9 - Componentes: ciclo de vida

Cuando se crea una componente ocurren una serie de pasos de inicialización. Durante este ciclo React le da la oportunidad al desarrollador de especificar código propio.

Los dos métodos más importantes que podemos sobreescribir a una componente son:

componentWillMount() //se ejecuta antes que el DOM de la componente se haya montado

componentDidMount() //se ejecuta luego que el DOM de la componente se ha generado (ya se ha ejecutado el método render())

componentWillUnmount() //se ejecuta antes que el DOM de la componente se destruya

Problema

Crear una componente llamada Reloj que se actualice cada segundo en pantalla.

Primero debemos crear el **proyecto008** desde la línea de comandos de Node.js con la aplicación create-react-app

La componente Reloj.js:

import React, {Component} from 'react'

class Reloj extends Component {

constructor(props) {

super(props);

this.state = ({

hora: new Date()

})

}

render() {

return (

<div>{this.state.hora.toLocaleTimeString()}</div>

);

}

componentDidMount() {

this.id = setInterval(()=>this.cambioSegundo(), 1000);

}

componentWillUnmount() {

clearInterval(this.id);

}

cambioSegundo() {

this.setState ({

hora: new Date()

})

}

}

export default Reloj;

En el constructor fijamos el 'estado' de la componente almacenando un objeto de la clase 'Date':

constructor(props) {

super(props);

this.state = ({

hora: new Date()

})

}

Luego según el ciclo de vida de una componente se ejecuta el método render() donde se muestra la hora actual obteniendo el dato del 'estado' de la componente:

render() {

return (

<div>{this.state.hora.toLocaleTimeString()}</div>

);

}

Por única vez luego de graficarse la componente se ejecuta el método componentDidMount() donde creamos un temporizador para que se ejecute una función anónima que le pasamos en el primer parámetro cada 1000 milisegundo:

componentDidMount() {

this.id = setInterval(()=>this.cambioSegundo(), 1000);

}

Guardamos en un atributo de la componente llamado 'id' la referencia del temporizador para poder liberarlo previo a que se destruya la componente.

El método cambiarSegundo() se ejecuta cada un segundo llamado por el temporizador. En su algoritmo modificamos el 'estado' de la componente lo que provoca que se actualice la componente con los cambios respectivos:

cambioSegundo() {

this.setState ({

hora: new Date()

})

}

Finalmente previo a destruirse la componente liberamos los recursos del temporizador al ejecutarse el método componentWillUnmount():

componentWillUnmount() {

clearInterval(this.id);

}

Ahora debemos codificar la componente 'App' donde consumimos la componente 'Reloj'.

App.js

import React, { Component } from 'react';

import './App.css';

import Reloj from './Reloj'

class App extends Component {

render() {

return (

<div>

<Reloj />

</div>

);

}

}

export default App;

Recordemos que siempre debemos importar las componentes que consumimos:

import Reloj from './Reloj'

En el método render() definimos una componente de tipo 'Reloj':

render() {

return (

<div>

<Reloj />

</div>

);

}

[Retornar](https://www.tutorialesprogramacionya.com/reactya/index.php?inicio=0)