**React.js Básico**

Yojhan Leonardo Rodríguez Ascencio

*Github @ylrodriguez*

Contenido

[Capítulo 1. ¿Qué es React? 2](#_Toc3990605)

[Capítulo 2. Comenzando con React 3](#_Toc3990606)

[Capítulo 3. Crear un proyecto en React 6](#_Toc3990607)

[Capítulo 4. Props y State 8](#_Toc3990608)

# Capítulo 1. ¿Qué es React?

React es una **librería[[1]](#footnote-1)** de JavaScript basada en componentes que se usa para construir interfaces de usuario. Es un proyecto de código abierto que empezó en Facebook y que es mantenido por desarrolladores de esta misma organización y por una gran comunidad de contribuidores. Lanzado en el 2013.

* Es una librería de *rendering*.
* Usado para actualizar componentes, áreas específicas de tu sitio web, etc.
* Usa un DOM virtual que **quizá** lo hace más rápido. Un DOM virtual es una representación en JS del DOM real. El DOM real se actualiza cuando la librería cree que es un re-render se necesita.
* React hace el desarrollo de SPA (Single Page Applications) de gran escala mucho más fáciles.

React 16.x

* Rendering asíncrono
* Retornar arrays de elementos

Configuraciones

* React developer tools

# Capítulo 2. Comenzando con React

JSX

JSX es una extensión de la sintaxis de JavaScript. Permite usar etiquetas directamente en un archivo de JavaScript.

Componentes

Los componentes dejan dividir la UI en piezas independientes y reusables y pensar separadamente en cada pieza.

**Documentación React:**

<https://reactjs.org/docs/react-component.html>

<https://reactjs.org/docs/components-and-props.html>

React deja crear componentes como clases o como funciones. Los componentes definidos como clases son más robustos dando más características:

**Componente como clase:**

class Welcome extends React.Component {

render() {

return <h1>Hello, {this.props.name}</h1>;

}

}

**Componente como función:**

function Welcome(*props*) {

return <h1>Hello, {props.name}</h1>;

}

Usando Arrow Functions:[[2]](#footnote-2)

const Welcome = (*props*) => {

      return(

        <h1> Welcome to React </h1>

      )

    }

Usando props (properties)

Este atributo es la forma en como los componentes se hablan los unos a los otros. Los props fluyen hacia abajo desde el componente padre o principal. En este caso, en el componente de *Hello* indicamos dentro de las etiquetas **{ \_ }** *(Esa es la forma de usar js dentro del html en JSX)* y le asignamos el valor desde donde llamemos el componente.

class Hello extends React.Component{

      render(){

        return(

            <div>

              <h1>Hola {this.props.firstName}</h1>

            </div>

        )

      }

    }

    ReactDOM.render(

      <Hello *firstName*="Eve"/>,

      document.getElementById('react-container')

    )

También podemos usar una función arroy y de esta manera tendríamos que enviar *props* a la función y no es necesario usar el *this.*

const HelloStateless = (*props*) => {

      return(

        <div>

              <h1>Hola {props.firstName}</h1>

        </div>

      )

    }

Usando destructuración en React

Para explicación profunda de esta característica de ES6:

<https://codeburst.io/es6-destructuring-the-complete-guide-7f842d08b98f>

En vez de usar todos los elementos que tenga *props,* podemos llamar uno en específico usando destructuración:

const HelloStateless = ({*firstName*}) => {

      return(

        <div>

              <h1>Hola {firstName}</h1>

        </div>

      )

    }

Usando State

Cuando el estado de los datos de un componente cambia, la función render volverá a ser llamada de nuevo.

Aunque ***state*** *y* ***props***son ambos objetos de JavaScript que influencian el output del render, son diferentes en una cosa importante:

***Props***son pasados al componente (similar a los parámetros de una función) mientras tanto ***State*** es administrado dentro del componente (similar a las variables declaradas dentro de una función). Los siguientes son artículos interesantes donde explican esta diferencia:

<https://github.com/uberVU/react-guide/blob/master/props-vs-state.md>

# Capítulo 3. Crear un proyecto en React

Documentación oficial para crear proyecto:

<https://reactjs.org/docs/create-a-new-react-app.html>

Primero se debe tener instalado *Node.js* y por defecto *npm*. Para verificar esto, solo basta ejecutar los siguientes comandos:

$ npm -v

6.4.1

$ node -v

v10.15.3

Ahora una vez validado que se tenga las herramientas apropiadas, solo basta ejecutar el siguiente comando:

**npx create-react-app mi-primer-react**

**----------**

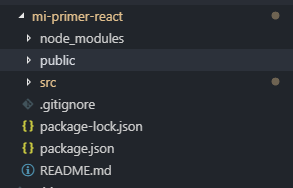
**cd mi-primer-react**

**npm start**

Estructura de archivos

<https://www.c-sharpcorner.com/article/folder-structure-of-react-application/>

Una vez creado el proyecto, obtenemos una carpeta parecida a la siguiente:



**Node\_modules:**

Todas las dependencias y sub dependencias para nuestro proyecto. Al momento de crear un proyecto de React, solo se tiene React, React DOM y React Scripts, pero este último tiene un montón de dependencias que se encuentran aquí. Esta carpeta, no debería subirse a GitHub o compartirse.

**Public:**

Tiene 3 archivos:

* Favicon.ico: Es el archivo del icono el cual contiene un icono que se muestra en el navegador.
* Index.html: Es como la plantilla de la página. Es un archivo importante. Gracias a este archivo, la carpeta *public* conocida como la carpeta *root*, mantiene contacto con el servidor.
* Manifest.json: Metadata de nuestra aplicación como, nombre, iconos, urls, etc.

**Src folder:**

La carpeta donde se aloja el código fuente de los desarrolladores. El lugar donde la aplicación de react es presentada. Los archivos alojados dentro de esta carpeta son procesados por Webpack.

Un archivo que destaca es *index.js* el cual guarda nuestra llamada de render principal desde el ReactDOM.

**Package.json**

Es un estándar en todos los proyectos. Contiene información como el nombre del proyecto, versiones, dependencias, etc. Cuando se instala una librería externa. Automáticamente se registra en este archivo.

Las tres dependencias por defecto son *React, react-dom y react scripts.*

**Un punto para recordar**, es que para compilar un proyecto de React, estos dos archivos deben existir con los mismos nombres:

* *Public.index.html* el cual es el template
* *Src/index.js* el cual es el punto de entrada de JavaScript

# Capítulo 4. Props y State

En el *capítulo dos* se da una pequeña introducción de estas dos propiedades de React.

1. Una librería es una serie de funciones que tienen un propósito y suelen ser añadidas a nuestro código. [↑](#footnote-ref-1)
2. // ES5

   var multiplyES5 = function(x, y) {

   return x \* y;

   };

   **// ES6**

   **const multiplyES6 = (x, y) => { return x \* y };** [↑](#footnote-ref-2)