npm install, npm i 🡪 Instalar dependecias

npm start 🡪 inicia un servidor desarrollo

package.json 🡪 partes:

* Dependencies: dependencias y versión del proyecto
* Scripts: Diferentes funciones que puede ejecutar el proyecto
  + Start: permite ejecutar el proyecto npm start 🡪 Local
  + Build: generar una versión del proyecto que se pueda desplegar a producción 🡪 Producción
* eslintConfig: Para que la console de mensajes, recomendaciones y consejos
* browserslist: A que versión de egma javaceript debe transpirar el proyecto.

.gitinore 🡪 ignorar carpetas y archivos que no se suben al repositorio

Carpeta Public:

* index.html 🡪 la estructura base de la aplicación
  + dentro de body el elemento **<div id=”root”></div>** el cual es donde reat js para renderizar todo el código. Reat js inserta el código en este elemento.
  + Esta etiqueta **<noscript>You need to enable JavaScript to run this app.</noscript>** sale un aviso para que el usuario active el javascript
  + Manifest para trabajar con progresive web app se necesita este archivo para que la aplicación sirva offilne y separa que colores usar
* Robots.txt 🡪 sirve para indicar a lo que se le puede hacer web scrapping y a lo que no

Carpeta Src, carpeta principal:

* Index.js 🡪 importar react, estilos y el archivo App

import React from 'react';

import ReactDOM from 'react-dom/client';

import './index.css';

import App from './App';

* const root = ReactDOM.createRoot(document.getElementById('root')); 🡪 Se crea un root para react con el element con root del html
* root.render(<App />); 🡪 Todo lo que este dentro del archivo de App se va a renderizar dentro del root del html, es decir App lo esta ingresando al Div Html con id root de index.html

App.js 🡪

function App() {} 🡪 por nomenclatura todo lo que inicie con una mayúscula se supone que es un componente

El return son los elementos que retornan al componente App

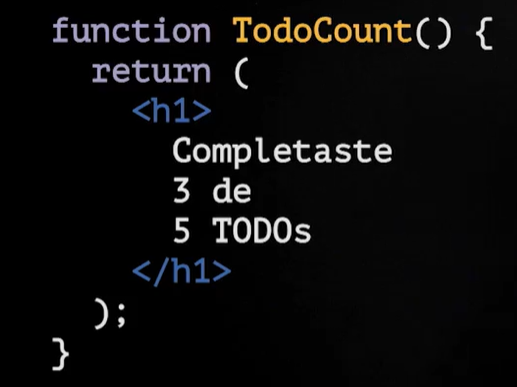
**El lenguaje dentro del return del componente es JSX, sintaxis para combinar javascript con xml, es una sixtasis parecida a html para ir sabiendo en que va a convertir cuando ya se renderizar.**

Los elementos de react comienzan con minúsculas y estos se van a convertir en etiquetas HTML

Insertar un componente dentro de otro componente, poner dentro de App

<TodoItem />

El componente sirve para no repetir el código, encapsular la funcionalidad y usarla nomas llamando el componente

A black screen with blue text

Description automatically generated

A computer screen with text and symbols

Description automatically generated se asocial al index.js de la siguiente manera

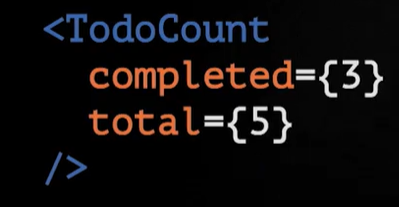
A computer screen with text and symbols

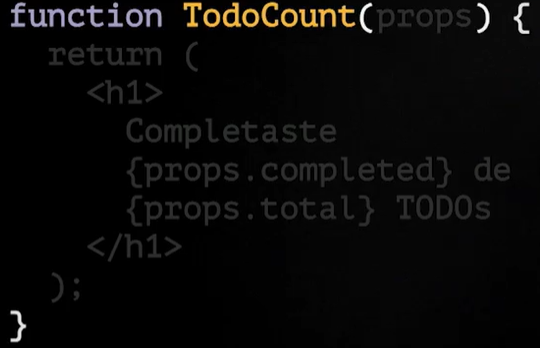
Description automatically generated Se puede rendizar el mismo componente varias vecesA white background with black text

Description automatically generated

A screen shot of a computer code

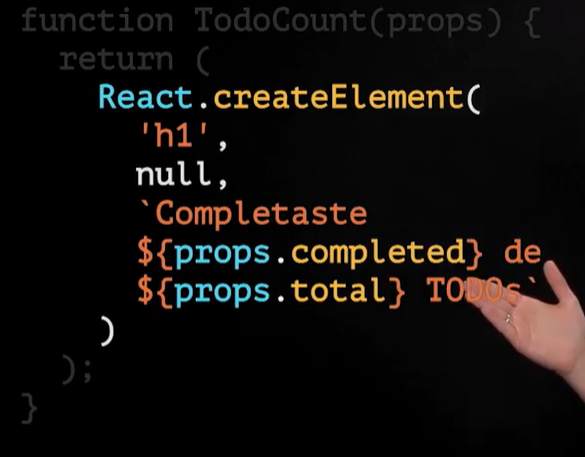
Description automatically generated Para hacer dinamico el componente se reciben props o propiedades, la cual van a reemplazar los valores

 se envían de la siguiente manera

 La fuction es el componente

Los componente reciben propiedad, los componentes no terminazan rederizando nada en le html , los elementos son encapsulados por los componentes, viven dentro del componente, son los elementos lo que se terminan renderizando

El JSX se parece al html a propósito para que se pueda usar de forma comoda, estoy tiene que pasarse por herramientas con Babel para que se convierta a java script de verdad

 esto seria con java script

A person's hands touching a blue rectangular object

Description automatically generated

Va a pasar por la fábrica de render y va dar como resultado las etiquetas html y código java script, y así crear la pagina web.

Si el componente va a manejar contenido interno de otros componentes no se cierra directamente si no que se maneja con apertura y cierre es decir

<TodoList>

<TodoItem />

<TodoItem />

<TodoItem />

</TodoList>

Exportar el componente de la siguiente manera:

Export {TodoCounter}, es decir exportar un objeto que adentro tenga el componente de TodoCounter y así evitar usar el export default es puedo permiter exportar con nombres incorrectos

Esto se llamar export nombrados