# Conceptos importantes de React

React construye “Single Page Applications”

- Aplicaciones que cargan una sola página de HTML y cualquier actualización la rhacen re-escribiendo el HTML que ya tenía.

-> npm install -g create-react-app
*para instalar la librería de React JS*

-> create-react-app nombreApp
*para crear una nueva aplicación de react js*

-> npm run start
*levanta la aplicación de react en el navegador*

-> npm install
*para instalar las dependencias de un proyecto al clonarlo de git/lab/hub/bitbucket etc.*

-> npm install bootstrap
*añade bootstrap css a nuestro proyecto*

- Carpeta: ***src***

Es donde se encuentra todo nuestro código de react.

- Archivo: ***index.js***

Punto de entrada a la aplicación.

- Concepto: ***Babel***

Traduce nuestro “javascript moderno” a un JS que todos los navegadores puedan interpretar.

- Estructura: ***Componente***

Es de lo que se arma react, son escencialmente piezas de lego. Bloques de construcción.

- Estructura: ***Elemento***

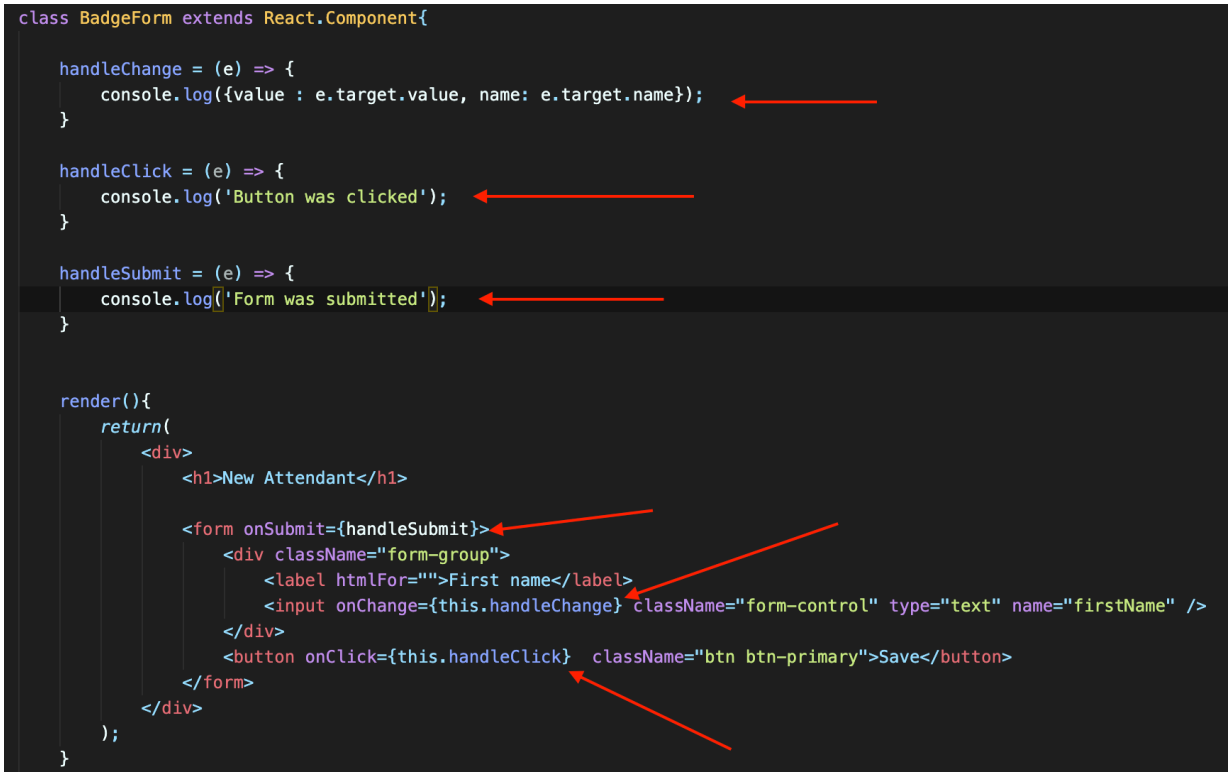
Un elemento es un objeto como un componente es a una clase.
Si el elemento fuera una casa el componente sería los planes para hacer esa casa.

- Css en html: ***className***

Para asignar una clase de css se debe fijar ‘className’ en vez de solo ‘class’

- Propiedad de React: ***Eventos***

React nos provee de un fenómeno llamado evento para la interacción de la UI. Un ejemplo de esto es el evento ‘onChange={}“ en la etiqueta html *input*, el ‘onClick={}’ en el botón o el ‘onSubmit={}’ en el form. Para controlar estos eventos debemos crear métodos llamados ‘handleChange’ (para textos), ‘handleClick’ (para botones) y ‘handleSubmit’ (para el formulario) estos métodos deben estar dentro del mismo archivo JS, arriba del render.



```
class BadgeForm extends React.Component{

    handleChange = (e) => {
        console.log({value : e.target.value, name: e.target.name});
    }

    handleClick = (e) => {
        console.log('Button was clicked');
    }

    handleSubmit = (e) => {
        console.log('Form was submitted');
    }


    render(){
        return(
            <div>
                <h1>New Attendant</h1>

                <form onSubmit={handleSubmit}>
                    <div className="form-group">
                        <label htmlFor="">First name</label>
                        <input onChange={this.handleChange} className="form-control" type="text" name="firstName" />
                    </div>
                    <button onClick={this.handleClick}  className="btn btn-primary">Save</button>
                </form>
            </div>
        );
    }
```

## Orden de ejecución de métodos en un componente

1. Constructor

```
constructor(props) {
  super(props);
  console.log("1. Constructor");
```

2. render

```
render() {
  console.log("2. Render");
```

3. componentDidMount

```
componentDidMount() {
  console.log("3. componentDidMount");
```

4. componentDidUpdate

```
componentDidUpdate() {
  console.log("4. componentDidUpdate");
```

5. componentWillUnmount

```
componentWillUnmount() {
  console.log("5. componentWillUnmount");
```

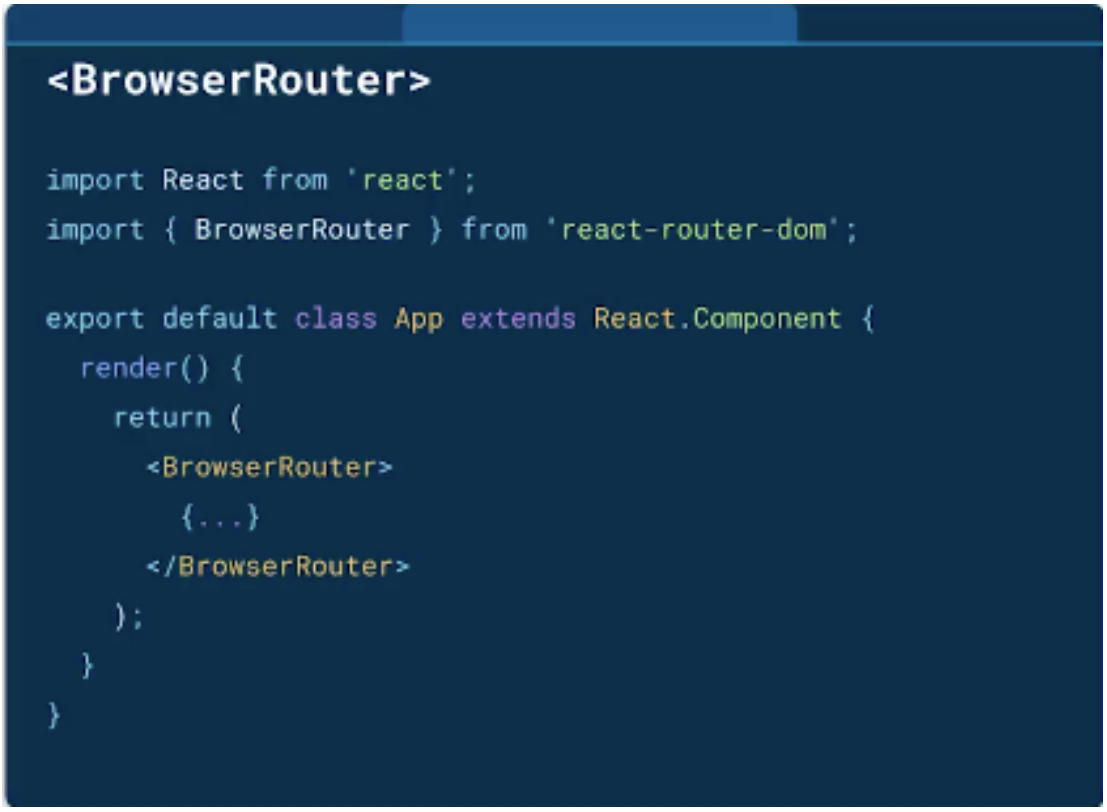
## Formas de routear en React

*-> npm install react-router-dom*

## React Router (v4)

- Nos da las herramientas para poder hacer SPA fácilmente.
- Usaremos 4 componentes:
  - `BrowserRouter`
  - `Route`
  - `Switch`
  - `Link`

**BrowserRouter**: Va en lo más alto del renderizado. Lo que hará es que todo lo que esté dentro de nuestra etiqueta funcionará como una SPA. Lo que permitirá es que para las demás herramientas de routing funcionen.




**Route:** Representa una dirección de internet.
*Ejemplo:* <Route path="/" component={Home}/>

- Cuando hay un match con el path, se hace render del componente.
- El component va a recibir tres props: match, history, location.

**Swtich:** Servirá para presentar 1 sola ruta de varias, es parecido a metodo de condicionalidad 'switch'.

- Dentro de switch solamente van elementos de Route.
- Switch se asegura que solamente un Route se renderea.

**Link:** Tomará el lugar del elemento ancla (<a> </a>). Hará lo mismo pero sin tener que recargar la página completa.

- Toma el lugar del elemento <a>.
- Evita que se recargue la página completamente.
- Actualiza la URL.

Así es como se vería un componente que sólo es de rutas:

```
function App(){
    return (
        <BrowserRouter>
            <Layout>
                <Switch>
                    <Route exact path="/" component={Home} />
                    <Route exact path="/badges" component={Badges}/>
                    <Route exact  path="/badges/new" component={BadgeNew}/>
                    <Route component={NotFound} />
                </Switch>
            </Layout>
        </BrowserRouter>
    );
}

export default App;
```