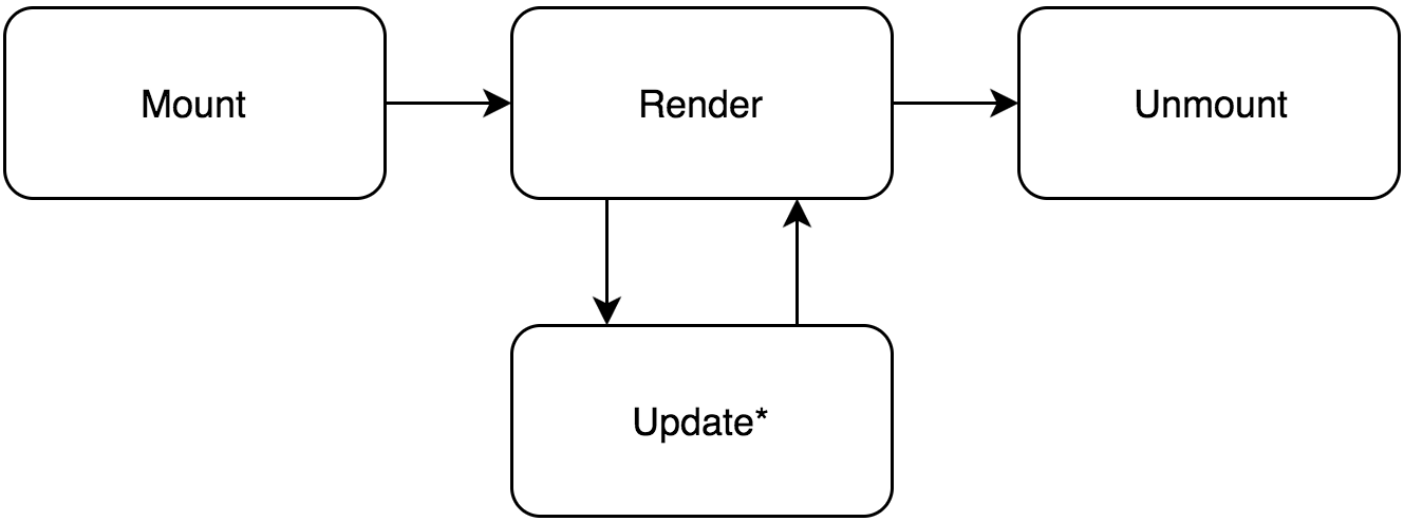
Desde que una persona entra en nuestra aplicación hasta que la abandona, pueden ocurrir gran cantidad de eventos: cambiar de vista, hacer click en un botón, volver atrás… En React, las aplicaciones son del tipo SPA (*Single Page Applications*), **por lo que los componentes son *montados*, mostrados, actualizados y *desmontados***. Todos estos estados (en este caso no me refiero al `state` del componente) por los que pasa un componente pertenecen a su **ciclo de vida o *lifecycle***. Nosotros podemos *escuchar* dichos eventos y realizar acciones cuando ocurren.

Podemos resumir el ciclo de vida en este diagrama:



Mount

Cuando requerimos un componente que no está en la interfaz, este se monta en nuestra aplicación. **En esta fase, el componente es iniciado y mostrado por primera vez**. Los métodos que se ejecutan son:

1. `defaultProps` : se inicializan los props combinando los valores disponibles con los que hemos definido por defecto.

2. `constructor(props, context)` : instanciamos el componente e iniciamos su estado. Para inicializar el estado, almacenamos el estado inicial en `this.state` . Este método recibe las props y el contexto de la aplicación. En la sección de *Rutas en React*entraremos en detalle en el contexto.

3. `componenteWillMount()` : este método es llamado antes de renderizar el componente. También podemos inicializar `this.state` en este punto. No obstante, este se suele utilizar cuando se renderizan los componentes en servidor y no en cliente, por lo que no nos será útil.

4. `render()` : renderizamos la vista con los valores por defecto.

5. `componentDidMount()` : el componente ya ha sido montado y está listo para ser usado. Este método es ideal para realizar peticiones al servidor ya que podemos mostrar un mensaje de cargando por defecto. Al llamar a `setState` en este método provocamos que se vuelva a renderizar el componente.

Código en Codepen.

Update

> Un componente sólo se actualiza cuando cambian sus props o su state.

Los dos eventos que pueden desencadenar el ciclo de actualización de un componente son:

- El cambio de los valores de los props
- Una llamada a `setState`

Ambos ciclos ejecutan los mismo métodos salvo `componentWillReceiveProps` , que solo es llamado en cuando cambia el valor de algún prop.

1. `componenteWillReceiveProps(nextProps)` : recibe como parámetros los nuevos props del componente. En este momento tenemos disponibles los props antiguos en `this.props` . Se suele utilizar para actualizar el estado o para volver al valor inicial alguna variable del estado.

2. `shouldComponenteUpdate(nextProps, nextState)` : este método es muy importante al hablar de **rendimiento en React**. Este método nos permite comparar los nuevos props y state con los valores actuales almacenados en `this.props` y `this.state` . Si este método retorna `false` , los siguientes métodos del ciclo no se ejecutarán y el método `render` tampoco será llamado. No obstante, hay que tener cuidado con este método pues puede provocar comportamientos inesperados. **Generalmente, no sobrescribiremos esté método puesto y dejaremos que React se encargue de decidir si debe de volver a renderizar el componente**.

3. `componenteWillUpdate(nextProps, nextState)` : similar a `componenteWillMount` . Este método se llama antes de volver a renderizar el componente. No podemos llamar a `setState` en este punto.

4. `render()` : renderizamos la vista **con los nuevos valores**.

5. `componentDidUpdate(prevProps, prevState)` : similar a `componenteDidMount` . En este caso recibimos los anteriores props y state, ya que los nuevos está disponibles como atributos de `this` . Este también es un buen momento para realizar peticiones al servidor basándonos en los nuevos valores del componente.

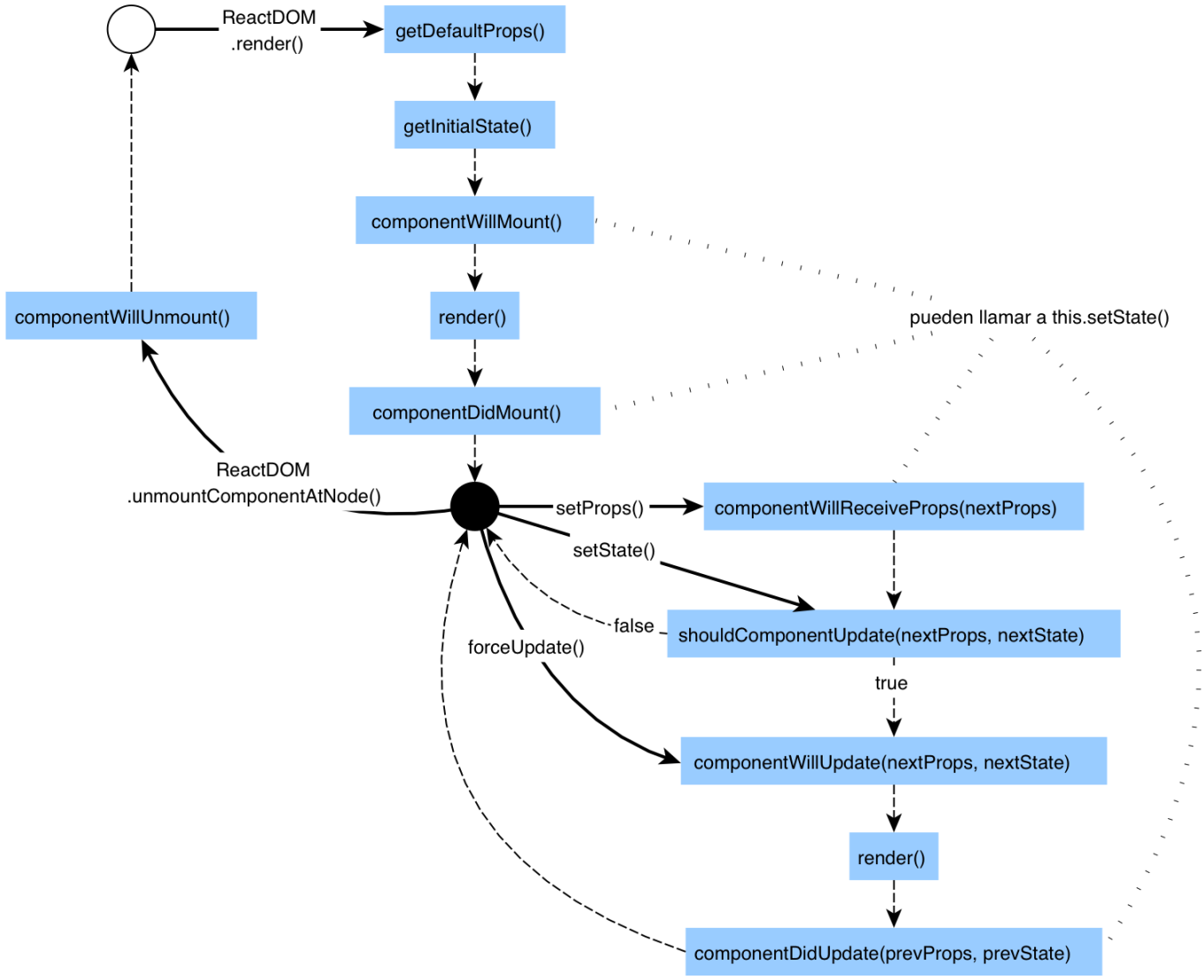
Unmount

Esta es la fase final del ciclo de vida de un componente. Cuando un componente ya no es necesario en la interfaz React le notifica y procede a desmontarlo y eliminarlo.

1. `componenteWillUnmount()` : este método es llamado justo antes de que un componente se desmonte y se elimine. Es muy útil para liberar recursos como eliminar elementos de la interfaz, cancelar peticiones al servidor o invalidar temporizadores.

Diagrama

A modo de resumen, os dejo un diagrama con todos los métodos del ciclo de vida de un componente. Este diagrama fue creado por el usuario de reddit flying-sheep.


ReactDOM
.render()
getDefaultProps()
getInitialState()
componentWillMount()
render()
componentDidMount()
componentWillUnmount()
ReactDOM
.unmountComponentAtNode()
pueden llamar a this.setState()
setProps()
componentWillReceiveProps(nextProps)
setState()
false
shouldComponentUpdate(nextProps, nextState)
forceUpdate()
true
componentWillUpdate(nextProps, nextState)
render()
componentDidUpdate(prevProps, prevState)