5 - Estado de la componente

Hay un atributo fundamental que puede tener toda componente en React donde almacenamos valores que luego en forma dinámica se actualizan en el método render() mediante expresiones.

La propiedad state se debe inicializar en el constructor, por ejemplo:

this.state = {

numero: 0

}

Podemos definir en state todos los datos que van a variar durante la ejecución de la aplicación y queremos que se actualicen en pantalla.

Problema

Crear un nuevo proyecto llamado: proyecto004.  
Definir en la interfaz visual un botón que cada vez que se presione se actualice en pantalla un número aleatorio entre 0 y 9.

import React, { Component } from 'react';

import './App.css';

class App extends Component {

constructor(props) {

super(props)

this.generarAleatorio = this.generarAleatorio.bind(this);

this.state = {

numero: 0

}

}

render() {

return (

<div>

<p>Número aleatorio: {this.state.numero}</p>

<button onClick={this.generarAleatorio}>Generar número aleatorio</button>

</div>

);

}

generarAleatorio() {

const v=Math.trunc(Math.random()\*10);

this.setState( {

numero: v

})

}

}

export default App;

Definimos en el constructor el valor del atributo state (debe llamarse así ya que en la clase padre Component lo define):

constructor(props) {

super(props)

this.generarAleatorio = this.generarAleatorio.bind(this);

this.state = {

numero: 0

}

}

En el método render() lo mostramos mediante una expresión:

render() {

return (

<div>

<p>Número aleatorio: {this.state.numero}</p>

<button onClick={this.generarAleatorio}>Generar número aleatorio</button>

</div>

);

}

Es decir que cuando se inicia la aplicación aparece el número cero. Podemos inicializarlo con cualquier valor, por ejemplo:

this.state = {

numero: 'No se ha generado número aleatorio aún'

}

Luego cuando se presiona el botón se dispara el evento generarAleatorio:

generarAleatorio() {

const v=Math.trunc(Math.random()\*10);

this.setState( {

numero: v

})

}

En este método cambiamos el valor del atributo 'numero' pero no le asignamos el valor directamente:

this.state.numero=v;

Sino que mediante la llamada al método setState (que se hereda de la clase Component) modificamos su valor, con esto la librería React se encarga de ejecutar nuevamente el método render() pero solo actualizando los estados cambiados y sin tener que redibujar la página completa.

Problema

Modificar el problema anterior para que se muestren 5 valores aleatorios. Almacenar los 5 valores en this.state en un vector.

import React, { Component } from 'react';

import './App.css';

class App extends Component {

constructor(props) {

super(props)

this.generarAleatorios = this.generarAleatorios.bind(this);

this.state = {

numeros: []

}

}

render() {

return (

<div>

<p>Números aleatorios:</p>

{this.state.numeros.map(function(num)

{ return (<p>{num}</p>);

}

)}

<button onClick={this.generarAleatorios}>Generar números aleatorios</button>

</div>

);

}

generarAleatorios() {

const vec=new Array(5)

for(let x=0; x<vec.length; x++)

vec[x]=Math.trunc(Math.random()\*10);

this.setState( {

numeros: vec

})

}

}

export default App;

Almacenamos en la propiedad 'numeros' del estado de la componente un vector vacío:

this.state = {

numeros: []

}

En el método generarAleatorios() procedemos a crear un vector de 5 elementos y guardar los 5 valores aleatorios. Seguidamente llamamos al método setState para actualizar la propiedad numeros con la referencia del vector:

generarAleatorios() {

const vec=new Array(5)

for(let x=0; x<vec.length; x++)

vec[x]=Math.trunc(Math.random()\*10);

this.setState( {

numeros: vec

})

}

Para mostrar todos los elemento del vector llamamos al método map y le pasamos una función anónima que retorna para cada elemento del vector un valor encerrado entre las marcas 'p':

{this.state.numeros.map(function(num)

{ return (<p>{num}</p>);

}

)}

En el Javascript moderno es más común utilizar una función flecha (Arrow function):

{this.state.numeros.map((num) => { return (<p>{num}</p>) } )}

Problema

Crear un nuevo proyecto con la herramienta create-react-app llamado proyecto005.  
Almacenar en el estado de la componente el siguiente vector:

articulos: [{

codigo: 1,

descripcion: 'papas',

precio: 12.52

},{

codigo: 2,

descripcion: 'naranjas',

precio: 21

},{

codigo: 3,

descripcion: 'peras',

precio: 18.20

}]

Mostrar en una tabla HTML dichos datos. Cuando se presione un botón borrar el último elemento de la tabla.

import React, { Component } from 'react';

import './App.css';

class App extends Component {

constructor(props) {

super(props)

this.state = {

articulos: [{

codigo: 1,

descripcion: 'papas',

precio: 12.52

},{

codigo: 2,

descripcion: 'naranjas',

precio: 21

},{

codigo: 3,

descripcion: 'peras',

precio: 18.20

}]

}

this.eliminarUltimaFila = this.eliminarUltimaFila.bind(this);

}

render() {

return (

<div>

<table border="1">

<thead><tr><th>Código</th><th>Descripción</th><th>Precio</th></tr></thead>

<tbody>

{this.state.articulos.map(elemento => {

return (

<tr key={elemento.codigo}>

<td>

{elemento.codigo}

</td>

<td>

{elemento.descripcion}

</td>

<td>

{elemento.precio}

</td>

</tr>

)

})}

</tbody>

</table>

<button onClick={this.eliminarUltimaFila}>Eliminar última fila</button>

</div>

);

}

eliminarUltimaFila() {

if (this.state.articulos.length > 0) {

var temp = this.state.articulos;

temp.pop();

this.setState({

articulos: temp

})

}

}

}

export default App;

En el constructor inicializamos la propiedad this.state con la referencia al vector.

Para mostrar los datos del vector nuevamente empleamos la llamada al método 'map' y le pasamos una función anónima. Dentro de la función anónima generamos cada fila de la tabla con los datos de un producto:

<table border="1">

<thead><tr><th>Código</th><th>Descripción</th><th>Precio</th></tr></thead>

<tbody>

{this.state.articulos.map(elemento => {

return (

<tr key={elemento.codigo}>

<td>

{elemento.codigo}

</td>

<td>

{elemento.descripcion}

</td>

<td>

{elemento.precio}

</td>

</tr>

)

})}

</tbody>

</table>

Como podemos comprobar la fila de títulos de la tabla la hacemos previo a llamar a map.

Cuando se presiona el botón llamamos al método 'eliminarUltimaFila. En este método primero comprobamos si el vector tiene elementos accediendo al atributo 'length', en caso afirmativo procedemos a guardar la referencia del vector en una variable temporal, luego eliminamos el último elemento llamando al método pop y finalmente actualizamos el atributo 'this.state' con el nuevo vector:

eliminarUltimaFila() {

if (this.state.articulos.length > 0) {

var temp = this.state.articulos;

temp.pop();

this.setState({

articulos: temp

})

}

}

Problema

Modificar el ejercicio anterior para que aparezca un botón en cada fila de la tabla y permita borrar dicho artículo.

import React, { Component } from 'react';

import './App.css';

class App extends Component {

constructor(props) {

super(props)

this.state = {

articulos: [{

codigo: 1,

descripcion: 'papas',

precio: 12.52

},{

codigo: 2,

descripcion: 'naranjas',

precio: 21

},{

codigo: 3,

descripcion: 'peras',

precio: 18.20

}]

}

this.borrar = this.borrar.bind(this);

}

render() {

return (

<div>

<table border="1">

<thead><tr><th>Código</th><th>Descripción</th><th>Precio</th><th>Borra?</th></tr></thead>

<tbody>

{this.state.articulos.map(elemento => {

return (

<tr key={elemento.codigo}>

<td>

{elemento.codigo}

</td>

<td>

{elemento.descripcion}

</td>

<td>

{elemento.precio}

</td>

<td>

<button onClick={()=>this.borrar(elemento.codigo)}>Borrar</button>

</td>

</tr>

)

})}

</tbody>

</table>

</div>

);

}

borrar(cod) {

var temp = this.state.articulos.filter((el)=>el.codigo !== cod);

this.setState({

articulos: temp

})

}

}

export default App;

Cuando generamos ahora la tabla en la última celda debemos disponer un botón que llame a un método y le pase como parámetro el código de artículo a borrar:

<button onClick={()=>this.borrar(elemento.codigo)}>Borrar</button>

Cuando llamamos a un método debemos plantear una función anónima que se le asigna al evento 'onClick'.

El método 'borrar' recibe como parámetro el codigo de artículo a borrar:

borrar(cod) {

var temp = this.state.articulos.filter((el)=>el.codigo !== cod);

this.setState({

articulos: temp

})

}

Para borrar un determinado elemento del vector utilizamos el método filter que genera otro vector con todas las componentes que cumplen la condición que le pasamos en la función anónima.

Finalmente actualizamos el estado para que se redibuje la página.

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