¿Qué es React?

"Una biblioteca de JavaScript para construir interfaces de usuario" - https://es.reactjs.org/

React será el código que se ejecuta en el navegador de nuestro usuario, mostrando la información que queremos que vean, reaccionando a la información que el usuario ingrese, siendo responsable no solo de la apariencia de nuestra aplicación web, sino también de su interactividad.

¿De dónde viene React?

React es un proyecto de código abierto mantenido por Facebook, lanzado bajo la licencia permisiva del MIT. Junto con Angular y Vue, es uno de los frameworks front-end más populares que se utilizan en la actualidad.

¿Por qué debemos aprender React?

React se puede usar para crear una aplicación de página única (Single Page Application) o SPA, lo que hará que nuestro sitio web se sienta más responsivo al eliminar la carga de páginas para nuestros usuarios.

React no tiene opiniones sobre qué tecnología usamos con él, podemos elegir las bibliotecas que usamos para: solicitudes http, enrutamiento front-end, estilo, etc.

React no nos impide usar la manipulación DOM ya integrada en JavaScript

React es muy popular (hay más descargas semanales en NPM para React que para Angular o Vue)

En la Encuesta para desarrolladores de StackOverflow.com de 2019 (una lectura divertida si tienes tiempo)

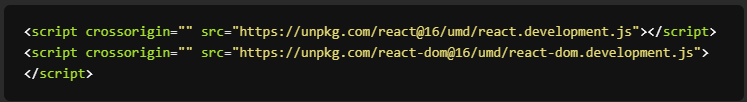
React ha superado a Angular / Angular.js en uso (solo después de jQuery)

React es el framework web más querido por sus desarrolladores, superando por poco a Vue

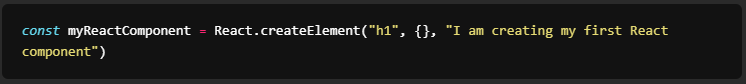
Utilizando React

React funciona utilizando objetos llamados componentes. Estos componentes se pueden crear de varias maneras, pero la idea general es dividir tu DOM en una variedad de componentes que tienen comportamientos específicos. Por ejemplo, si tenemos un formulario en nuestra página, podemos crear un componente solo para ese formulario. Entonces, podemos tener un componente separado sólo para la barra de navegación, y así sucesivamente.

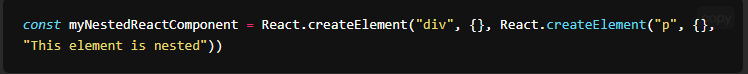
Si queremos usar React dentro de nuestro html (interactuaremos con React de manera un poco diferente en la siguiente sección), debemos incluir los scripts React en nuestro html.



Con React, podemos hacer un componente básico usando la funciòn React.createElement. React.createElementtiene tres parámetros principales. Primero, debemos decirle a React qué tipo de elemento queremos crear, digamos una etiquetah1. A continuación, podemos pasar los accesorios (props). Exploraremos este concepto con profundidad más adelante, pero por el momento, pasemos un objeto vacío usando {}. Por último, le diremos a React.createElementlos hijos que queremos incluir. Es decir, tenemos que decirle a React los elementos que queremos dentro de nuestra etiquetah1. Esto podría ser un string, como "Estoy creando mi primer componente React". O bien, podríamos anidar otro componente a través de otra llamada de funciónReact.createElement. Un ejemplo sería el siguiente:



¡Eso es! Si quisiéramos crear un componente de React anidado, podríamos hacer algo como esto:



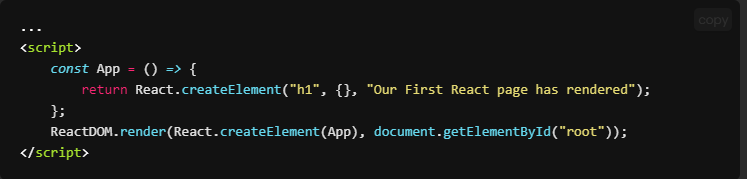
Genial, pero ¿cómo podemos usar esto en html? Tendremos que crear una página html y decirle a React que queremos mostrar los componentes en el DOM. Haremos esto cuando creemos nuestro primer componente React en la siguiente lección.

Primer React

Comencemos directamente con React. Como React es solo una biblioteca de Javascript, podemos conectarlo directamente a nuestro html. Primero, ve a la carpeta en la que deseas comenzar a crear tu proyecto React. Luego, dentro de esa carpeta, crea un archivo index.html. Luego, copia el siguiente código en tu archivo index.html:



Si abres index.html en tu navegador, deberías ver una etiqueta h1 que dice "First React page rendering ...". Además, hemos incluido dos scripts que necesitaremos para React. Todavía no hemos configurado React en nuestro index.html (aparte de incluir el Javascript externo), por lo que tendrás que esperar el resto del tiempo a que React se renderice. Entonces, agreguemos realmente scripts de React personalizados a nuestro archivo. Ya le hemos dicho a nuestro usuario que nuestra página React se está renderizando, así que hagámoslo! Después del último script de React externo, crea otra etiqueta de script para contener nuestro código. Agregaremos lo siguiente:



¿Que hicimos aquí? Creamos un nuevo método (usando la notación de flecha) que devuelve un nuevo elemento React. Ten en cuenta que le estamos pasando tres arguments al métodoReact.createElement.

Primero, tenemos un string que contiene h1. Esto le dice a React qué elemento queremos crear, en este caso es un elementoh1 html.

A continuación, tenemos llaves vacías donde pasaremos los props. No necesitamos preocuparnos por esto por el momento, aprenderemos sobre ellos más adelante.

Por último, enviamos a los hijos. Esto es lo que queremos crear dentro de nuestra etiquetah1. Esta podría ser una lista de muchos hijos diferentes, pero por ahora solo estamos enviando un string.

Después de definir el métodoApp, llamamos a ReactDOM.render. Le estamos diciendo a React que muestre nuestro componente APP dentro del elemento que tiene un id root.

Acabamos de escribir nuestro primer proyecto React. Si abres tu index.html, deberías ver brevemente "First React page rendering..." antes de ver "Our first React page has rendered".

Nuestro navegador mostrará primero el html que hemos escrito en el index. Luego, después de cargar los scripts React, nuestro navegador ejecuta el script que llamará al método ReactDOM.render. Este método sobrescribirá lo que originalmente teníamos en el div con id rootcon un h1que contiene "Our first React page has rendered ".

Si continúas actualizando tu página, verás que cambia rápidamente.

¡Felicidades! Has creado tu primer proyecto React!

JSX & Babel

Básicamente, JSX solamente proporciona azúcar sintáctica para la función React.createElement(component, props, ...children) .

Con React.createElement, sacrificamos la conocida sintaxis HTML fácil de leer. Afortunadamente, JavaScript Syntax Extension (JSX) fue diseñado para proporcionarnos algo de azúcar sintáctico. Desafortunadamente, JSX no es un lenguaje que podamos comenzar a escribir y esperar que las cosas funcionen correctamente. La advertencia con JSX es que no es un lenguaje que los navegadores puedan interpretar de inmediato. Si intentamos ejecutar código JSX en este momento, veríamos un error de sintaxis de JavaScript:

Uncaught SyntaxError: Unexpected token <

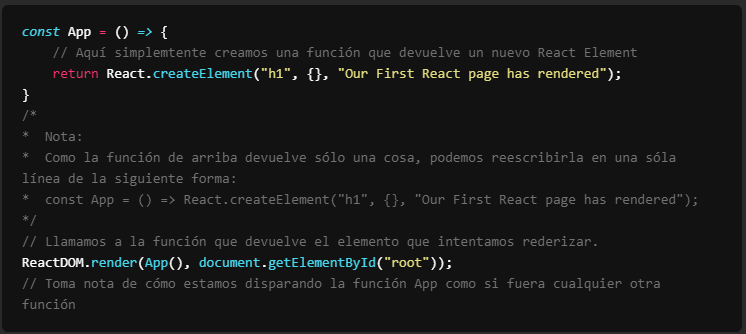
Para que JSX funcione, necesitamos usar un transpilador, llamado Babel.

Entrar a Babel

Babel es un transpilador que convertirá JSX en JavaScript para que los navegadores puedan entenderlo. Podemos agregar Babel a nuestra página usando una etiqueta de script simple.

Echemos un vistazo a la diferencia entre JSX y JavaScript.

JavaScript sin procesar (React.createElement)



JSX -> Babel -> Javascript



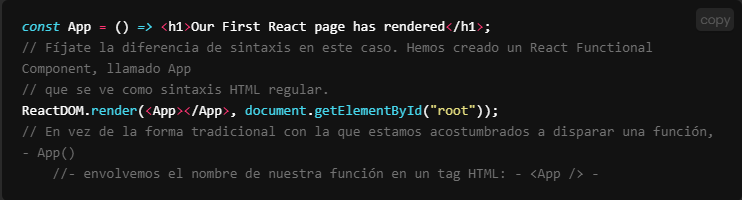
Entrar a JSX

JSX es una sintaxis opcional que nos permite:

1. Recuperar la sintaxis html normal
2. No tener que escribir React.createElement () todo el tiempo
3. Reducir la cantidad de código que escribimos.

Usar JSX no cambia las cosas. Estamos creando componentes React Functional pero, con JSX se ve mejor.

¿Recuerdas la nota mental que mencioné en el primer fragmento de código anterior sobre activar de la función App?



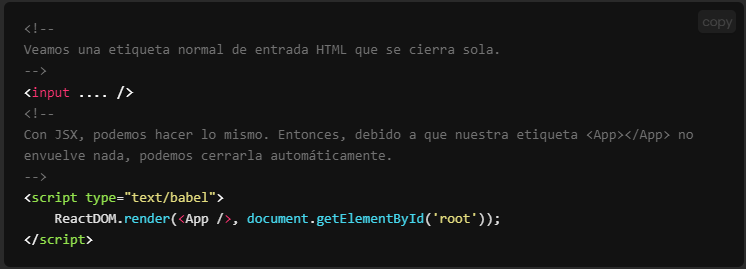
En el método ReactDOM.render ():

La forma en que distinguimos la diferencia entre un elemento HTML normal y un componente React, es que la primera letra de nuestro componente es mayúscula. Nota: React trata las letras minúsculas como etiquetas DOM.

Por ejemplo: si el nombre de la función de nuestra aplicación comienza con minúscula (es decir const app=()=> {...}:), cuando intentemos llamarla (<app></app>) desde ReactDOM.render (), React supondrá que estamos intentando utilizar una etiqueta html nativa con nombre apppero, como appno es una etiqueta html válida, arrojará el siguiente error:

*Warning: The tag app; is unrecognized in this browser. If you meant to render a React component, start its name with an uppercase letter.*

También podemos cerrar automáticamente nuestras etiquetas JSX. A diferencia del HTML normal, los elementos que no precisan cierre como hrdeben cerrarse explícitamente para que JSX sea válido. Por ejemplo, en JSX la etiqueta HTML <input>es una etiqueta de cierre automático: <input />.



Bastante ordenado, ¿eh? Puedes escribir código similar a HTML en JavaScript, y luego, al final, Babel compila ese JSX en llamadasReact.createElement() .

**IMPORTANTE: cuando creamos un proyecto a través de create-react-app, estamos instalando React. Dentro de nuestros archivos React, cuando escribimos**import React from 'react'**, podemos escribir en JSX dentro de ese archivo sin necesidad de agregar ninguna dependencia adicional.**

Recursos adicionales

JSX en profundidad : una inmersión más profunda en JSX

<https://reactjs.org/docs/introducing-jsx.html>

Traductor de Babel : muestra cómo JSX se transpila en JavaScript normal utilizando Babel

<https://babeljs.io/repl#?browsers=defaults%2C%20not%20ie%2011%2C%20not%20ie_mob%2011&build=&builtIns=false&corejs=3.21&spec=false&loose=false&code_lz=Q&debug=false&forceAllTransforms=false&shippedProposals=false&circleciRepo=&evaluate=false&fileSize=false&timeTravel=false&sourceType=module&lineWrap=true&presets=env%2Creact%2Cstage-2&prettier=false&targets=&version=7.19.3&externalPlugins=&assumptions=%7B%7D>

Create React App

Los proyectos modernos de JavaScript generalmente usan agrupadores (bundlers) para "empaquetar" el código fuente en archivos listos para producción. Uno de los empaquetadores más populares se llama Webpack. Webpack es una herramienta increíble que nos permite dividir nuestras aplicaciones en módulos más pequeños y transformarlas en base a herramientas llamadas loaders. Definitivamente es una tecnología que vale la pena investigar, y deberías pasar un tiempo trabajando en el tutorial de Inicio . <https://webpack.js.org/guides/getting-started/>

Sin embargo, Webpack no es el foco de este taller de React. En su lugar, nos apoyaremos en el testarudo kit de iniciocreate-react-app, que usa Webpack de forma encubierta.

create-react-app

Usar este kit de inicio es tan simple como ejecutar el siguiente comando en tu terminal:

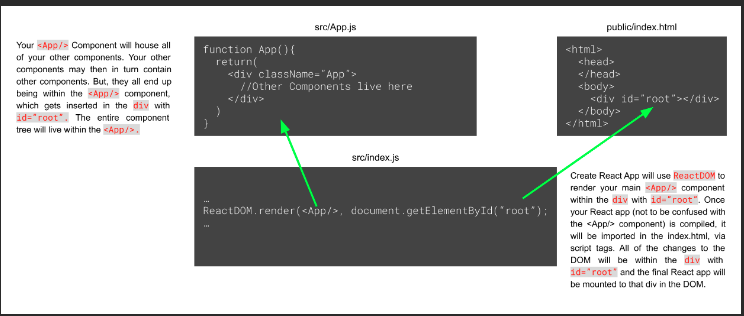
npx create-react-app your-project-name-here ( NOTA : npxno es un error tipográfico! Npx viene con npm versión 5.2+ y superior)

Una vez que se crea el proyecto, podemos navegar a la carpeta del proyecto recién creado y ejecutar

npm start

Esto ejecutará el servidor React del entorno de desarrollo, abrirá automáticamente una pestaña del explorador apuntando a nuestra aplicación. Este kit de inicio también viene con una función de recarga en vivo que volverá a cargar nuestra aplicación cada vez que guardemos cambios en nuestro código.

Cómo funcionará tu aplicación React:



Recursos adicionales

create-react-app

https://create-react-app.dev/

Iniciando Webpack

https://webpack.js.org/guides/getting-started/

Módulos ES6

https://hacks.mozilla.org/2015/08/es6-in-depth-modules/

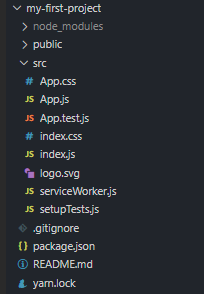
Introducción a los SourceMaps

https://web.dev/learn/

Estructura de carpetas de React

Como desarrollador de React, generalmente tendrás mucha libertad sobre cómo estructurar tus propios proyectos de React. Dado que React es técnicamente solo una biblioteca y, por lo tanto, muy popular, puedes tener control sobre ella. Presentaremos una forma muy simple de estructurar tu proyecto React. Esta estructura puede no ser ideal para proyectos grandes, pero para proyectos más pequeños, esto funcionará bien.

Si te sientes lo suficientemente cómodo con React, siéntete libre de experimentar con diferentes estructuras de carpetas. De todos modos, echemos un vistazo a la estructura actual que se crea con create-react-app.



El punto de entrada en nuestro proyecto React es el archivosrc/index.js. Este archivo es donde llamamos ReactDOM.rendery enviamos nuestro componente principal <App />. Vamos a poner todo el contenido en nuestro componente <App />en el divcon id=root. Al importar el componente<App />, podemos poner todo nuestro código React dentro de ese componente. Esto nos lleva a src/App.js. Este archivo será donde realmente podamos apuntar el resto de nuestro código. Se parece a esto:



Esta es la página de aterrizaje React predeterminada. Cuando navegues a tu proyecto y se ejecute npm run start, notarás que esta página se muestra en tu navegador. Sin embargo, no necesitamos este código para que nuestro proyecto funcione. Podemos eliminar todo dentro del <div className="App">. Sin embargo, la principal importación que necesitamos es import React from 'react'. Al importar React, podremos escribir en JSX, una extensión de sintaxis para JavaScript.

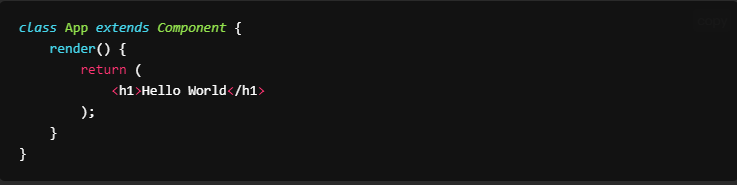
Dentro de nuestra carpetasrc, creemos una nueva carpeta llamada "componentes". En esta carpeta, crearemos todos nuestros componentes. Puedes pensar en un componente React como un divque contiene todo el estilo y JavaScript que necesitas.

Otras lecturas:

<https://reactjs.org/docs/introducing-jsx.html>

Usando JSX

JSX (JavaScript Syntax Extension) es una extensión de JavaScript que nos permite escribir código que se parece mucho a HTML sin tener que convertirlo en una cadena. Esta es la forma preferida de usar React.



iqueta HTML <h1> . Cualquiera que sea el contenido que deseemos hacer visible en la página, podemos escribirlo usando una sintaxis con la que ya estamos familiarizados.

Podemos usar cualquier etiqueta HTML con la que ya estamos familiarizados, prueba algunas como:

table

ul

form

input

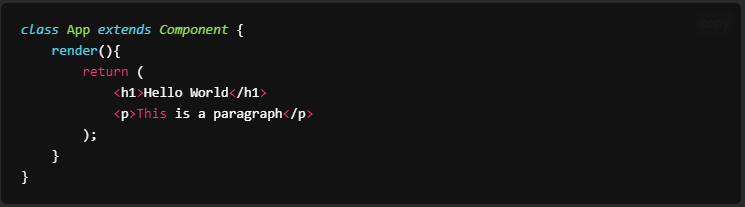
img

Algunas reglas a tener en cuenta

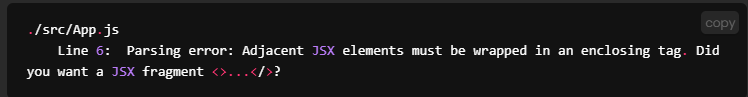
Sin embargo, no somos libres de poner lo que queramos.

Las expresiones JSX deben tener un elemento principal

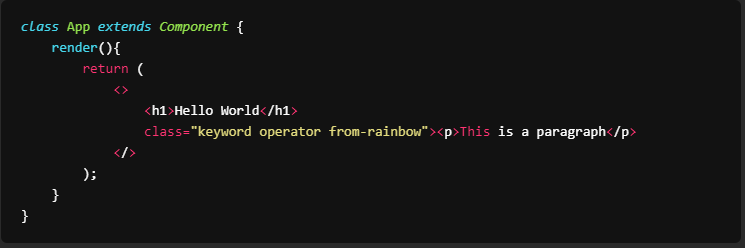
Considera el código de ejemplo



Cuando trabajamos con JSX, debemos asegurarnos de que nuestro código devuelto exista dentro de un elemento JSX. El ejemplo anterior no se compilará y devolverá el mensaje de error siguiente



Podemos colocar los elementos anteriores dentro de una sola etiqueta como <div> o si no queremos introducir elementos adicionales en el DOM, podemos hacer uso de un fragmento JSX como indica el mensaje de error anterior.



Versiones JSX de ciertos atributos HTML

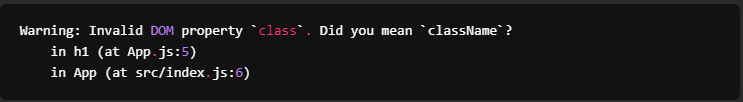
Considera el siguiente fragmento.



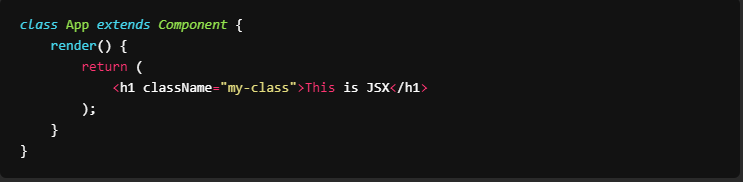
Si intentáramos usar esto en JSX, podríamos encontrar algo como



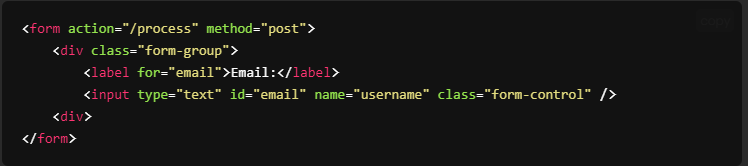
Pero fíjate en cómo tenemos class en la linea 1 y otra vez en la linea 4. Esto es más que un poco ambiguo, y cuando trabajamos con JSX, obtendremos un error en nuestra consola que nos dice



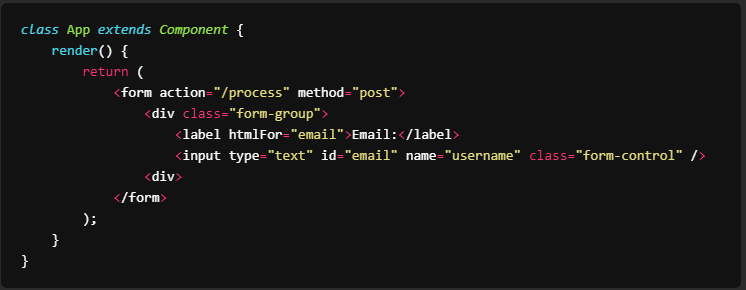
Cuando usamos la clase HTML, necesitamos usar el atributo className en JSX



Considera el ejemplo en el que podríamos tener un formulario en HTML.



Si reescribiéramos esto en JSX, ¿cómo debería JSX gestionar el for sobre lalinea 3 como el atributo forde HTML, y no como un forde JavaScript?



Para obtener información sobre estas diferencias, lee la excelente documentacion oficial.

<https://reactjs.org/docs/dom-elements.html>