Configurar un proyecto de desarrollo Front End.

1. Visual Studio Code

Extensiones:

-Prettier le da formato al código por si no lo hago con el estándar o es difícil de leer.

2. Installar Node.Js

- Con Node.js viene ya npm que se utilizará para getionar librerías o códigos externos a nuestro proyecto (React, jquery, Bootstrap o cualquier otro modulo que se necesite). Esto permite ubicarlos en un solo sitio y poder acceder a sus actualizaciones de una forma más eficiente y rápida.

- Mediante npm instalar un paquetador webPck o Parcel. La función de este empaquetador es interpretar el código dentro de tu proyecto de tal forma que el navegador lo interprete, por ejemplo si se usa SASS. Parcel gana por simplicidad en configuración.

>>npx parcel index.html

Npx es otra herramienta que viene con Node. Puede ejecutar scripts que ni siquiera están en tu computador.

Genera un servidor local para visualizar los cambio s en vivo de la pagina

1. Instalar librerías

-ESlint (seguir pasos en enlace: <https://www.freecodecamp.org/espanol/news/tutorial-de-javascript-como-configurar-un-proyecto-de-desarrollo-front-end/> )

React

-Los componentes en ract son funciones js que retirnan jsx (c’odigo que luce con html pero no lo es)

>>const a = ()=> <h1>Hello world</h1>;

React divide sus librerias en dos: react-DOM( para pp webs), react-native(para aplicaciones mobiles)

La nomenclaturta de los componentes DOM-related construidos dentro de React es camelCase, primera letra en minúscula

Los componentes que creamos como desarrolladores es PascalCased con la primera letra en mayúsculas .

class => es sustituido por className,

ejemplo de componente

const greeting =()=>(

<div>

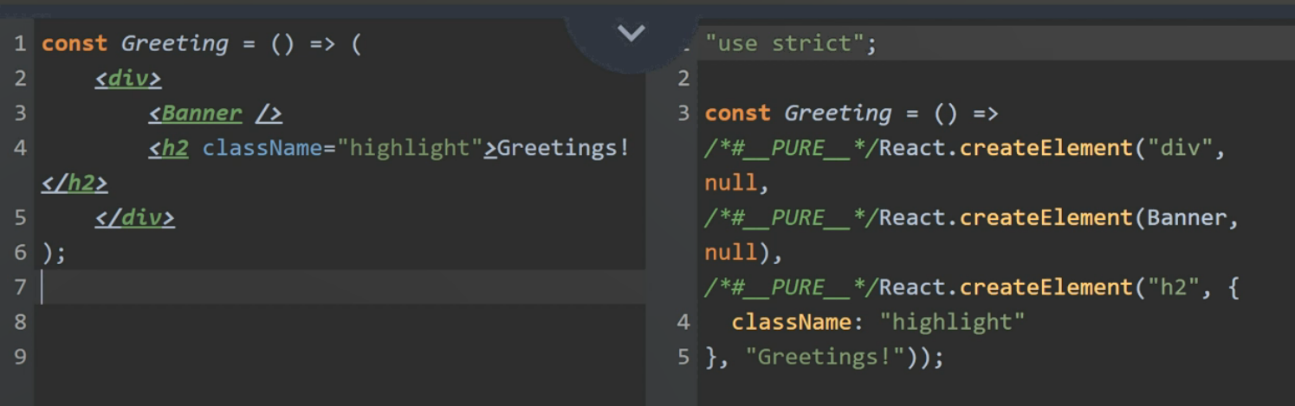
<Banner />

<h2 className=”highlight”>Greetings!</h2>

</div>

);

Cuando se intenta generar este código por detrás lo que sucede es que se llama a la librería React para que cree todos los elementos dentro del código:



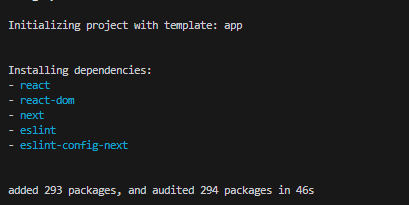
Comienzo

1. Agregar extension de react a navegador
2. En la carpeta donde voy a crear mi proyecto pongo en el terminal

>>npx create-next-app globomantics

Resaltado nombre del proyecto que voy a crear.

Esto permite crear una carpeta con el nombre del proyecto y los archivos iniciales que necesito.



>>npm run dev

Crea un host donde podremos ver nuestro proyecto durante el desarrollo es parte de las herramientas de next.js

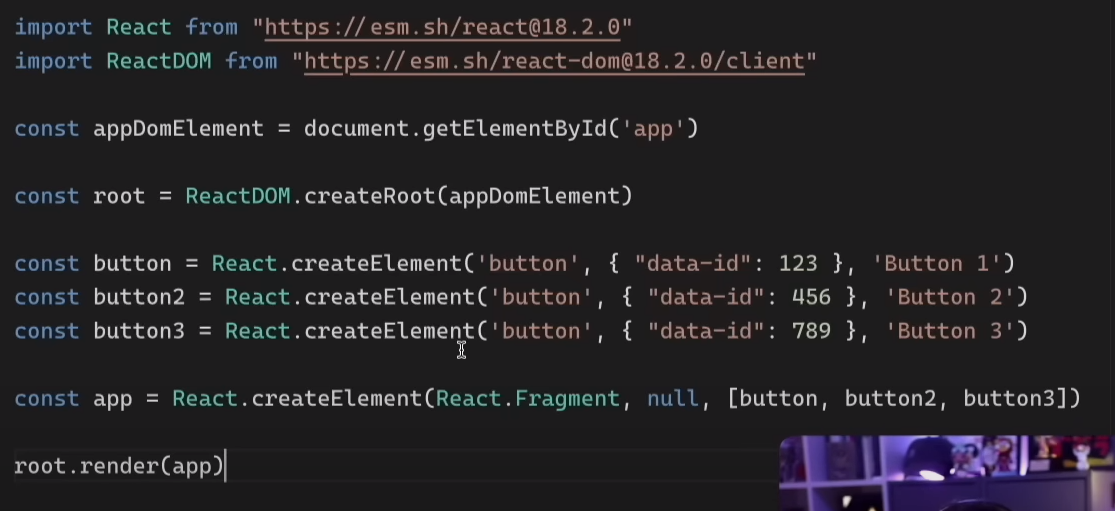
Genera las dependencias que faltan cra carpeta pages y .next (precisar para que es esto)

REACT

Es una herramienta que sirge para simplificar la creación de interfaces de usuarios. Dentro de las ventajas destaca que evita los ataques, esto se debe a la forma en que se estructura su código. Nace en 2011 pero no fue hasta 2013 que se convierte en código abierto.

Es la herramienta que utiliza Meta (Facebook, Instagram, Whatssap)

Código React:

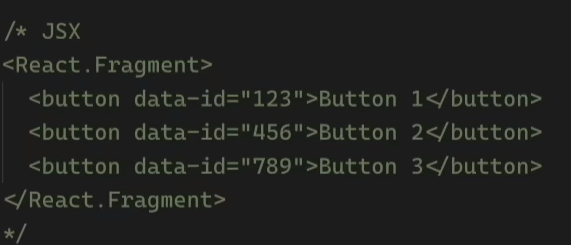


-React.Fragment genera un empaquetado para renderizar todos los botones juntos sin generar un nivel mas en el DOM.

-React.createElement(), se configura con tres parámetros, **nombreElemento**, **Atributo** (null en caso de no haber), **innerHTML** del elemento.

Con esta sintaxis controlo el DOM y creo los elemento a través de react pero aún sigue sin ser óptimo (sobre todo al crear estructuras más complejas) de ahí que se utilice JSX que es una extensión de ECMAscript basada en XML

El código superior puede ser escrito como



La interpretación de este código y llevarlo al de react lo hacen Herramientas como **babel** (<https://babeljs.io/> ) y **SWC** que no son mas que transpiladores en el que pasas el código en JSX y lo transpila a React. Estas herranmientas son las que utilizan algunos empaquetadores de aplicaciones web (alistan el proyecto para poderlo ver como un todo) como **Webpack**, **Next.js o Vite.**

Se puede ver lo difícil que se hace leer React cuando es algo más complejo

|  |  |
| --- | --- |
| JSX | React |
| <div>  <button>123</button>  <p><strong>Mucho mas dificil</strong></p>  </div> | /\*#\_\_PURE\_\_\*/\_jsxs("div", {  children: [/\*#\_\_PURE\_\_\*/\_jsx("button", {  children: "123"  }), /\*#\_\_PURE\_\_\*/\_jsx("p", {  children: /\*#\_\_PURE\_\_\*/\_jsx("strong", {  children: "Mucho mas dificil"  })  })]  }); |

Estilos en línea

<span style={{display:'flex', color:'#fff', alignItems:'center'}}></span>

**Opciones para empaquetadores web**:

-Create React App (forma official, pero va más lenta)

-WebPack (Es de las más utilizadas y con mas doc foro pero va lenta en comparación con otras opciones)

-Vite (Va bien rápido)

**Notas Importantes:**

Los nombres de los módulos que yo cree se escriben con ParselCase de lo contrario react renderizará en el DOM como si fueran elementos de HTML.

Creación de proyecto desde cero con vite como empaquetador web

1. Dentro de la carpeta seleccionada en el terminal de VS Code

(tener preparado VSCode <https://www.freecodecamp.org/espanol/news/tutorial-de-javascript-como-configurar-un-proyecto-de-desarrollo-front-end/>)

Las siguientes líneas generan una carpera **Projects** que contendrá todos los proyectos con los que iré aprendiendo. El primer proyecto que crearé es **FirstProject**

Inicializo npm en la carpeta raíz (fuera de la carpeta projects para instalar ahí los módulos que quiera compartir con todos los proyectos)

**>>npm init -y**

**>>mkdir projects**

//esta linea crea una carpera de proyectos dentro de la ruta en la que estoy trabajando (es opcional)

**>>cd project**

//para posicionarme dentro de la carpeta que recién he creado

**>>npm create vite**

//responder las preguntas que hace (nameProject, framework, variant)

//A continuación la consola te indica que códigos tengo que dar después que son

**>>cd NameProject**

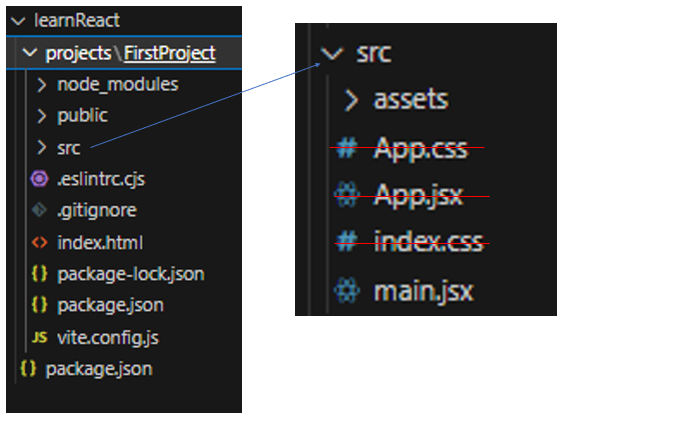
**>>npm install**

//Instala las dependencias del proyecto

**>>npm run dev**

//Inicializa proyecto y te da la url donde puede verse los cambios

1. Acceder a las carpetas generadas y borrar lo que no necesito de momento para crearlo desce cero. (Esto es mientras estoy aprendiendo para agregarlo por mi cuenta)



1. Preparar el main.js

Main.js es el punto de partida de mi app, mi fichero principal, ahí importaré todos los demás ficheros incluso los de css.

Código default

import React from 'react'

import ReactDOM from 'react-dom/client'

import App from './App.jsx'

import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(

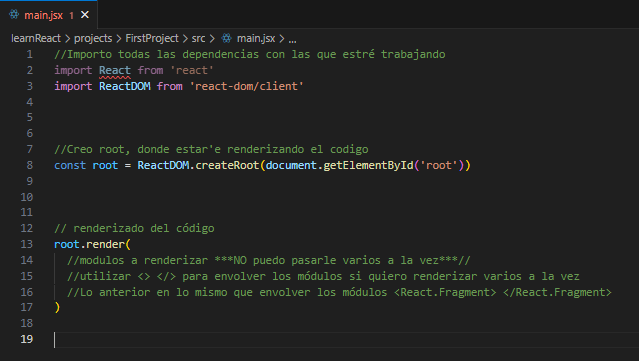
  <React.StrictMode>

    <App />

  </React.StrictMode>,

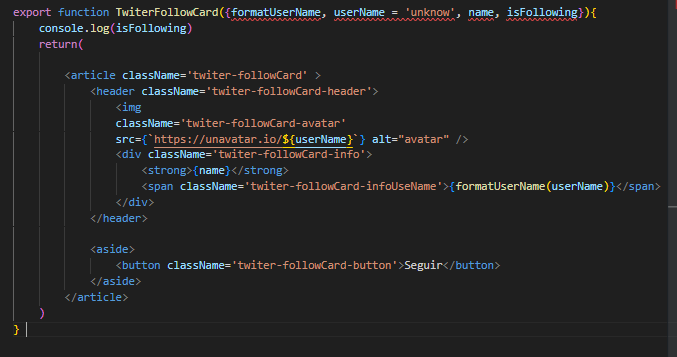
)

Código modificado



1. Crear un módulo, darle estilo y renderizarlo

**-Modulo**

****

* 1. **Agregar modulo al proyecto al proyecto**

-Crear un archivo para el módulo **TwitterFollowcard.jsx**

-Crear archivo para el CSS **TwitterFollowcard.css**

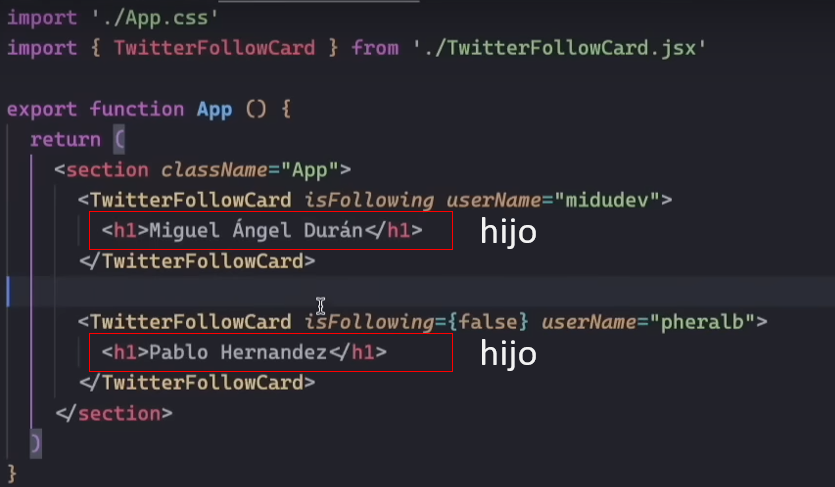
-Importar a este archivo del CSS en el del módulo.

import './TwitterFollowcard.css'

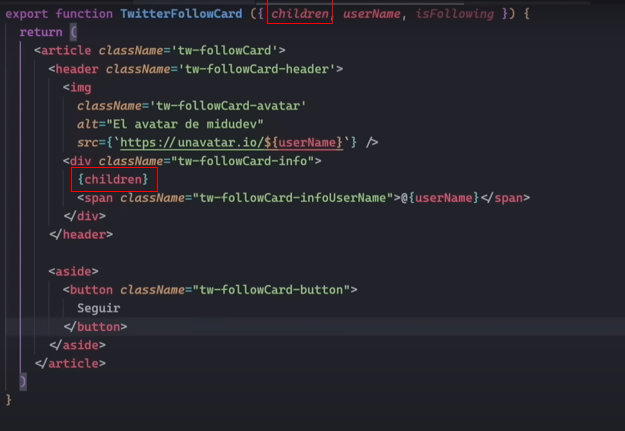
-Importar módulo en **main.jsx**

import { TwiterFollowCard } from './TwiterFollowCard.jsx'

1. **Cosas importantes de React**
   1. **Prop Children**

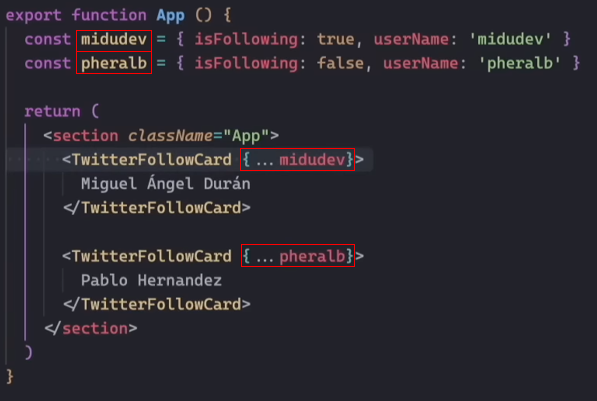
****

Se trata de una prop que le puedo pasar a TwitterFollowCard para recuperar el contenido que esté envolviendo. En este caso *<h1>Miguel Angel Durán</h1>* y *<h1>Miguel Angel Durán</h1>* respectivamente y por separadopara renderizarlo en el DOM

****

**El children seleccionado recupera el contenido que envuelve en este caso Miguel Angel Durán**

* 1. **Pasar todas las propiedades de una vez, aunque suele ser mala práctica puede tener sentido cuando sean muchas propiedades.**

****

* 1. **Comentarios dentro de codigo jsx. No es una buena pr’actica pero por si es necesario.**

**{/\* comentario \*/}**

* 1. **React solo actualiza la parte del DOM que se cambió pero eso no quiere decir que no renderice elementos aun cuando sus datos no hayan cambiado. Si cambio un modulo todos los hijos de ese módulo se vuelven a renderizar pero al DOM solo llega lo que fue cambiado realmente.**

**Hay formas de evitar que todos los hijos se rendericen cdo hay un cambio**