Curso de React.js

# ¿Qué es React.js?

**React** cumple su función como biblioteca ya que para utilizar su código se debe importar. También es un Framework aunque las convenciones de cómo debe ser organizado todo no son estrictas.  
En este curso aprenderás las prácticas que la comunidad ha decidido que son buenas.

**React es declarativo**, lo que quiere decir que se le indica qué debe hacer pero no cómo debe hacerse, ahorrando de esta manera muchos pasos.

**JSX** es HTML dentro de Javascript, esto se verá más adelante en detalle.

React está estructurado por **componentes** que son como pequeños bloques de lego que al ser unidos forman aplicaciones de React. Estos componentes pueden tener estilos, ser enlazados a eventos y sus estados pueden ser modificados.

Con React también se tiene la ventaja de que será escrito una sola vez y podrá ser utilizado en aplicaciones web, móviles, entre otras.

# Pre-requisitos

Estos son los conocimientos que deberás tener antes de comenzar con este curso:

* **Desarrollo web online:** Esto implica tener familiaridad y fortaleza en el uso de HTML y CSS.
* **Javascript:** React es Javascript. Es importante saber usar Javascript en el navegador. Es deseable conocer JQuery y saber sobre promesas, clases y tener conocimientos sobre asincronía.
* **Terminal:** La línea de comandos es indispensable para instalar herramientas, correr servidores y hacer diversas tareas.

# Herramientas que usaremos

Estas son las herramientas que usaremos en el curso:

* **Navegador:** Especialmente Chrome ya que cuenta con óptimas herramientas de desarrollo.
* **React Developer Tools:** Es una herramienta Open Source creada por Facebook y tiene instalación para Chrome o Firefox. Nos dejará ver el código de React inspeccionando elementos.
* **Editor de texto:** Puedes usar cualquiera, en este curso sugerimos **Visual Studio Code**. Tiene muchos plugins útiles para el desarrollo.
* **Prettier:** Es un plugin que hace que el código se vea bien sin importar cómo está escrito.

# Create-react-app

Para todos los que estan siguiendo el tutorial al pie de la letra. Les cuento que el comando npm install -g create-react-app ya fue sustituido por npx. Asi que lo que tiene que hacer es lo siguiente. En caso de haber instalado previamente create-react-app se debe desinstalar:  
npm uninstall -g create-react-app.  
Y luego usar el comando npx create-react-app hello-react  
En caso de no tener instalado npx. Sencillamente instalarlo con:  
npm install -g npx

Hola a todos; para aquellos que intentan correr Create-react-app y obtienen vulnerabilidades al finalizar de descarga. **No lo intenten hacer por medio del código del profesor** debido a que es **Obsoleto.🐛** Intenten borrar la carpeta que les creo, y vuélvanlo hacer con este comando en su terminal.

npx create-react-**app** my-**app**

**cd** my-**app**

npm start

Esto generara la carpeta del archivo sin vulnerabilidades **“Por mucho una vulnerabilidad baja”**

**Borren la instalación que hicieron globalmente que el profesor les recomendó, por medio del siguiente comando** ya que el comando que les recomendé anteriormente lo hace de forma local y no global.

npm **uninstall** -g **create**-react-app

Para mayor información pueden consultar en el siguiente link <https://create-react-app.dev/docs/getting-started>

Fundamentos

# Clonar el código de GitHub

En esta clase vamos a comenzar clonando el código del proyecto del repositorio en GitHub.

Es importante que tú y yo tengamos un punto de partida en común. Así vamos a poder asegurarnos que cada cambio que yo haga en el código, tú también lo recibas.

Para hacer esto, en la terminal ve a una carpeta donde quieras que exista el proyecto. Entonces escribes lo siguiente:

$ git **clone** **https**://github.com/Sparragus/platzi-badges.git

Eso va a clonar el repositorio del curso a una carpeta llamada platzi-badges.

Ahora es necesario que te muevas a esa carpeta.

$ cd platzi-**badges**

Ahora necesitamos instalar todas las dependencias necesarias para poder correr el proyecto. Lo haremos utilizando npm.

$ npm **install**

Este proceso puede tardar un poco. Lo que esta haciendo es descargando todas las bibliotecas de código que el proyecto necesita.

Una vez haya concluido, estamos listos para echar a correr el servidor. Lo hacemos con el comando

$ npm run start

Cuando el servidor comience, automáticamente va a abrir una pantalla en el navegador con la aplicación.

Si todo salió bien, vas a ver una pantalla que dice “Hello, Platzi Badges”.

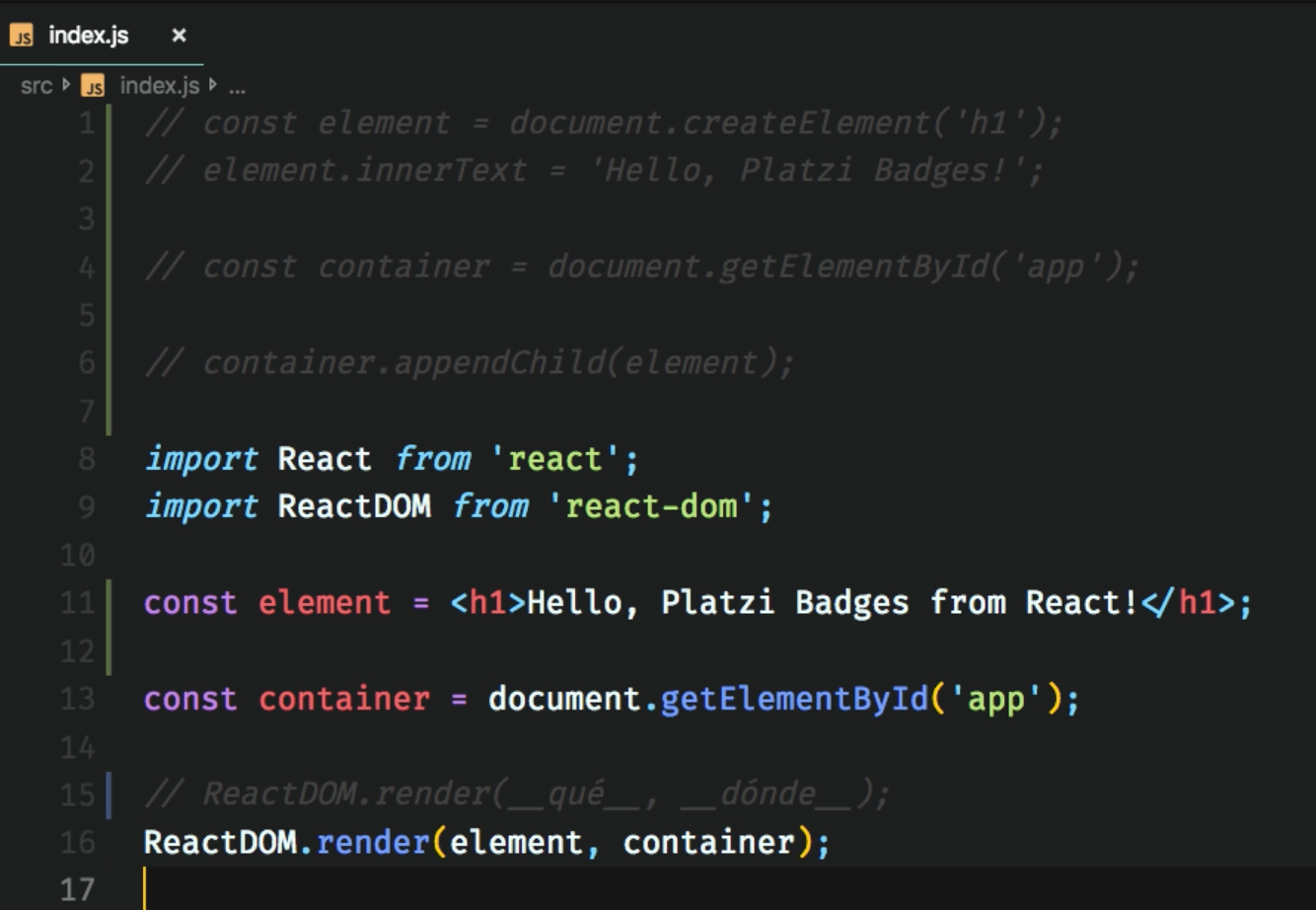
En la próxima clase vas a aprender como fue que “Hello, Platzi Badges” llegó desde el código hasta la pantalla de tu navegador.

\*[Aquí](https://github.com/sparragus/platzi-badges) encuentras el repositorio.

Lo que hice y me resultó, fué ir a la solapa de **"Archivos y enlaces"** y bajar el archivo **platzi-badges-1.ReactDOM.render.zip**. Lo descomprimí, modifiqué el nombre de la carpeta descomprimida a “**platzi-badges**”, luego ingresé a dicha carpeta y corrí **npm install**. Una vez terminado, corrí el comando **npm run start** y listo!  
Eso fue todo amigos!

# ReactDOM.render

* **React** y **ReactDOM** trabajarán en conjunto.
  + React como análogo a **createElement**
  + ReactDOM a **appendChild**
* **ReactDOM.render()** toma dos argumentos: Qué queremos renderizar y dónde lo queremos renderizar.
* Siempre que escribas **JSX** es requisito importar **React.**





# JSX

JSX es una extensión de JavaScript creada por Facebook para el uso con la biblioteca React. Sirve de preprocesador (como Sass o Stylus a CSS) y transforma el código generado con React a JavaScript.

JSX tiene su alternativa que es **React.createElement** pero es preferible JSX porque es mucho más legible y expresivo. Ambos tienen el mismo poder y la misma capacidad.

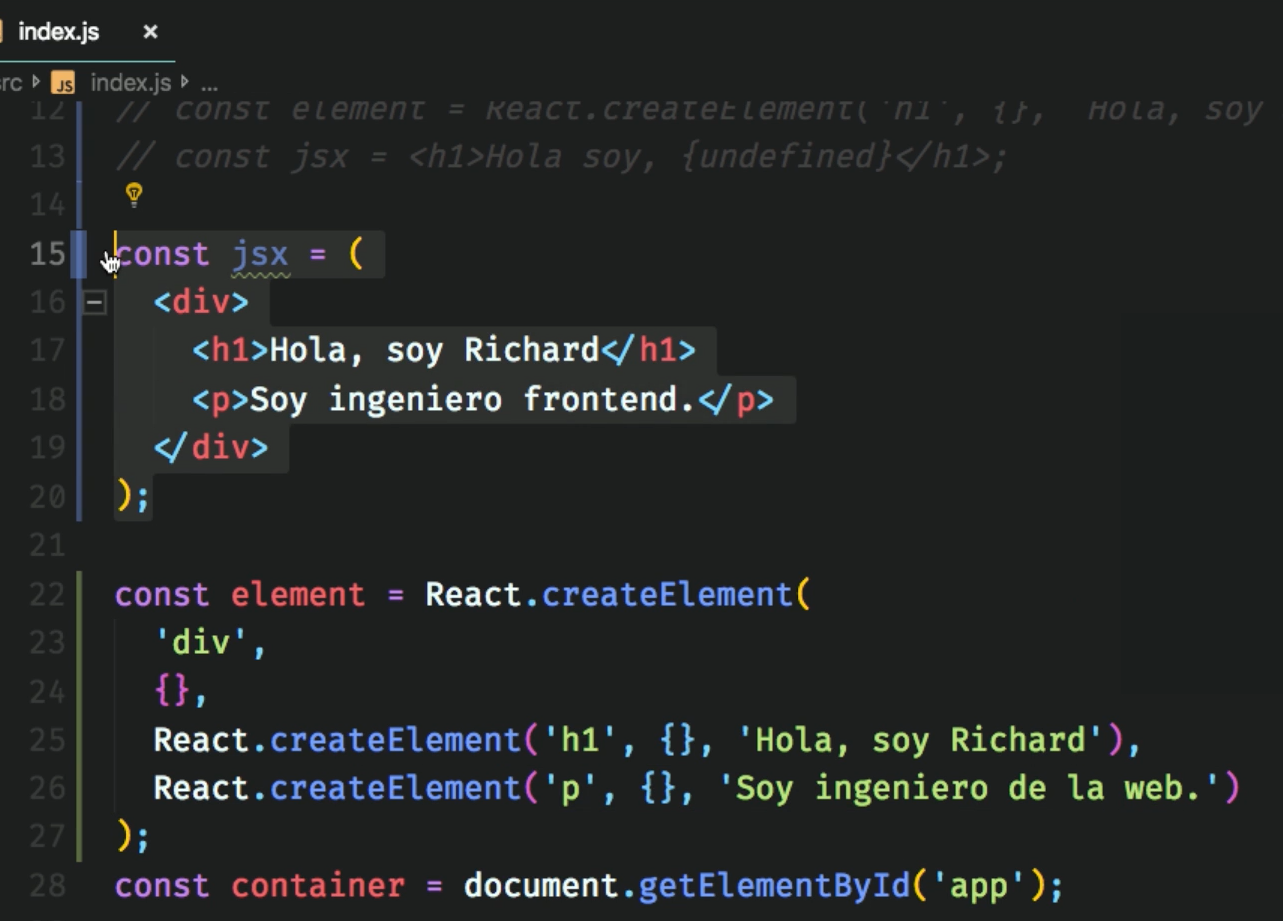
**React.createElement** recibe 3 argumentos:

* El tipo de elemento que estamos creando
* sus atributos o *props*
* y el *children* que es el contenido.

Ejemplo:  
React.createElement(‘a’, { href: ‘https://platzi.com’ }, ‘Ir a Platzi’);

En JSX se utilizan las llaves para introducir variables o expresiones de Javascript. Lo que sea que esté adentro se va a evaluar y su resultado se mostrará en pantalla.

Las expresiones pueden ser llamadas a otras funciones, cálculos matemáticos, etc. Si las expresiones son false, 0, null, undefined, entre otros, no se verán.



Creación y diseño de componentes

# ¿Qué es un componente?

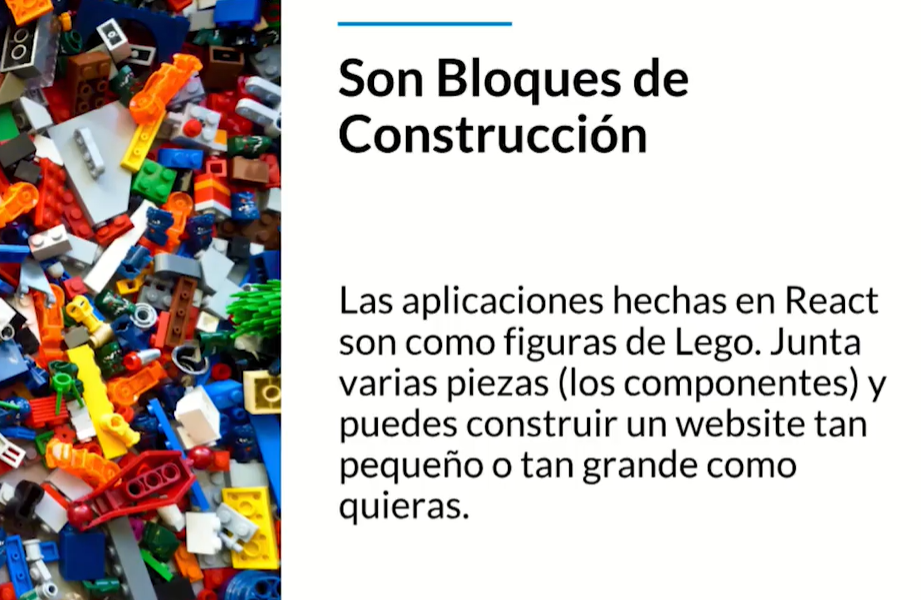
Los componentes en React son **bloques de construcción**.  
Las aplicaciones hechas con React son como figuras de Lego. Junta varias piezas (componentes) y puedes construir un website tan pequeño o tan grande como quieras.  
Los componentes serán barras de búsquedas, enlaces, encabezados, el *header*, etc.

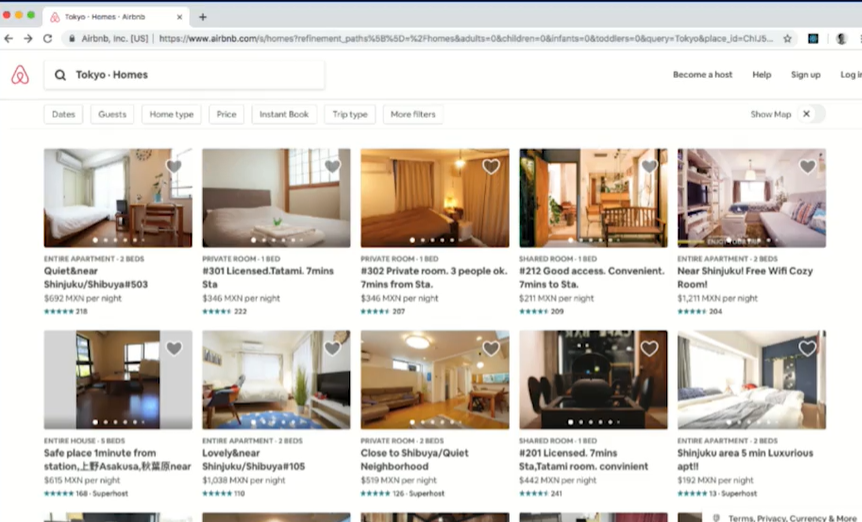
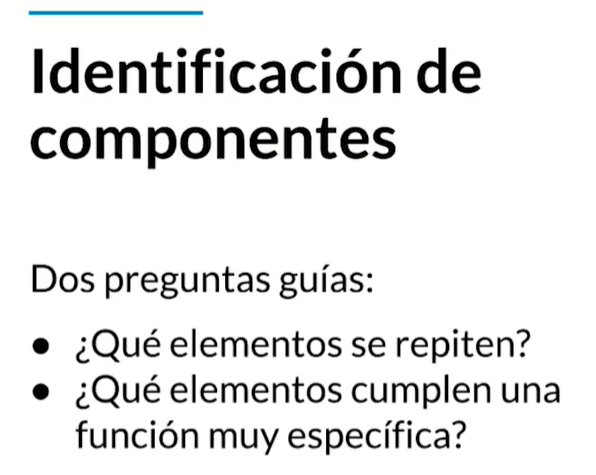
**”Componente” vs “elemento**  
Un elemento es a un objeto como un componente es a una clase. Si el elemento fuera una casa, el componente serían los planos para hacer esa casa.

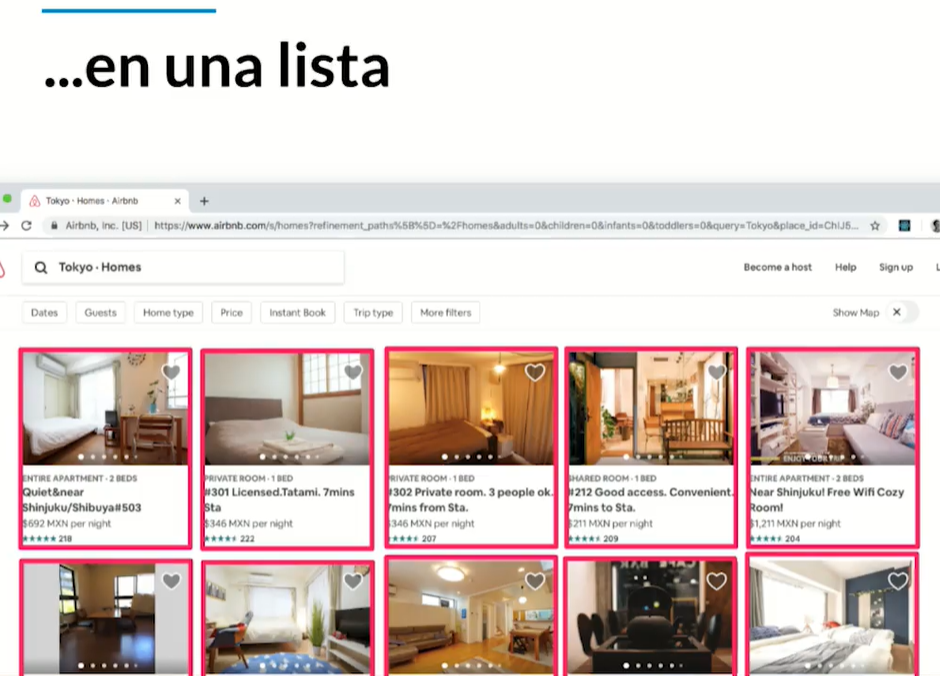
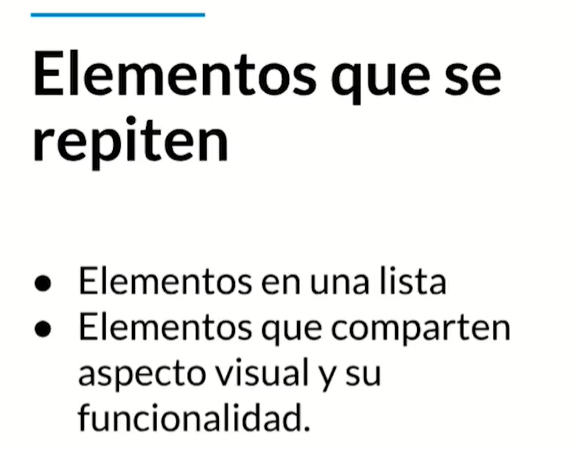
**Identificación de componentes**  
Para identificarlos debes hacerte las siguientes preguntas:

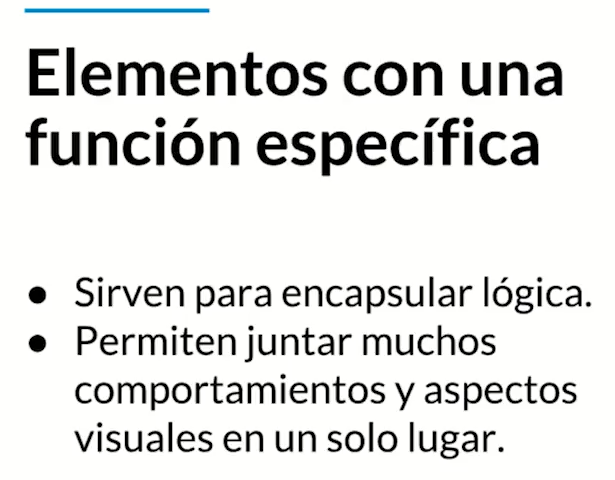
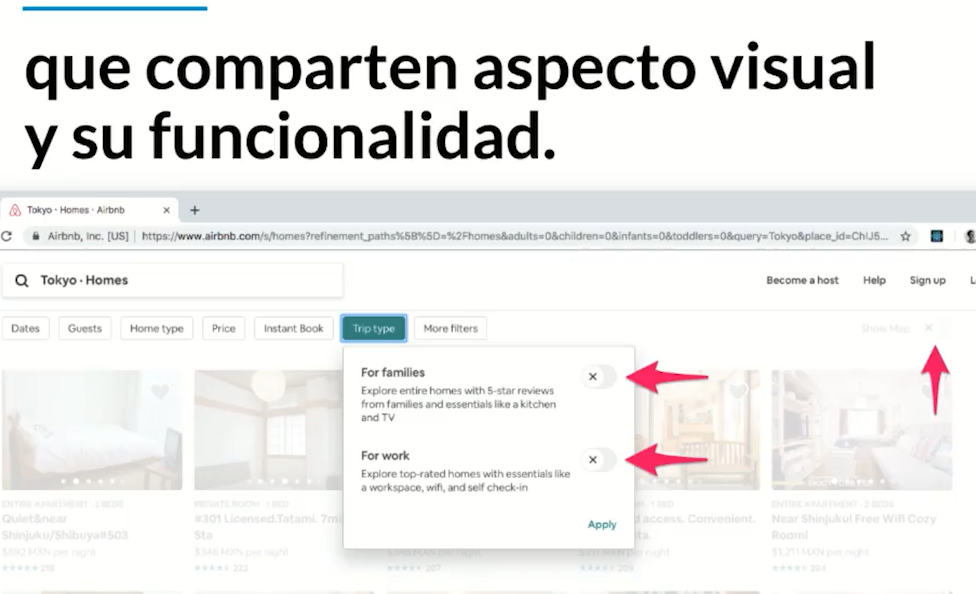
* ¿Qué elementos se repiten? Estos son los elementos en una lista o los que comparten aspecto visual y su funcionalidad
* ¿Qué elementos cumplen una función muy específica? Estos sirven para encapsular la lógica y permiten juntar muchos comportamientos y aspectos visuales en un solo lugar.

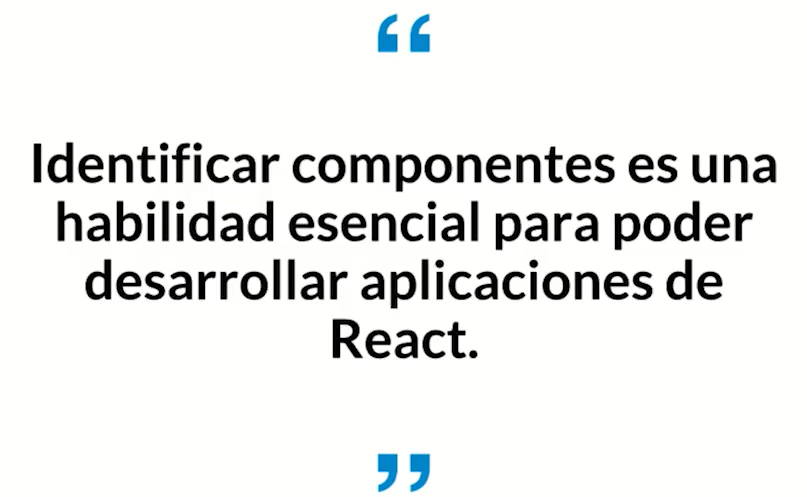
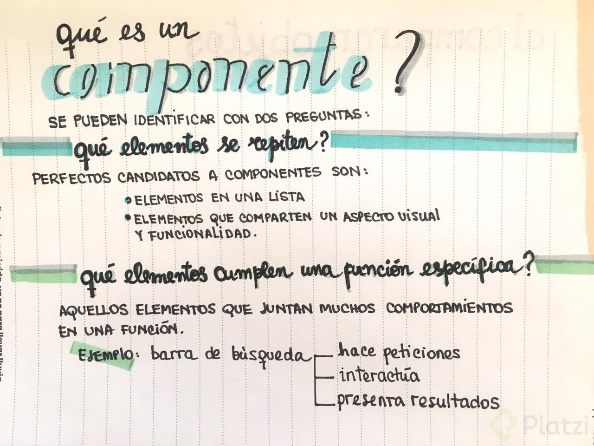
**Identificar componentes es una habilidad esencial para poder desarrollar aplicaciones de React.**



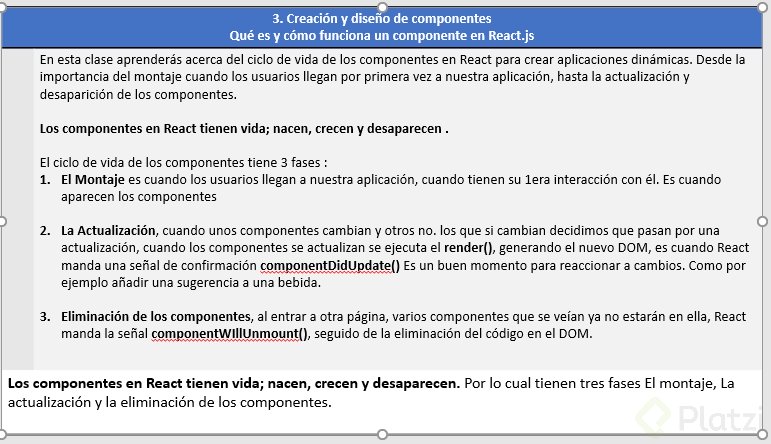


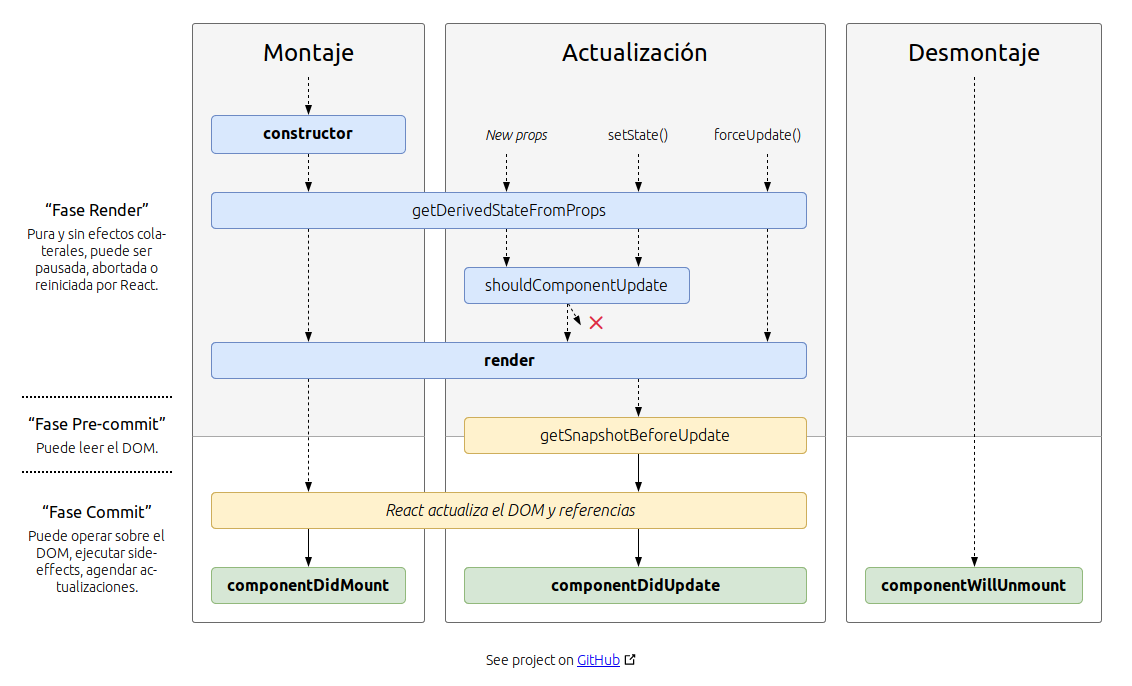




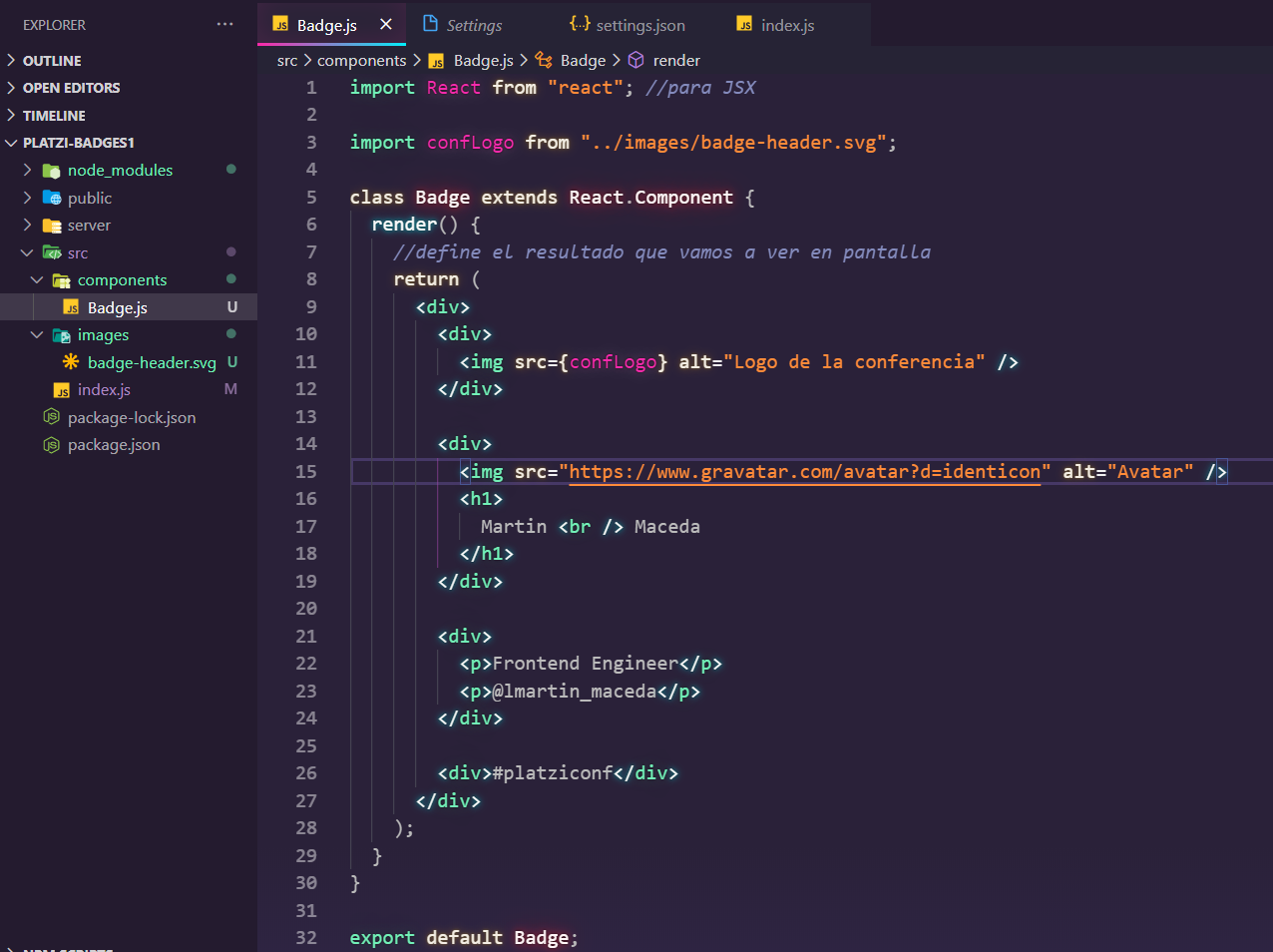
# Qué es y cómo funciona un componente en React.js





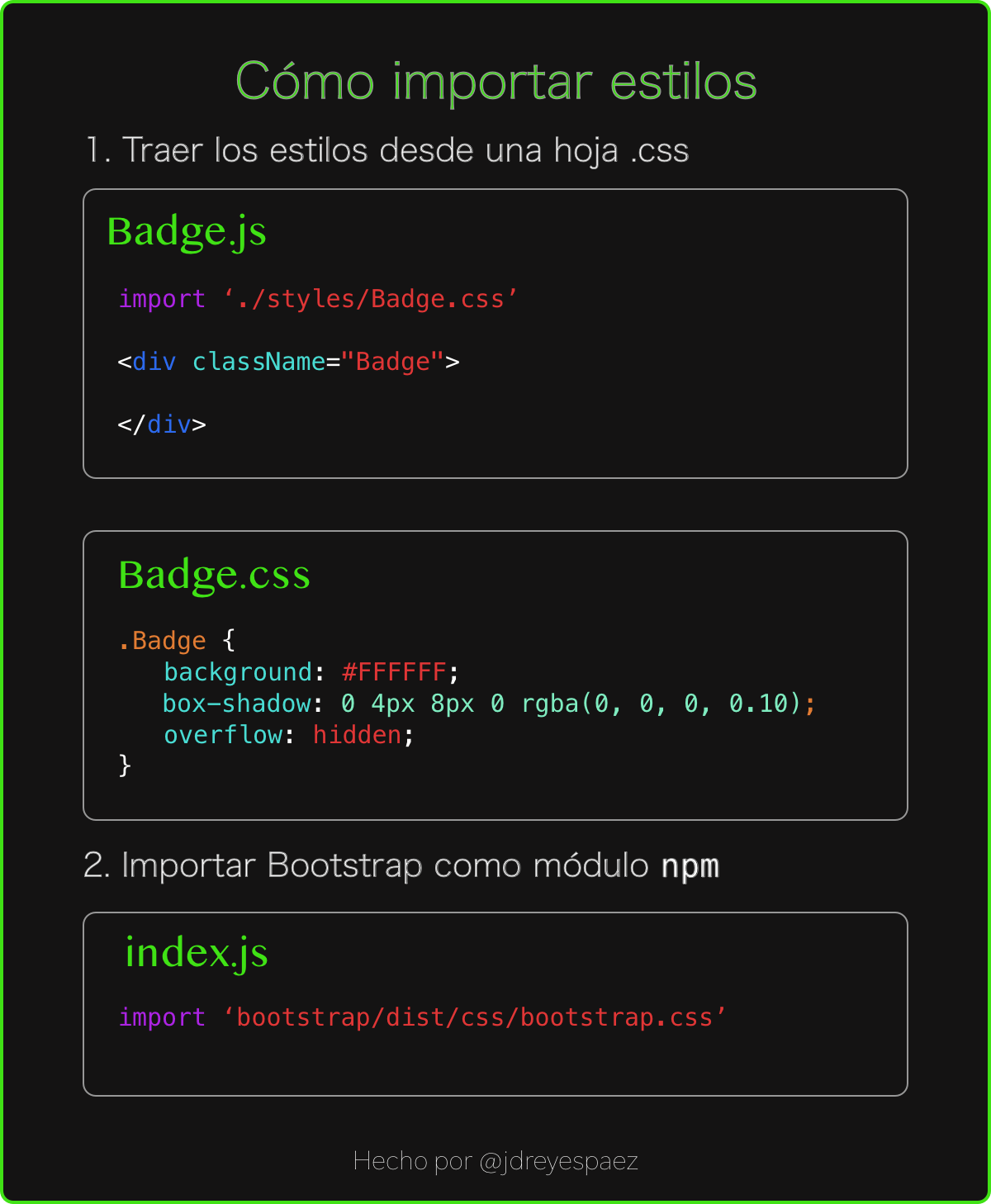
# Nuestro primer componente

* Es una buena práctica que los componentes vivan en su propio archivo y para ello se les crea una carpeta.
* Todos los componentes requieren por lo menos el método **render** que define cuál será el resultado que aparecerá en pantalla.
* El *source* de las imágenes en React puede contener direcciones en la web o se le puede hacer una referencia directa importándola. Si se importa deben usarse llaves para que sea evaluado.



# Cómo aplicar estilos

* Para los estilos crearemos una carpeta llamada **Styles** y allí vivirán todos los archivos de estilos que tienen que ver con los componentes.
* Para usar los estilos es necesario importarlos con *import*
* React funciona ligeramente diferente y para los atributos de clases no se utiliza *class* sino *className*
* Es posible utilizar **Bootstrap** con React, sólo debe ser instalado con npm install bootstrap y debe ser importado en el **index.js**
* Existen estilos que son usados de manera global o en varios componentes, así que deben ser importados en el index.js



En create-react-app tenemos una configuración de Webpack que nos permite importar estilos directamente en un archivo de javascript.  
.  
Para eso podemos importar directamente los estilos del componente con un importado del archivo

**import** './estilos.css'

.  
Ésto nos va a importar los estilos a nuestro archivo, pero los estilos no se aislan completamente (se importan en el momento de crear el componente, pero siguen interactuando con toda la aplicación)  
.  
Para evitar estos problemas de especificidad tenemos que generar convenciones de nombrado de estilos (clases), una de las más populares es BEM (Bloque\_\_elemento—modificador), ésto va a hacer que no se sobreescriban nuestros estilos.  
.  
Para aplicar los estilos sobre clases de CSS, la convención dice que envés de utilizar el atributo (prop) class, utilicemos className:

**const** Componente = ()=>(

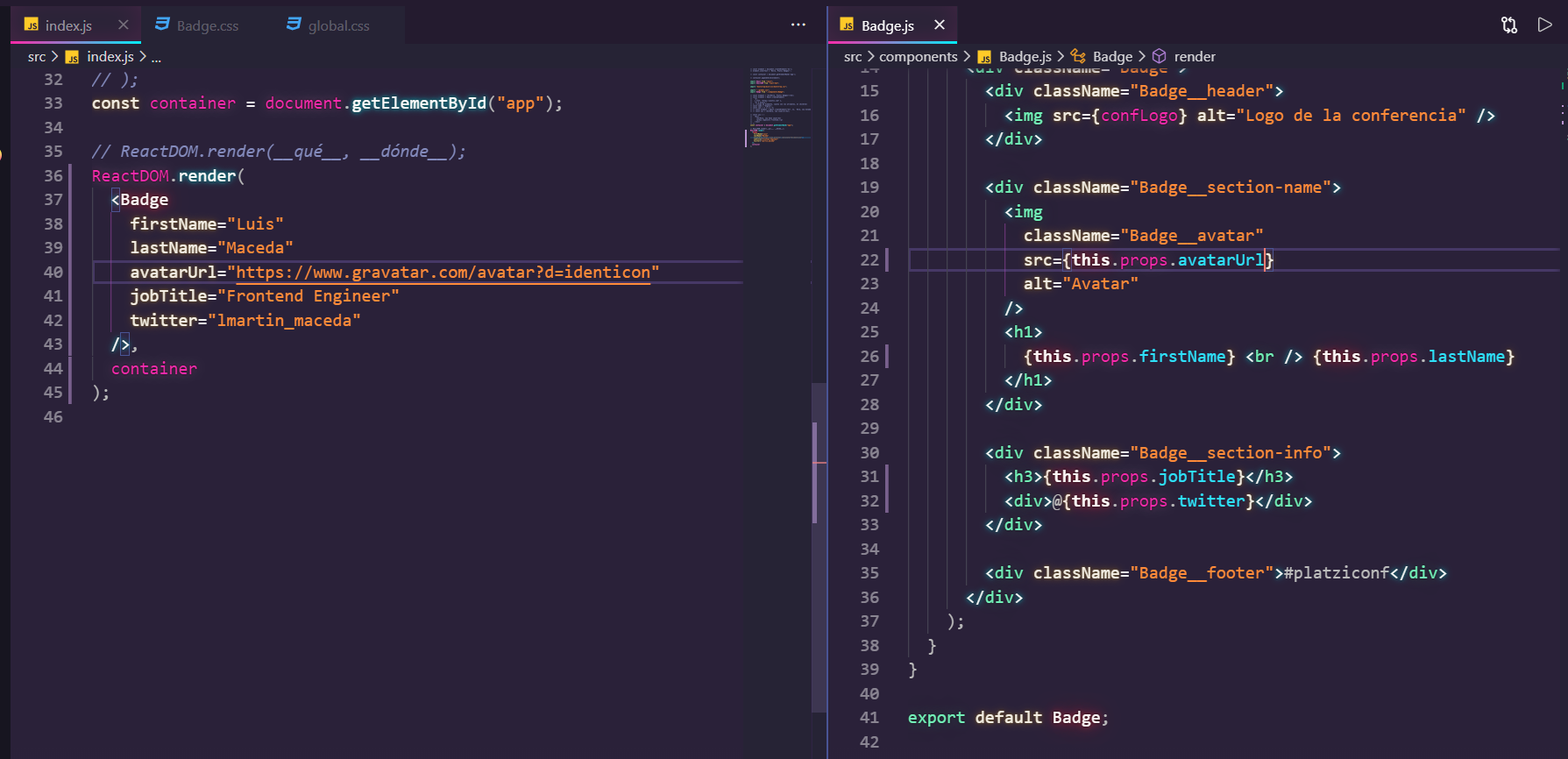
<div className="clase">Childrens del componente</div>

);

# Props

Los **props** que es la forma corta de properties son argumentos de una función y en este caso serán los atributos de nuestro componente como **class**, **src**, etc.

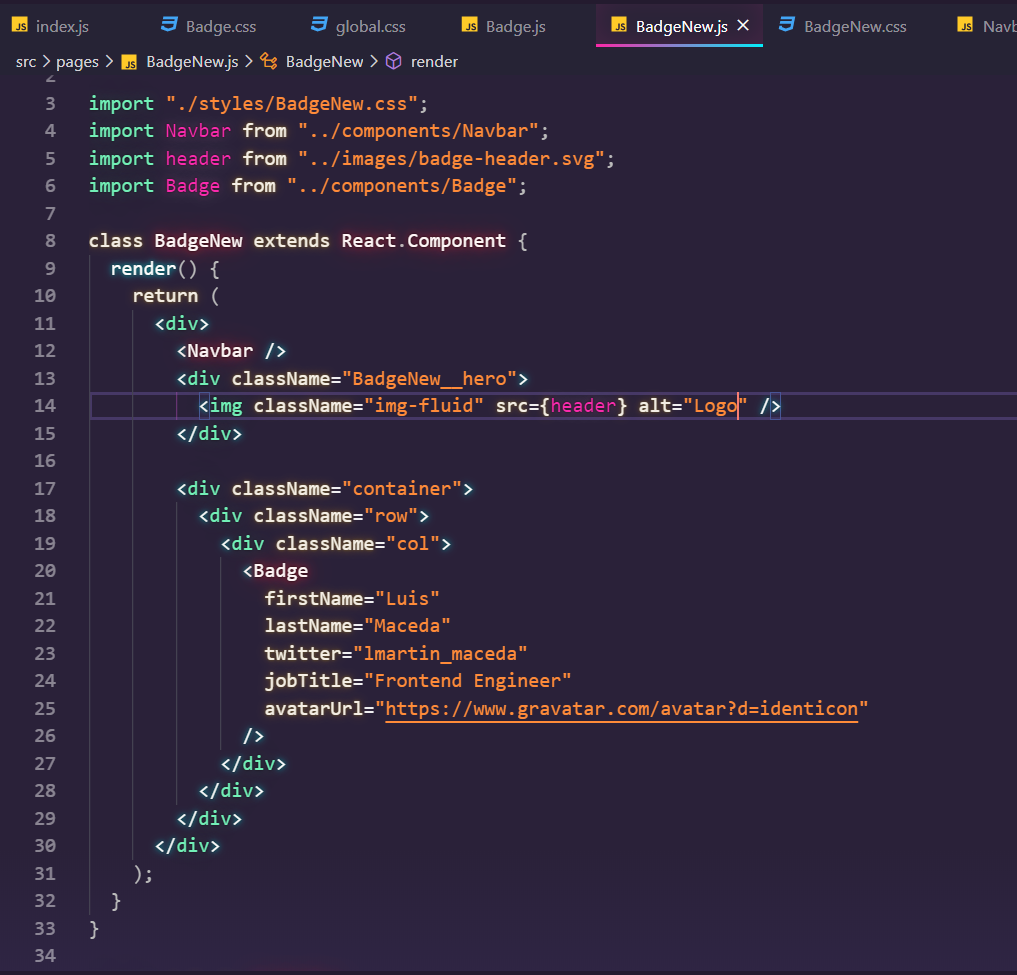
Estos props salen de una variable de la clase que se llama this.props y los valores son asignados directamente en el **ReactDOM.render()**.



# Nuestra primera página

Las páginas en React son componentes y conseguir distinguirlas nos servirá para saber que es un componente que adentro lleva otros componentes.

* Al escribir los *props* no importa el orden en el que lo hagas, únicamente importa el nombre.



# Enlazando eventos

* React dispone de **eventos**. Cada vez que se recibe información en un input se obtiene un evento **onChange** y se maneja con un método de la clase *this.handleChange*
* Los elementos *button* también tienen un evento que es **onClick**.
* Cuando hay un botón dentro de un formulario, este automáticamente será de tipo **submit**. Si no queremos que pase así hay dos maneras de evitarlo: especificando que su valor es de tipo *button* o manejándolo desde el formulario cuando ocurre el evento **onSubmit**.

