3 - Formato JSX

Creamos en el concepto anterior la primer aplicación en React utilizando la aplicación create-react-app y vimos que se genera un archivo llamado App.js:

import React, { Component } from 'react';

import './App.css';

class App extends Component {

render() {

return (

<h1>Hola Mundo</h1>

);

}

}

export default App;

Analizaremos ahora cual es el objetivo del método render() y que formato tiene.

Problema

Crear el segundo proyecto escribiendo desde la línea de comandos de Node.js:

create-react-app proyecto002

El método render() no es XHTML o Javascript puro, es un nuevo formato propuesto por los creadores de React que luego de ser compilado se genera Javascript puro que lo pueden entender los navegadores.

Hay ciertas reglas que debe cumplir el JSX (**J**ava**S**cript **X**ML), presentaremos algunas de ellas en este segundo ejercicio.

Modifiquemos el método render() de la clase App:

import React, { Component } from 'react';

import logo from './logo.svg';

import './App.css';

class App extends Component {

render() {

const siglo=21;

const persona = {

nombre: 'Juan',

edad: 34

}

return (

<div>

<h1>Título nivel 1</h1>

<hr />

<p>Estamos en el siglo {siglo}</p>

<h3>Acceso a un objeto</h3>

<p>{persona.nombre} tiene {persona.edad} años</p>

<h3>Llamada a un método</h3>

<p>Un valor aleatorio llamando a un método.</p>

{this.retornarAleatorio()}

<h3>Calculo inmediato de expresiones</h3>

3 + 3 = {3+3}

</div>

);

}

retornarAleatorio() {

return Math.trunc(Math.random() \* 10);

}

}

export default App;

El método render() tiene por objetivo generar la interfaz visual de la componente 'App' (por el momento desarrollaremos toda nuestra aplicación en una única componente, luego veremos que un programa se descompone un muchas componentes.

El método render() mediante un return devuelve el JSX que se deberá mostrar en el navegador. Como vemos dentro del bloque de JSX podemos disponer etiquetas HTML tal como conocemos:

<h1>Título nivel 1</h1>

Una restricción de JSX es que siempre los elementos HTML deben tener su marca de comienzo y fin, y en el caso que solo tengan una etiqueta que es tanto de comienzo como fin debemos agregar el caracter '/':

<hr />

Si nos olvidamos de agregar la barra de cierre se genera un error cuando tratamos de compilar la aplicación:



Dentro del bloque JSX podemos acceder a variables o constantes indicando entre llaves dicha variable o constante:

<p>Estamos en el siglo {siglo}</p>

Luego cuando se compila en lugar de la expresión se muestra el contenido de la variable o constante:



En forma similar podemos disponer expresiones para acceder a propiedades de un objeto previamente definido:

<p>{persona.nombre} tiene {persona.edad} años</p>

Otra posibilidad en una expresión es hacer la llamada a métodos definidos en la misma clase (recordar de anteceder la palabra this):

{this.retornarAleatorio()}

Podemos inclusive disponer una operación que será ejecutada previo a su visualización:

3 + 3 = {3+3}

Otras restricciones que tiene el formato JSX

Se recomienda dividir el JSX en varias líneas para facilitar la lectura, también recomiendan envolverlo entre paréntesis para evitar los inconvenientes de la inserción automática de punto y coma.

Para definir valores a las propiedades de un elementos HTML mediante expresiones no debemos disponer las comillas. Por ejemplo modifique el método render() y pruebe esto:

render() {

const buscadores=['http://www.google.com',

'http://www.bing.com',

'http://www.yahoo.com'];

return (

<div>

<a href={buscadores[0]}>Google</a><br />

<a href={buscadores[1]}>Bing</a><br />

<a href={buscadores[2]}>Yahoo</a><br />

</div>

);

}

No deben ir las comillas en la asignación de la propiedad href si el valor se extrae de una expresión:

<a href={buscadores[0]}>Google</a><br />

Si queremos darle el valor directamente si se requieren las comillas:

<a href="http://www.google.com">Google</a><br />

Podemos plantear métodos en la clase que retornen trozos de JSX que luego se agregan al que retorna el método render(), probar de modificar nuevamente el problema con:

import React, { Component } from 'react';

import './App.css';

class App extends Component {

render() {

return (

<div>

{this.mostrarTitulo('Hola Mundo')}

{this.mostrarTitulo('Fin')}

</div>

);

}

mostrarTitulo(tit) {

return (<h1>

{tit}

</h1>)

}

}

export default App;

Una restricción del método render es que siempre debe retornar un elementos HTML que puede tener en su interior otros elementos anidados, pero nunca dos elementos HTML hermanos, esto genera un error:

render() {

return (

<div>

<h1>Titulo 1</h1>

</div>

<div>

<h1>Titulo 2</h1>

</div>

);

}

Debemos disponer obligatoriamente solo un elemento div que envuelva todo:

render() {

return (

<div>

<div>

<h1>Titulo 1</h1>

</div>

<div>

<h1>Titulo 2</h1>

</div>

</div>

);

}

Se puede utilizar otros elementos HTML para envolver todo el JSX como por ejemplo 'span', 'section' o cualquier otro, siempre y cuando no haya elementos hermanos en la raíz.

Importante

Otra cosa muy importante que hay que tener en cuenta que los nombres de las propiedades de los elementos HTML cambian en varias situaciones:

Debemos utilizar la palabra 'className' en lugar de class (debido a que class es una palabra clave de Javascript):

<h1 className="recuadro">Titulo 1</h1>

Si el nombre de la propiedad está formada por más de una palabra luego el primer caracter a partir de la segunda palabra debe ir en mayúsculas:

<input type="text" tabIndex="1" />

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