Creando primer componente

En esta sección, vamos a crear un componente reutilizable en React, además de unas etiquetas de tipo fragmento que van a englobar todo el contenido HTML que va a retornar el componente.

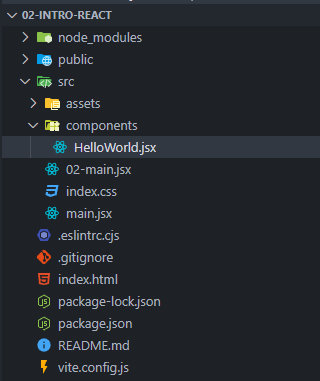
Definir un nuevo componente en React

Para crear un nuevo componente en React, primero necesitamos crear un archivo separado en formato JSX o JS. Actualmente al ejecutar la aplicación debe mostrar una pantalla en blanco en el navegador, dado que no hay contenido en el archivo **main.jsx**.

Es una buena práctica organizar nuestro proyecto en carpetas dentro de la carpeta **src**, por lo cual vamos a crear una carpeta llamada **components** dentro de la carpeta raíz **src**. Es normal que el icono de la carpeta sea diferente.

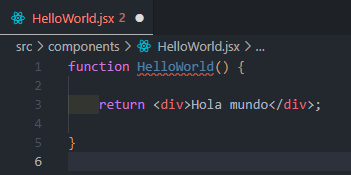
Dentro de la carpeta **components**, crearemos un nuevo archivo para nuestro componente llamado **HelloWorld.jsx**. Es importante tener en cuenta que los archivos JSX, que son componentes, comienzan con mayúsculas y utilizan la notación CamelCase para palabras compuestas.

Proceda a eliminar los archivos **App.css** y **App.jsx**, pues no se van a utilizar en este proyecto.



Definir una función exportable en el componente

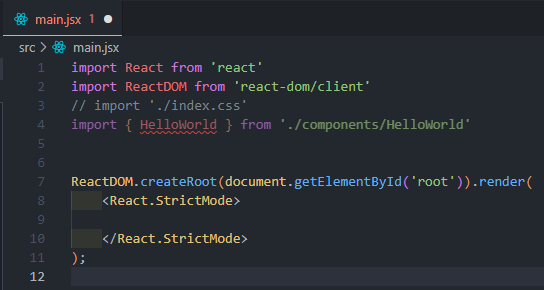
Todo componente en React es una función con nombre (por lo general, el mismo nombre del archivo). La función devuelve elementos HTML.



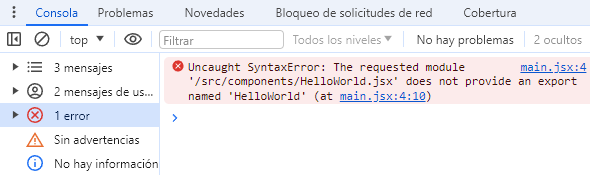
**Nota:** Luego de crear un archivo **jsx**, evite cambiar el nombre de ese archivo para no tener inconvenientes.

Para utilizar esta función, debemos importarla en nuestro archivo **main.jsx**. Dado que está en una carpeta, primero especificamos la ruta de la carpeta raíz **src** con **./**, luego la carpeta **components**, y finalmente el nombre del componente sin la extensión.

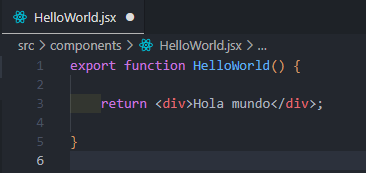
También se elimina la importación de la hoja de estilos **index.css**.



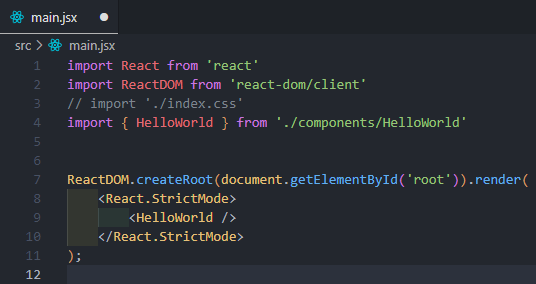
Al guardar el archivo **main.jsx**, podemos ver un error en la consola del navegador que indica que el componente **HelloWorld** no está siendo exportado.

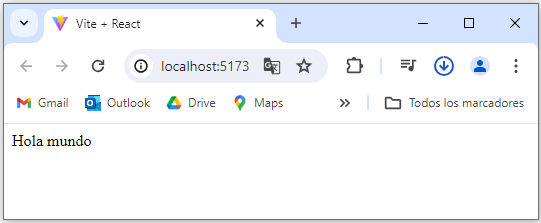


Para solucionarlo, agregamos la palabra clave **export** antes de la declaración de la función **HelloWorld** en el archivo **HelloWorld.jsx**.



Ahora podemos visualizar el componente **HelloWorld** en el navegador colocando **<HelloWorld />** (o **<HelloWorld></HelloWorld>**)dentro del método **render**, dentro de **<React.StrictMode>**.



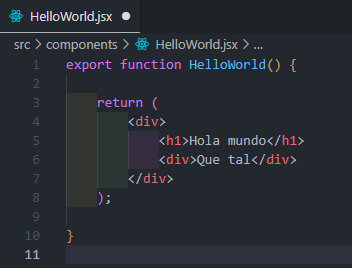


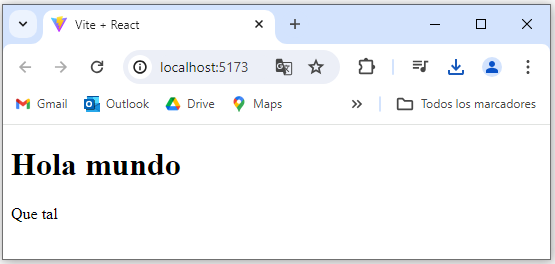
Componentes y fragmentos

Si queremos incluir unas etiquetas **<h1>** dentro del componente **HelloWorld**, además de las etiquetas **<div>**, envolvemos todos los elementos HTML que va a retornar el componente entre paréntesis.

Es importante tener en cuenta que si no lleva paréntesis, significa que solo admite una sola línea de código. Sin embargo, un componente puede tener varias líneas de código, pero solo puede tener un elemento principal.

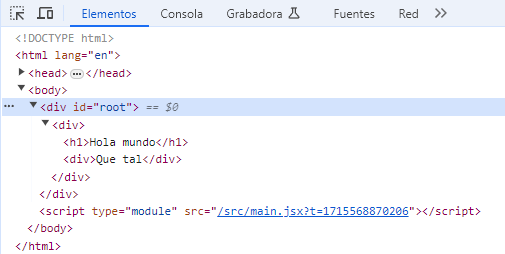
Normalmente, se define un **<div>** principal que engloba a todos los elementos HTML.



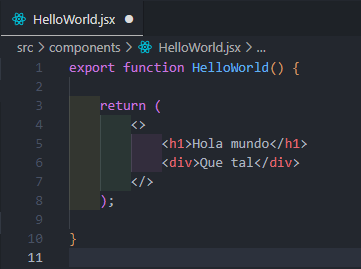


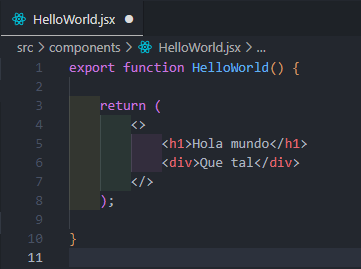
Problemas con los estilos CSS

Sin embargo, si utilizamos un elemento **<div>**, podríamos tener problemas con los estilos CSS que afecten al **<div>** principal, ya que automáticamente va a formatear el fragmento principal o raíz que contendrá los demás elementos del componente.

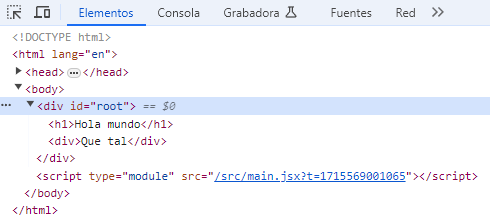


Para evitar que el estilo se aplique al **<div>** root o principal del componente, podemos utilizar un fragmento; este último es un componente que no tiene nombre y se define con **<>** y **</>**.





Ten en cuenta que este fragmento no es un **<div>**. Cuando se renderiza en el navegador, esa etiqueta desaparece. Se puede comprobar en el código fuente de la página que se muestra en el navegador.



Este enfoque resulta más limpio y evita interferencias con los estilos. Es por ello que se recomienda usar fragmentos.

El elemento <Fragment>

Es una característica de React que se utiliza como un envoltorio para agrupar múltiples elementos hijos sin necesidad de añadir un nodo adicional al DOM.

Por ejemplo, si tienes varios elementos que quieres devolver desde un componente en React, pero no quieres agregar un contenedor adicional al DOM, puedes utilizar las etiquetas **<Fragment></Fragment>** para envolverlos o las etiquetas **<></>** para definir el fragmento.

Otro punto es que al escribir **<Fragment>**, automáticamente React importa el componente **Fragment**.

