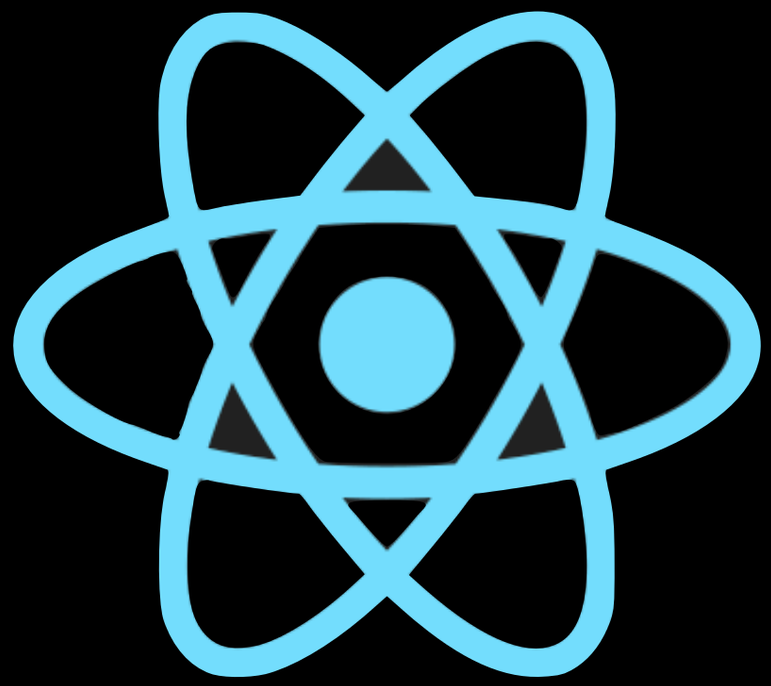
corebiz.



.hello

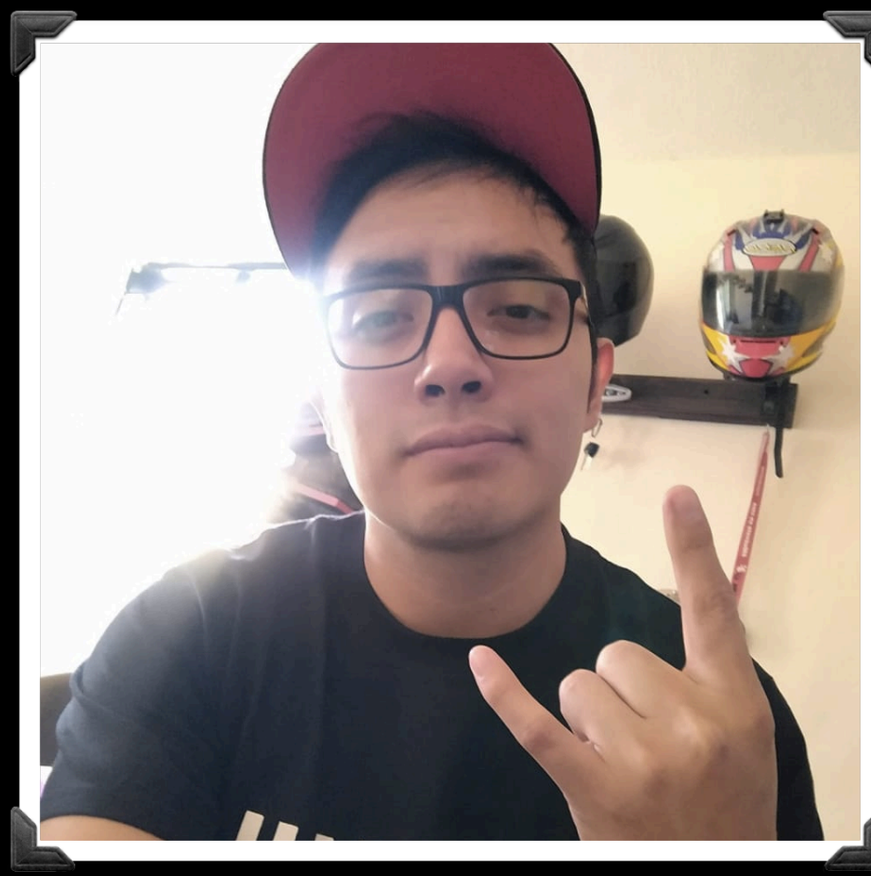
React JS

Presentadores

Fernando Robles

linkedin.com/in/fernandoroblesriveralerbios

github.com/starla01/



Axel Palacios

https://www.linkedin.com/in/axel-palacios-66a8aa118/

github.com/axl1dev/

Herramientas

Visual Studio Code

Terminal

Google Chrome

Pre requisitos

Hay algunas cosas que debe saber de antemano antes de comenzar a jugar con React. Si nunca antes ha usado JavaScript o DOM, por ejemplo, me familiarizaría con ellos antes de intentar abordar React.

Y estos son los requisitos previo que tenemos que considerar antes de entrar a abordar REACT JS

- Conocimientos Básicos con HTML & CSS
- Conocimientos Básicos de Javascript
- Familiaridad con la sintaxis de las funciones de ES6
- Node JS instalado.

Objetivo

Aprender sobre conceptos escenciales de REACT y términos relacionados, como Babel, componentes, props, estados o state.

Presentar una aplicación simple que demuestre los conceptos anteriores

¿Que es React?

- React es una biblioteca de Javascript, una de las mas populares, con mas de 100,000 estrellas en GitHub.
- React no es un framework
- React es un proyecto de código abierto creado por Facebook.
- React se utiliza par crear interfaces de usuario (UI).
- React es la capa de vista de una aplicacion.

Uno de los aspectos mas importantes de React es el hecho de que puede crear componentes, que son como elementos HTML personalizados y reutilizables, para construir interfaces de usuario de manera rápida y eficiente. React también agiliza la forma en que se almacenan y manejan los datos, utilizando el estate y los props.

JSX

Fundamentalmente, JSX solo proporciona azúcar sintáctica para la función

JSX

```jsx
<MyButton color="blue" shadowSize={2}>
  Haz click en mí
</MyButton>
```

Javascript

```jsx
React.createElement(
  MyButton,
  {color: 'blue', shadowSize: 2},
  'Haz click en mí'
)
```

JSX

Consideremos la declaración de la siguiente variable

```
const element = <h1>Hello, world!</h1>;
```

Esto no es un string ni tampoco una variable

Es JSX y es una extension de la sintaxis de JS y es recomendable usarlo con REACT para construir las interfaces de usuario

Expresiones JSX

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

JSX

Esto significa que puedes usar JSX dentro de control de JS asignarlo a variables, aceptarlo como argumentos y retornarlo

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```

Especificando Atributos

Se puede usar comillas para especificar strings literales como atributos

```
const element = <div tabIndex="0"></div>;
```

Pero también puedes insertar expresiones JS en un atributo usando { }

```
const element = <img src={user.avatarUrl}></img>;
```

Nomenclatura camelCase

Dado que JSX es mas cercano a JS que a HTML, React utiliza la convención de nomenclatura camelCase en vez de nombres de atributos HTML

Pro ejemplo, “class” se convierte en “className, y así en listo otros ejemplos

tabindex -> tabIndex

z-index -> zIndex

background-color -> backgroundColor

Renderizado de elementos

Digamos que hay un elemento en alguna parte de nuestro archivo html:

```
<div id="root"></div>
```

Es llamado nodo “raiz” por que todo lo que este dentro de el será manejado por REACT DOM

Las apps construidas solo con REACT usualmente tienen un único nodo “root” en el DOM. Dado el caso de que estes integrando REACT en una app existente, puedes tener tantos nodos raiz del DOM aislados como quieras.

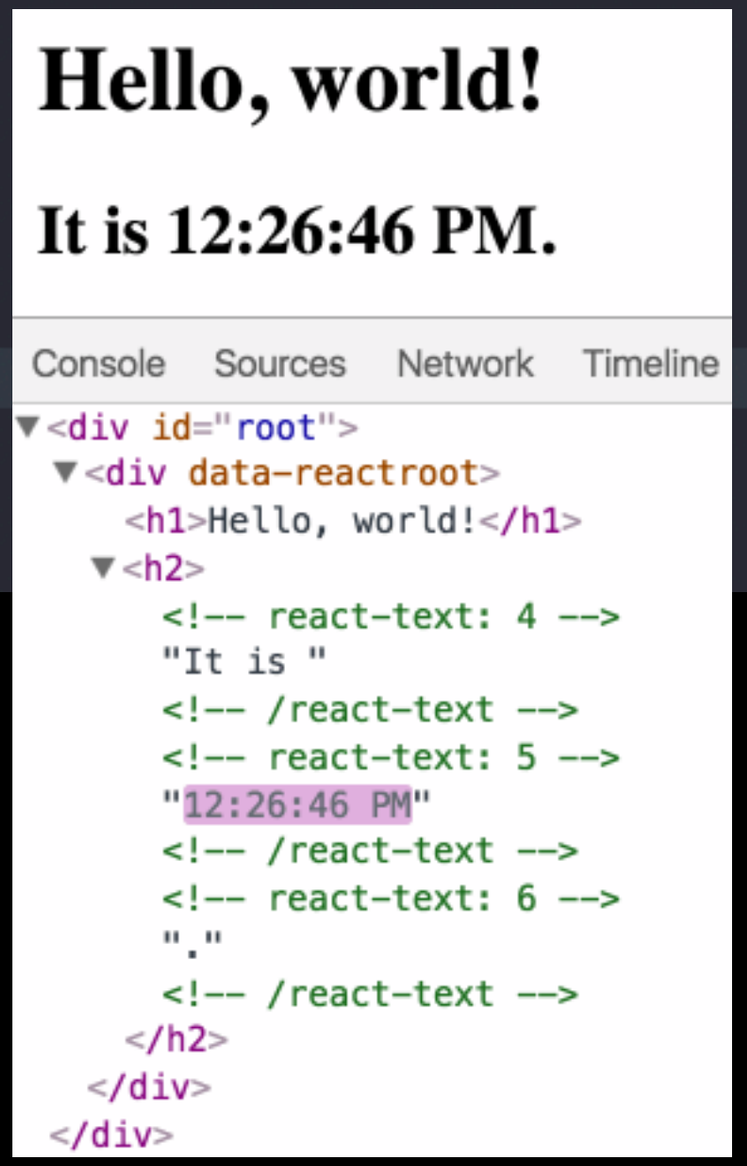
Para renderizar un elemento de REACT en un nodo “root”, ambos se tienen que pasar a ReactDOM.render()

```
const element = <h1>Hello, world</h1>;
ReactDOM.render(element, document.getElementById('root'));
```

Actualizando elementos renderizados

Los elementos de REACT son inmutables. Una vez creado, no puedes cambiar a sus hijos o atributos, Un elemento es como un fotograma solitario en una película: este representa la interfaz de usuario en cierto punto en el tiempo.

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root'));
}

setInterval(tick, 1000);
```



Componentes y props

Conceptualmente, los componentes son funciones de JS. Aceptan entradas arbitrarias (props “de solo lectura”) y devuelven a REACT Elementos que describen lo que debe aparecer en pantalla.

La forma más sencilla es escribir una función.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

Renderizado de un componente

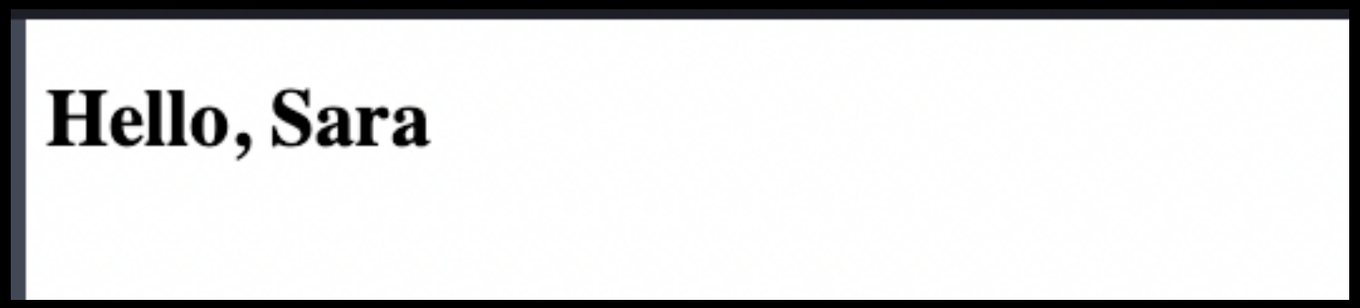
```
const element = <Welcome name="Sara" />;
```

Ejemplo completo

Conceptualmente, los componentes son funciones de JS. Aceptan entradas arbitrarias (props) y devuelven a REACT Elementos que describen lo que debe aparecer en pantalla.

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="Sara" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Resultado



Documentación

react-4-beginners-1 > src > components > Button > index.js > ...

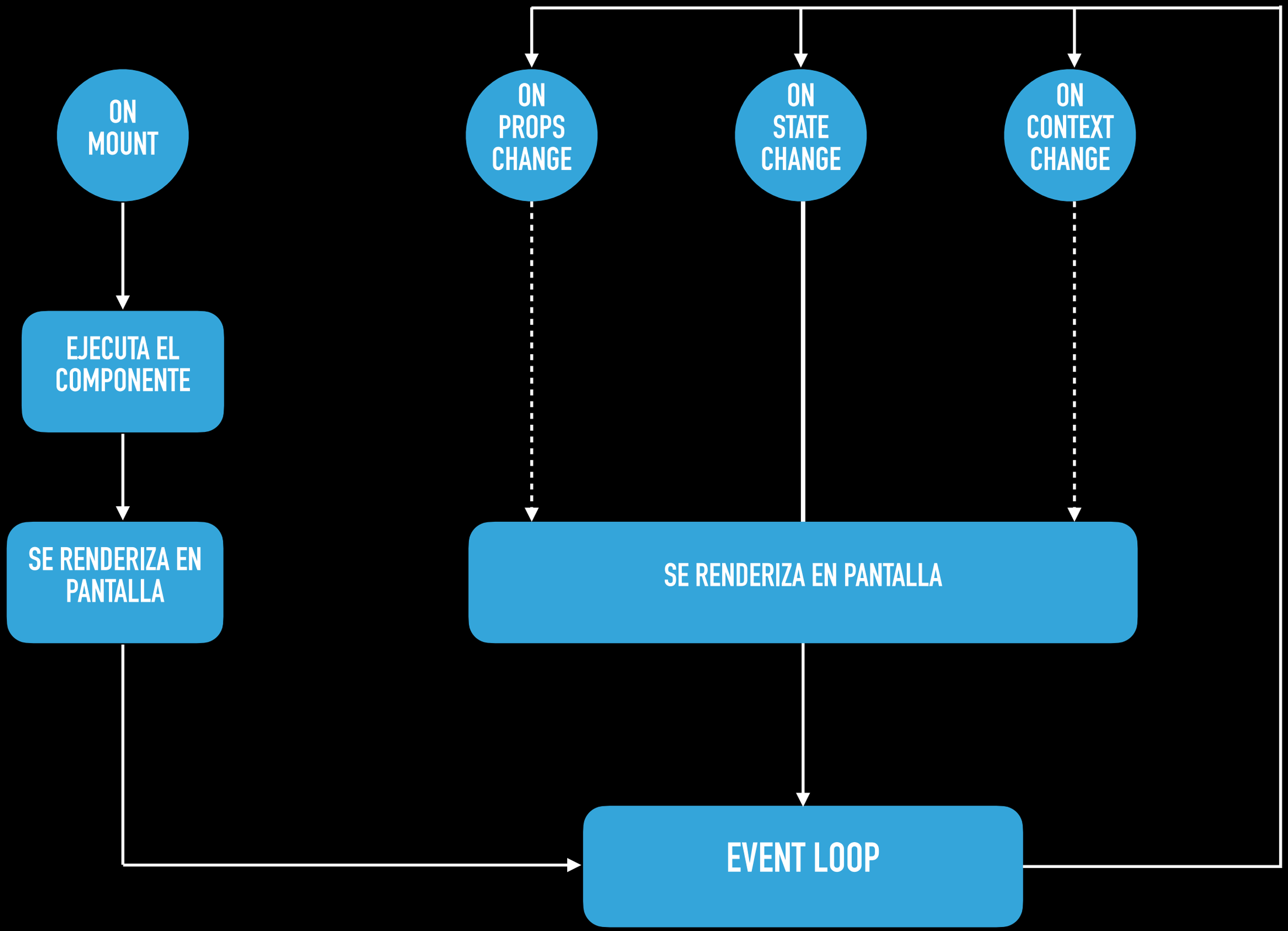
```jsx
import React from "react";
import styles from "./index.module.sass";

/**
 * @name Button
 * @description Componente para renderziar un boton o un Link segun el type
 *
 * @param {type} props.type props del tipo de componente button|link
 * @param {function} props.callback funcion que ejecutara en el evento click del boton
 * @param {string} props.url url a lacual redigira en caso de que el type sea Link
 * @param {Node} props.children Texto a renderizar
 * @param {string} props.classname Clase custom (sass) para estilizar el componente
 * @param {string} props.target Clase custom (sass) para estilizar el componente
 *
 * @example
 *
 *          <Button type="button" callback={function}>
 *            visita nuestro sitios de corebiz
 *          </Button>
 *
 *          <Button type="link" url="https://www.corebiz.ag" target="_blank">
 *            visita nuestro sitios de corebiz
 *          </Button>
 *
 * @returns {React.Component}
 **/

function Button({ callback, url, target, children, classname }) {
  return (
    (classname === "button" && (
      <button onClick={callback} className={classname}>
        {children}
      </button>
    )) || (
      <a href={url} target={target}>
        {children}
      </a>
    )
  );
}

export default Button;
```

¿Que son los hooks en React?

Son funciones proporcionadas por React que nos permiten contactarnos a las funciones de React desde nuestros ***componentes funcionales***

Nos centraremos en los 2 Hooks mas importantes "**useState**" y "**useEffect**" que seguramente usaremos mas en nuestros componentes de React

useState Flow



useState

```jsx
import React, { useState } from 'react';

function Example() {
  // Declara una nueva variable de estado, que llamaremos "count".
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>You clicked {count} times</p>
      <button onClick={() => setCount(count + 1)}>
        Click me
      </button>
    </div>
  );
}
```