

### ¿Qué es una react?

React es una librería de JavaScript. Al ser una librería permite integrarse a un CDN. Es la interfaz gráfica de las aplicaciones.

React utiliza JavaScript por defecto y no lo usa al 100, sino que utiliza JSX el cual mezcla JavaScript y XML. Lo cual permite usar etiquetas de html y crear etiquetas propias (componentes).

React previene ataques de tipo xss

Esta basado en componentes y es declarativo

### ¿Qué son las Single page application SPA ?

Reemplazan un modelo MPA (Multy page application). Son aplicaciones web que caben en una única página, la cual muestra datos en tiempo real, sin recargar, no se satura y no hace peticiones cte a un servidor.

SPA interactúa con 2 tipos de servicios: REST y Serverless. Descentralizan completamente la lógica de la aplicación de la interfaz gráfica.

**REST:** Tu haces una petición a un servidor (Backend), este servidor responde en formato json, javascript lo reconoce de manejar nativa y esto da ventaja de que sea más fácil de trabajar o manejar.

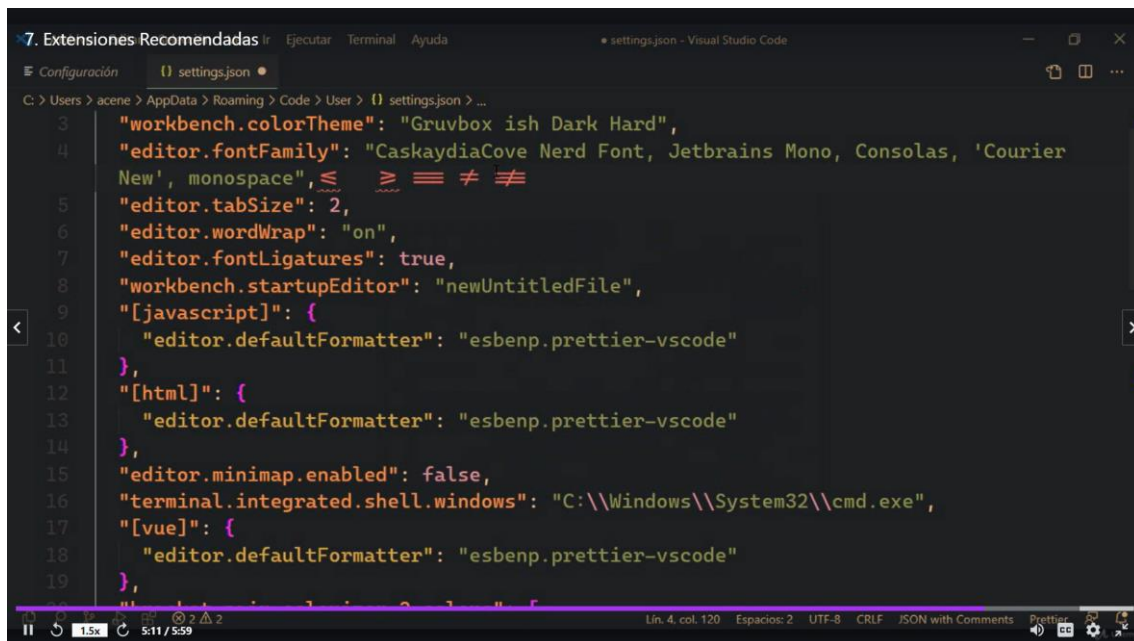
**Serverless:** Consumo servicios desde mi app.

Instalar yarn

npm i -g yarn

Extensiones útiles

- Bracket Pair Colorizer 2: Pinta paréntesis y demás
- Activitus Bar
- Css- snippets
- ES7- React/Redux/GraphQL/React-Native snippets
- Gruvbox ish
- Html css support
- Live server: Permite levantar servidor y ver cambios en tiempo real
- Prettier – code format
- Spanish Language Pack for Visual Studio Code

A screenshot of the Visual Studio Code editor showing the settings.json file. The file is open in the 'Configuración' (Settings) view. The settings include: 'workbench.colorTheme' set to 'Gruvbox ish Dark Hard', 'editor.fontFamily' set to 'CaskaydiaCove Nerd Font, JetBrains Mono, Consolas, 'Courier New', monospace', 'editor.tabSize' set to 2, 'editor.wordWrap' set to 'on', 'editor.fontLigatures' set to true, 'workbench.startupEditor' set to 'newUntitledFile', and three language-specific settings for '[javascript]', '[html]', and '[vue]' all set to 'esbenp.prettier-vscode'. The status bar at the bottom shows 'Lin. 4, col. 120', 'Espacios: 2', 'UTF-8', 'CRLF', 'JSON with Comments', and 'Prettier'.

Cambiar los símbolos visuales.

## Tipos de Variables JS

**Let:** Variables locales

**Const:** Variables constantes

**Var:** Variables Globales, estas se almacenan en memoria y permite

**Callbacks:** Funciones de flecha que van a regresar algo

Por seguridad es mejor usar const, para condiciones por eso es mejor usar operadores ternarios.

**Backticks:** Son las apostrofes (`), permiten interpolar cosas, meter variables dentro de un texto.

## Diferencia entre una promesa y un arrow function?

Las promesas retornan un objeto llamado "new Promise". Y Recibe 2 parámetros, el primero es una función de flecha que también recibe 2 parámetros llamados: resolve y reject, resolve es la respuesta que va a dar en caso la respuesta sea exitosa y el reject es lo contrario, respuesta a cuando algo sale mal.

Ajax

```
XMLHttpRequest() //Ajax
//SOAP
<xml>
</xml>
```

Instalar Axios:

1. Npm install axios
2. 

```
<script
  src="https://unpkg.com/axios/dist/axios.min.js"></script>
```

Async y Await:

Async convierte todo en promesas.

**Hooks:** Son todos los que dicen use

```
► useCallback: f (a,b)
► useContext: f (a,b)
► useDebugValue: f (a, b)
► useEffect: f (a,b)
► useImperativeHandle: f (a,b,c)
► useLayoutEffect: f (a,b)
► useMemo: f (a,b)
```

**React DOM:** es un dom virtual, realmente el código renderizado dentro de este, no se crea.

React sin jsx

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

  <script>
    ReactDOM.render(
      React.createElement(
        "ol",
        null,
        React.createElement("li", null, "Uva"),
        React.createElement("li", null, "Limon"),
        React.createElement("li", null, "Mango")
      ),
      document.getElementById("root")
    );
  </script>
</body>
```

Babel

Sirve para traducir código de js actual al antiguo.

```
<script src="https://unpkg.com/@babel/standalone/babel.min.js"></script>
```

Componente

Una parte que compone toda una aplicación, pequeñas partes de código que juntas componen la aplicación.

Forma antigua de crear un componente, usando clases

```
<body>
  <div id="root">
    <script type="text/babel">
      class Saludo extends React.Component {
        render() {
          return (
            <ul>
              <li>Audrie</li>
              <li>Annelisse</li>
              <li>Ochoa</li>
            </ul>
          );
        }
      }
      ReactDOM.render(<Saludo />, document.getElementById("root"));
    </script>
  </div>
</body>
```

Forma actual de usar componentes, forma funcional

```
<body>
  <div id="root">
    <script type="text/babel">
      const Saludo = () => {
        return (
          <ol>
            <li>Hola</li>
            <li>Hola</li>
          </ol>
        );
      };
      ReactDOM.render(<Saludo />, document.getElementById("root"));
    </script>
  </div>
</body>
```

## Estados

Los estados no se manejan de forma directa. React nos obliga a redefinir el estado con el set que destructuramos.

```
<body>
  <div id="root">
    <script type="text/babel">
      const Interpolar = () => {
        //const nombre = prompt("Cuál es tu nombre?");

        const [numero, setNumero] = React.useState(0);

        setInterval(() => {
          setNumero(numero + 1);
        }, 2000);

        return (
          <div>
            <h1>hola {numero}</h1>
            <hr />
          </div>
        );
      };
      ReactDOM.render(<Interpolar />, document.getElementById("root"));
    </script>
  </div>
```

Módulos