antes este método debe ser puro.

componentDidUpdate(prevProps, prevState)

Esta última parte de la actualización de un componente ocurre justo después de que se renderiza en el DOM nuestro componente. Al igual que con *componentDidMount()* acá es posible interactuar con el DOM y cualquier API de los navegadores.

Desmontado

Esta última fase consiste en un solo método que se ejecuta antes de que un componente se elimine (desmonte) de la UI de nuestra aplicación.

componentWillUnmount()
Este único método de la fase de desmontado nos permite realizar cualquier tipo de limpieza antes de remover el componente.

Acá es posible dejar de escuchar eventos de window, document o el DOM, desuscribirse de un WebSocket o Store o cancelar peticiones HTTP que hayan quedado pendientes.

¡Muchas gracias!

¡Sigamos trabajando!