**Curso React.js**

**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.

**¿Qué es un componente?**

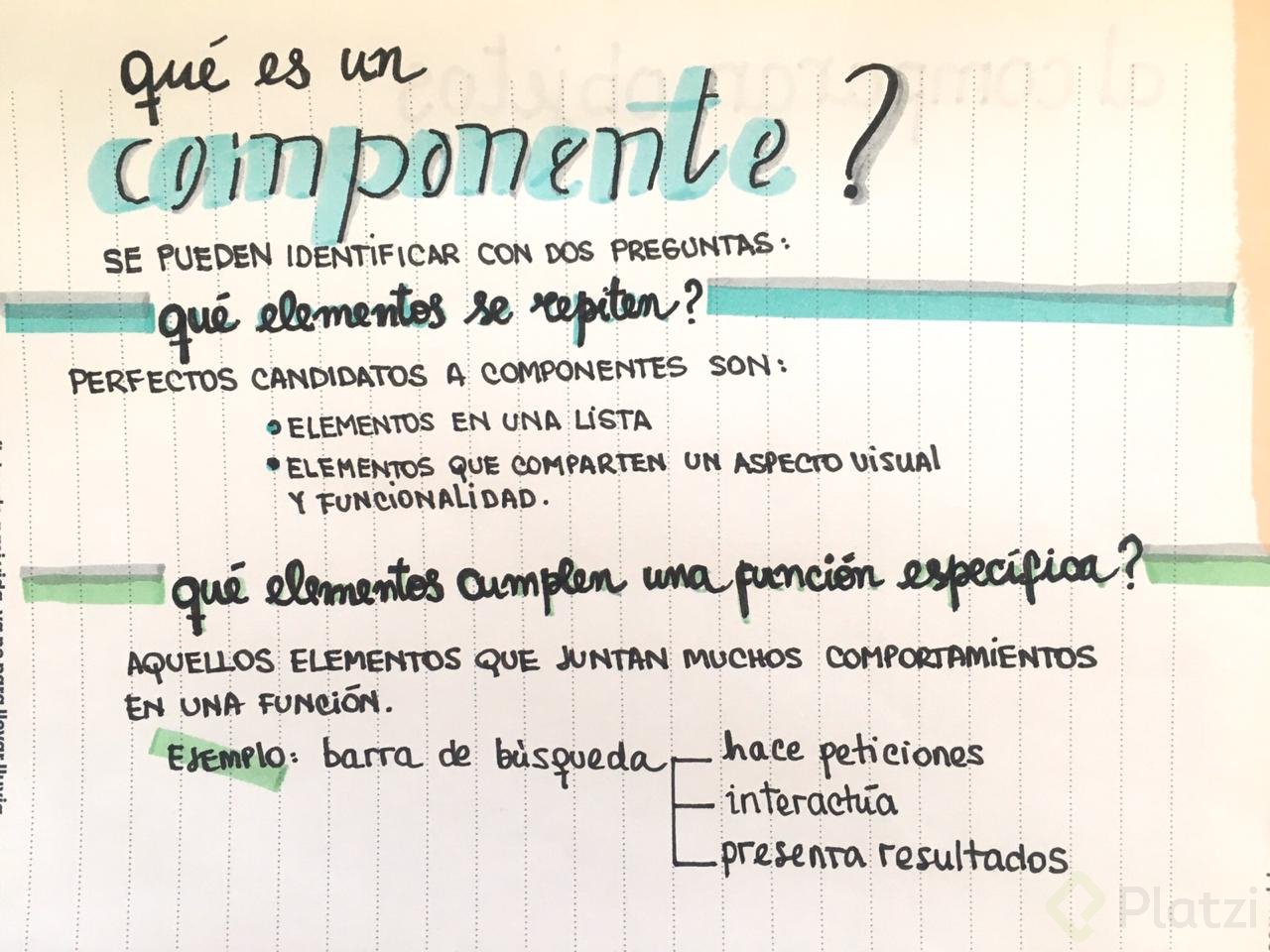
Los componentes en React son bloques de construcción.  
Las aplicaciones hechas con React son como figuras de Lego. Junta varias piezas (componentes) y puedes construir un website tan pequeño o tan grande como quieras.  
Los componentes serán barras de búsquedas, enlaces, encabezados, el *header*, etc.

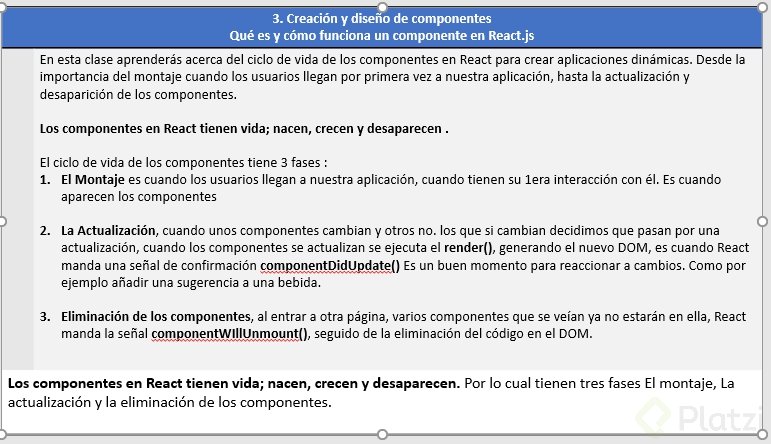
”Componente” vs “elemento  
Un elemento es a un objeto como un componente es a una clase. Si el elemento fuera una casa, el componente serían los planos para hacer esa casa.

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.





**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

**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().