DigitalHouse>

Certified Tech Developer
The Ultimate Degree

Front End III

Glosario de repaso

1. Ciclo de vida: React, en sus principios de diseño, manifiesta que el armado de aplicaciones mediante componentes es su característica clave. Los componentes tienen un ciclo de vida, es decir, pasan por varios estadíos. Un componente se instancia cuando es montado en el DOM (modelo de objetos del documento). Del mismo modo, podemos entender que su instancia puede eliminarse al ser desmontada del DOM.

2. Fases del ciclo de vida: montaje, actualización y destrucción o desmontaje.

3. Métodos en el ciclo de vida en componentes de clase: los componentes de clase proporcionan una serie de métodos. Habrá métodos específicos para cada estadío del componente.

4. Constructor: este es llamado antes de montarse el componente. Si bien no es obligatorio, es usado para iniciar el estado y enlazar manejadores de eventos a una instancia (en la próxima clase hablaremos de estos eventos). Al usar el constructor en una clase que extiende React.Component deberá llamarse a super(props), de este modo, this.props quedará definido en el constructor.

5. Render: es el único método obligatorio en un componente de clase. Se trata de una función pura, no interviene sobre el estado ni interactúa con el navegador, solo representará lo que reciba. Al ser llamado, examinará this.props y this.state e inmediatamente retornará alguna de las siguientes opciones:

- Elementos de React creados con JSX
- Arrays
- Fragments
- Portals (lo veremos más adelante)
- String, numbers, booleans o null

6. componentDidMount(): se invoca una vez que el componente se montó en el DOM (aparece en el HTML) y no vuelve a ejecutarse. Es utilizado para realizar suscripciones. También es posible llamar a setState antes de que la pantalla se actualice.

7. componentDidUpdate(): es invocado ante cada actualización de state o props que suceda luego del renderizado inicial. Permite comparar valores previos con actuales y tomar decisiones sobre la base de condiciones.

8. componentWillUnmount(): es el único método que se llama cuando un componente está por desmontarse. Si bien React desmonta el componente por sí solo, puede desconocer especificidades del componente, por lo cual no sabrá exactamente cómo limpiar el componente removido. Esa es la utilidad de este método. La limpieza es útil cuando el componente ha realizado llamadas asincrónicas. El problema básico a abordar es si el componente realizó una llamada a una API asincrónicamente y antes de recibir la respuesta es desmontado. Si bien, no pasará nada grave, se registrará una advertencia y no se establecerá el estado. Es útil

registrar esta advertencia para poder rastrear errores. Las acciones asincrónicas deben ser cancelables.

9. static getDerivedStateFromProps(props, state): este método es invocado justo antes de render(). Devolverá un objeto en el caso de tener que actualizar el estado o null en caso contrario. Se usa en casos raros en los cuales se le permite a un componente actualizar su estado interno como resultado de cambios en las props. Es un método estático porque se espera que se comporte como una función pura y no tenga efectos secundarios.

10. shouldComponentUpdate(nextProps, nextState): compara nuevos estados o props con los previos. Entonces puede devolver false si no es necesario volver a renderizar el componente. Este método se utiliza para hacer que los componentes sean más eficientes.

11. getSnapshotBeforeUpdate(prevProps, prevState): facilita hacer operaciones sobre los elementos del DOM antes de que el componente sea realmente comprometido con el DOM. Por ejemplo, la posición exacta del scroll en un momento determinado. La diferencia entre este método y render() es que getSnapshotBeforeUpdate() no es asincrónico. Con render() puede pasar que la estructura DOM cambie entre el momento en que se llama y cuándo se realizan los cambios en el DOM. Cualquier valor que se devuelva en este método de ciclo de vida se pasará como parámetro a componentDidUpdate().

12. Eventos en React: permiten a los usuarios interactuar con la capa visual y de datos de la aplicación. ¿A qué llamamos eventos de usuario? Por ejemplo, cuando hacemos clic en un botón, modificamos un input (change) o hacemos hover sobre una imagen, entre una gran cantidad de opciones. Se declaran en las etiquetas JSX en el render del componente.

13. Manejador de eventos: es una función o método que ejecutará las instrucciones declaradas en su interior una vez que el evento haya sido disparado.

14. SyntheticEvent: haciendo una aproximación ligera al tema, al instanciar un manejador de eventos (al disparar un evento), se le pasará a este una instancia de SyntheticEvent. ¿Para qué sirve esto? Simplificando, podríamos decir que provee un contenedor al que no le importará en lo más mínimo qué navegador se esté usando, por lo que permitirá que los eventos funcionen de igual modo en todos los navegadores.

15. Componentes controlados: los componentes controlados son controlados por React y los no controlados son controlados por el DOM.

16. ¿Qué queremos decir con controlar?: llevar un registro de qué pasa a cada momento en un input, sus valores, sus actualizaciones. Estamos controlando los datos, asegurando desde el comienzo la integridad de los mismos antes de, por ejemplo, guardarlos en una base de datos.

17. Formik: es una biblioteca de código abierto que nos permite trabajar en React con formularios. Se encarga de simplificar el seguimiento de valores / errores / campos visitados, manejar las validaciones y submits. Simplifica radicalmente la configuración de estados y controladores y facilita la depuración, las pruebas y el razonamiento sobre sus formularios.

18. SweetAlert: generador de modales.

¡Hasta la próxima!