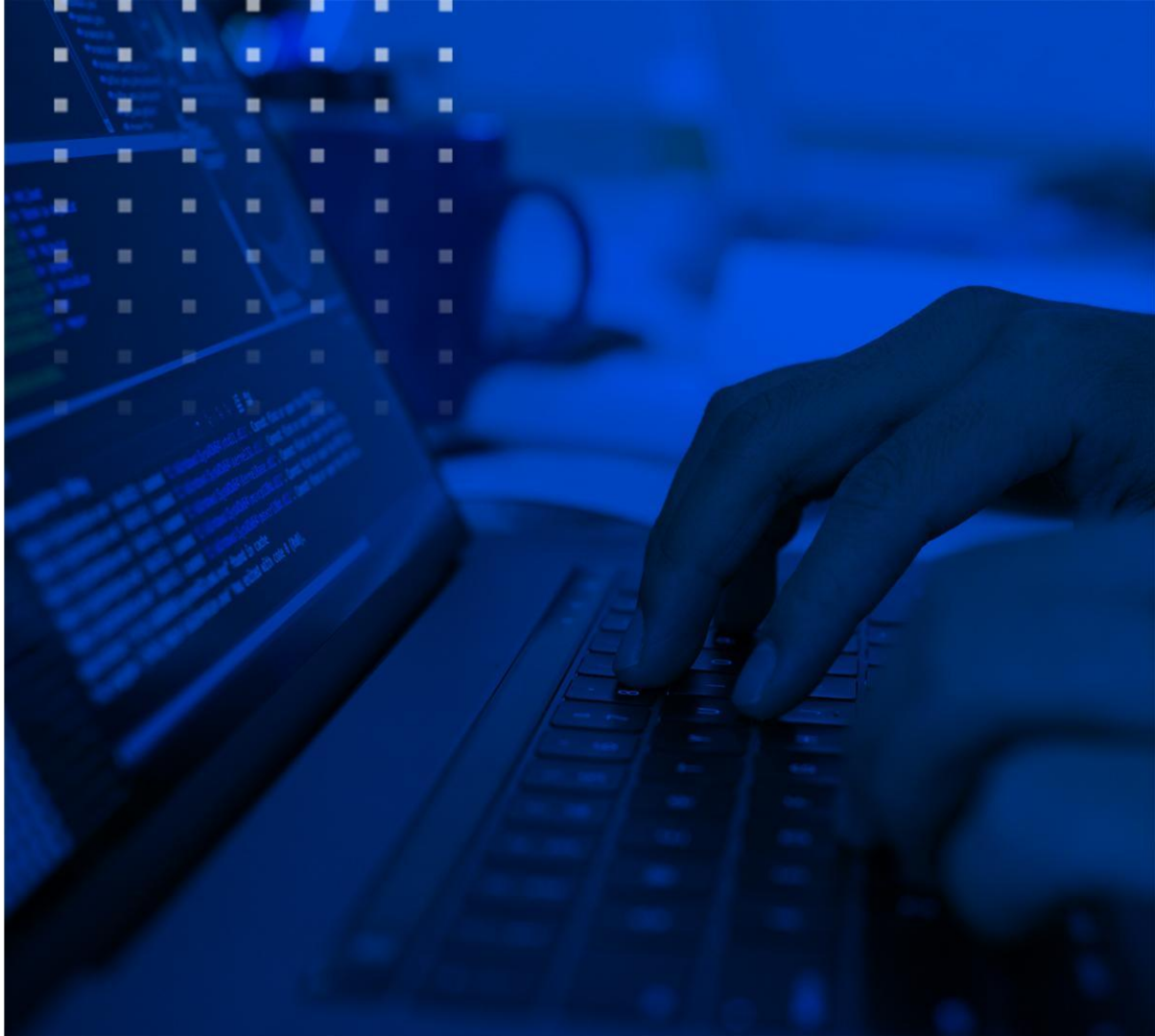
CURSO

FULL STACK DEVELOPER
NIVEL INICIAL

UNIDAD 18
JavaScript: Introducción a ReactJS



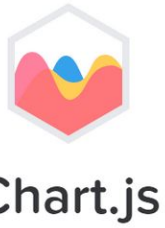
Full Stack Developer Inicial

JavaScript: Introducción a ReactJS

¿Qué es una librería?

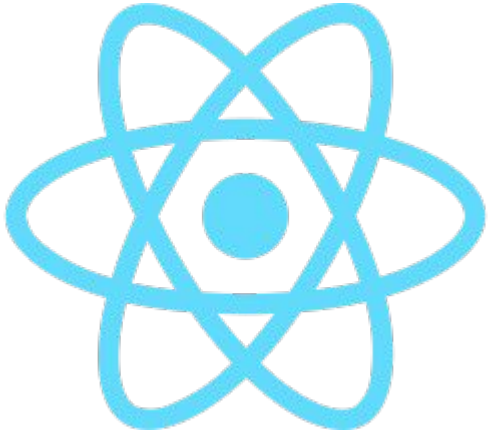
Las librerías (o bibliotecas) JavaScript son un **conjunto de métodos** que facilitan el trabajo.

Algunos ejemplos: jQuery, Chart.js, etc.



¿Qué es React?

Es una **librería** Javascript de código abierto diseñada para crear **interfaces de usuario** con el objetivo de facilitar el desarrollo de aplicaciones en una sola página. Es mantenido por Facebook y la comunidad de software libre.



Concepto básicos

- Usa un **virtual DOM** en vez del DOM real.
- Basado en **componentes**. Con estados y ciclos de vida.
- Se puede trabajar con React del **lado del servidor** con Node
- Se puede crear **aplicaciones móviles** con React Native.

NodeJS

Node.js

Node.js es un entorno de ejecución de javascript que le **permite al código JS** ser ejecutado en nuestra computadora.

NPM

Ya que Node ser un sistema modular viene prácticamente "vacío". Por lo tanto, para utilizar una funcionalidad de alguna librería publicada, deberemos instalar módulos adicionales.
Esta operación se realiza de forma muy sencilla con la herramienta npm (**Node Package Manager**).

NPM

Entonces, NPM es un **gestor de paquetes** JavaScript, que nos va a permitir instalar y desinstalar paquetes, gestionar versiones y gestionar dependencias necesarias para ejecutar un proyecto desde la línea de comandos.

Instalación de Node.js

1. Vamos a ingresar a **https://nodejs.org/en/**
2. Descargamos la última versión de Node
3. Ejecutamos el archivo de instalación y seguimos los pasos

CLI

CLI

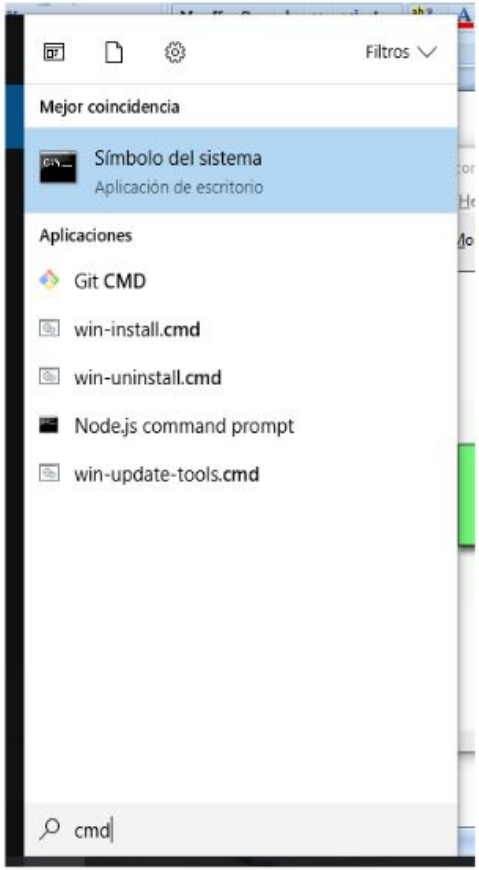
La **interfaz de línea de comandos** es un programa que nos permite dar instrucciones a algún programa informático por medio de una línea de texto simple.

CLI

En Windows podemos usar **Command Prompt** (símbolo del sistema) o **Windows PowerShell.**

Abrir la terminal / MacOS

Abrir la terminal / Windows



CLI

React tiene su propia CLI, y admite la creación de una aplicación a través del comando: **create-react-app**.

Create-react-app

¡Vamos a crear nuestra primer React App! ✨

1. Ubicarnos en el directorio específico con la terminal

```
naranjas@naranjas:~$ cd Documentos/Proyectos
```

2. Instalamos React CLI con el comando:

Opción 1:

```
npm install -g create-react-app
```

```
create-react-app <project-directory>
```

Nota: En linux y mac agregar sudo: `sudo npm install -g create-react-app`

2. Instalamos React CLI con el comando:

Opción 2:

```
npx create-react-app <project-directory>
```

```
   npx create-react-app my-react-app
```

(si escribimos . en vez de un nombre, el proyecto tomará el nombre de la carpeta contenedora)

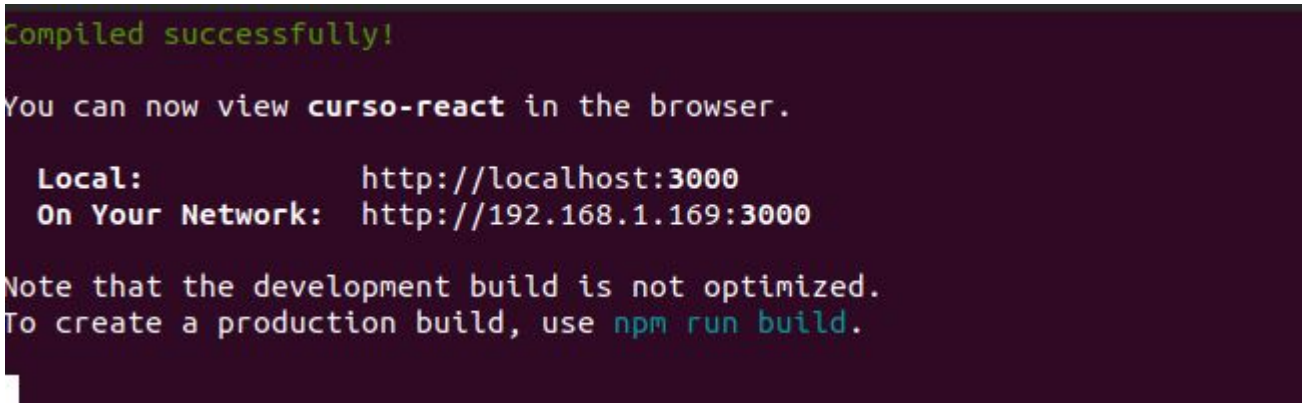
Si está OK... nos aparecerá lo siguiente

```
Success! Created curso-react at /home/naranjas/Documentos/Proyectos/curso-react
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd curso-react
  npm start

Happy hacking!
```

3. Nos **movemos** a la carpeta de nuestra app (usando cd) y la iniciamos usando **npm start**

```
npx create-react-app my-app
cd my-app
npm start
```

4. Si todo sale bien nos debería mostrar algo así:

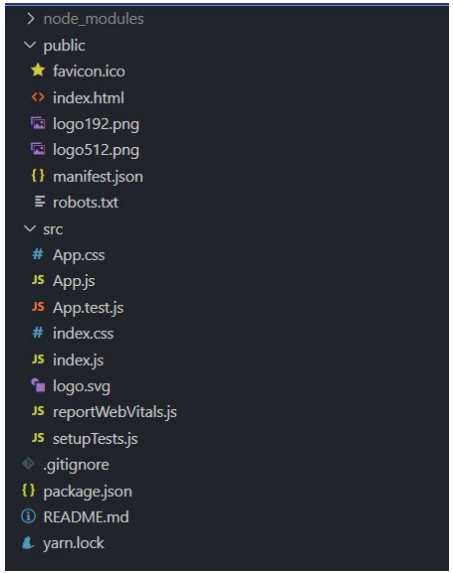


5. Ahora podremos ver nuestra app en **http://localhost:3000/**

¡Ya tenemos creada nuestra primer app React!

Repaso por la app

Archivos



Vemos que la app se compone por varios archivos base...

Vamos a ir viendo en nuestra carpeta qué son éstos archivos y cuáles vamos a utilizar.

Bootstrap

React Bootstrap

React Bootstrap es la versión **reconstruida para React** del **framework front-end más popular**

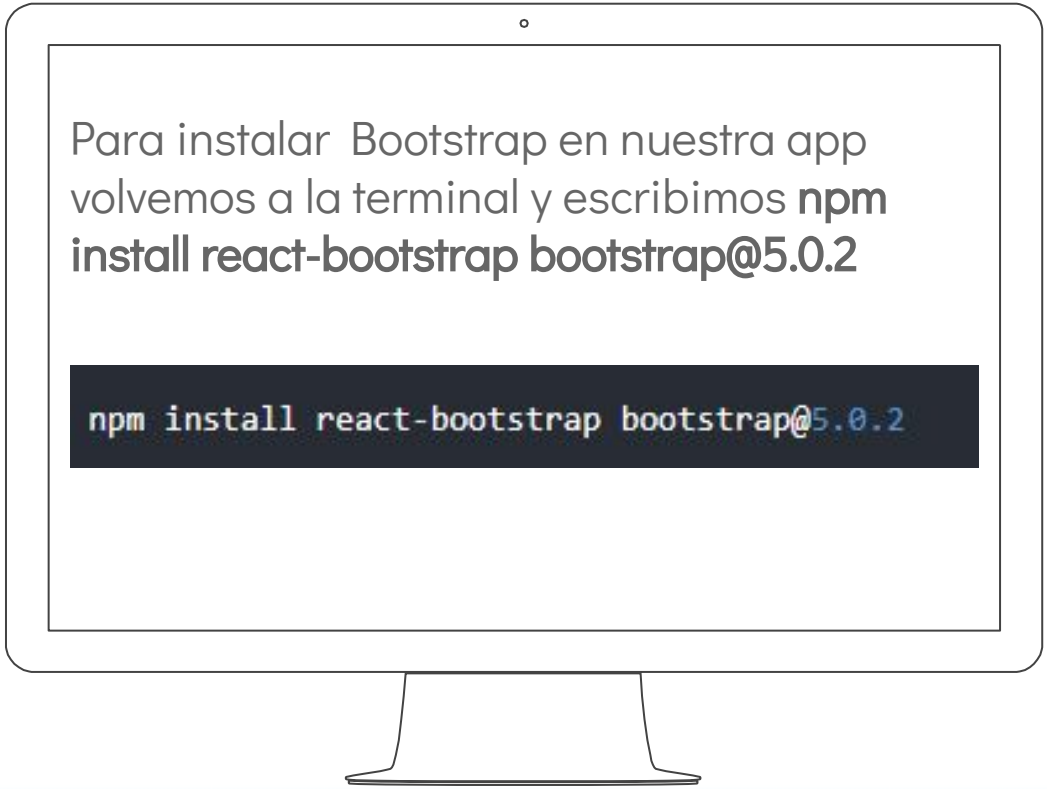
https://react-bootstrap.github.io/

Van a notar que los componentes son muy similares pero cambia la sintaxis y la forma de incluirlos.

Bootstrap

La mejor manera de consumir React-Bootstrap es a través del paquete npm con el que puede instalar npm (o yarn si lo prefiere)

(X) NO vamos a utilizar el tag <link>


Para instalar Bootstrap en nuestra app volvemos a la terminal y escribimos **npm install react-bootstrap bootstrap@5.0.2**
npm install react-bootstrap bootstrap@5.0.2

Importación de componentes

Debemos importar componentes individuales como: en react-bootstrap/Button lugar de toda la biblioteca. Al hacerlo, solo se incorporan los componentes específicos que usa, lo que puede reducir significativamente la cantidad de código que termina enviando al cliente.

```
import Button from 'react-bootstrap/Button';
```

Una vez instalado, abrí tu **app.js** y añadí algún elemento Bootstrap a tu app.

JSX frente a HTML

JSX (JavaScript XML) es una **extensión a JS** que nos permite escribir HTML dentro de nuestro código sin necesidad de representarlo como un *string* (encerrándolo entre comillas).

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

- Es muy similar a HTML, lo que aporta facilidad de escritura y comprensión.

Entonces...

- Se **parece a HTML**. No es ni una cadena de caracteres, ni HTML.
- Es una extensión que permite hacer llamadas a funciones y a construcción de objetos.
- JSX es una **extensión de Javascript**, no de React.
- Esto significa que **no hay obligación** de utilizarlo, pero es **recomendado** en la documentación oficial de React.
- Al final, JSX **se transforma en código JavaScript**.

JSX

Esto nos da algunas **ventajas** como:

- ★ Ver errores en tiempo de compilación
- ★ Asignar variables
- ★ Utilizar métodos

Algunas diferencias

- El estilo **CSS en línea** en JSX tiene una sintaxis diferente.
- El modo nombrar una **clase** en las etiquetas HTML es distinto.
- JSX puede hacer **referencia a variables** de JavaScript.

CSS en línea en JSX

HTML → `style="background-color:cyan;color:blue"`

JSX → `style={{backgroundColor:'cyan',color:'blue'}}`

CSS en línea en JSX

1. En JSX, en lugar de envolver el estilo entre comillas dobles, usamos llaves dobles, **style = {{...}},** la llave exterior indica que vamos a colocar una variable de JavaScript en esta y las segundas llaves indican un objeto JavaScript.
2. El nombre de la propiedad tiene un guión, tenemos que eliminar ese guión y cambiarlo al formato de camelCase comenzando con minúsculas, es decir, en lugar de background-color usamos **backgroundColor.**

CSS en línea en JSX

3. El valor de la propiedad se ajustará como una cadena entre comillas simples o dobles.
4. Las diferentes propiedades se separan con una coma en lugar del punto y coma.

Nombre de clase en JSX

HTML → `class="unameinput"`

JSX → `className="unameinput"`

En JavaScript moderno, la palabra clave de clase está reservada para clases de JavaScript y para evitar esa colisión usamos **className** para dar un nombre de clase en etiquetas HTML en JSX en lugar de class.

className

```
class App extends React.Component{
    render(){
        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>Login</button>
            </div>
        );
    }
}
```

Referencia a las variables de JavaScript de JSX

JSX puede hacer referencia fácilmente a variables de JavaScript, es decir, podemos usar variables o funciones de JavaScript en lugar de texto sin formato colocándolas, las primeras llaves indican la variable de JavaScript.

```
<button>{buttonName}</button>
```

o

```
<button>{getButtonName()}</button>.
```

Si queremos reemplazar el texto del botón con la función JavaScript.

```
class App extends React.Component{
    render(){

        const buttonName="Log in";

        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>{buttonName}</button>
            </div>
        );
    }
}
```

Usando la variable JS en lugar de texto sin formato para el botón.

Si queremos reemplazar el texto del botón con la función JavaScript.

```
class App extends React.Component{
    render(){

        const getButtonname=()=>{
            return "Log in";
        }

        return(
            <div>
                <label>Username:</label>
                <input className="unameinput" type="text" id="uname" />
                <button style={{backgroundColor:'cyan',color:'blue'}}>{getButtonname()}</button>
            </div>
        );
    }
}
```

Usando la función JS en lugar de texto sin formato para el botón.

Podemos asignar estilos inline con JSX, lo único que tenemos que hacer es incluirlo entre llaves ({{ }}).

Aplicar una regla CSS a algún elemento en tu app.js

SENATI