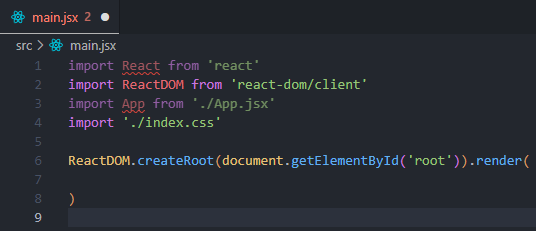
Manejo del DOM en React

En esta sección, vamos a comenzar a manipular el DOM en React, una tarea esencial para crear interfaces de usuario dinámicas e interactivas.

Manipulación del DOM en React

React trabaja por detrás del DOM. Cuando tienes un componente con una etiqueta HTML, React la convierte a JavaScript y manipula el DOM para renderizarla. React simplifica trabajar con vistas y plantillas usando etiquetas HTML, que se convierten en código JavaScript detrás de escena, formando un árbol de estructura HTML y renderizándolo en el navegador.

Primero procede a limpiar el código en **main.jsx** (borra todo el contenido definido dentro del método **render**) para comenzar con un ejemplo básico de manipulación del DOM utilizando React.



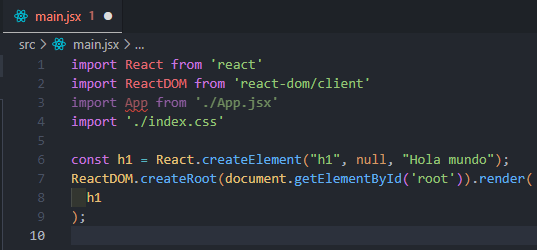
La función **render** es un método esencial en React que se utiliza para renderizar un elemento de React en un nodo DOM específico.

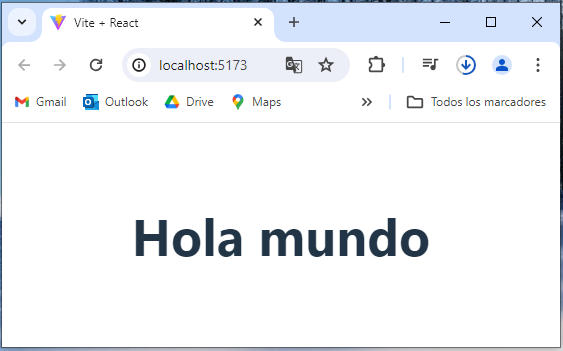
Crear un componente HTML

Para crear un componente HTML utilizando React, se utiliza el método **createElement** de la biblioteca de **React**. Este método admite tres argumentos:

* **Nombre del elemento HTML**: El primer argumento es el nombre del elemento HTML encerrado en comillas.
* **Atributos del componente**: El segundo argumento es un objeto que lleva las configuraciones y atributos del componente HTML. Si no se necesita ningún atributo, se puede dejar como **null**.
* **Contenido del elemento**: El tercer argumento es el contenido que tendrá el elemento. Puede ser un texto, una variable o incluso otro elemento que se irá anidando.

A continuación, se muestra cómo crear un componente **<h1>** con el texto **"Hola mundo"**. También se llama a la constante **h1** dentro del método **render** para mostrarlo en el navegador.

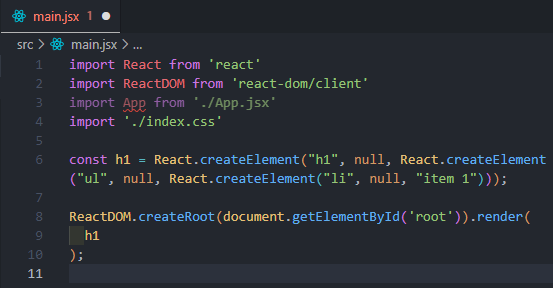


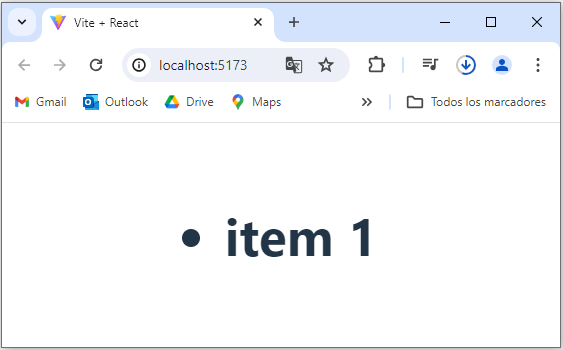


Elementos HTML anidados

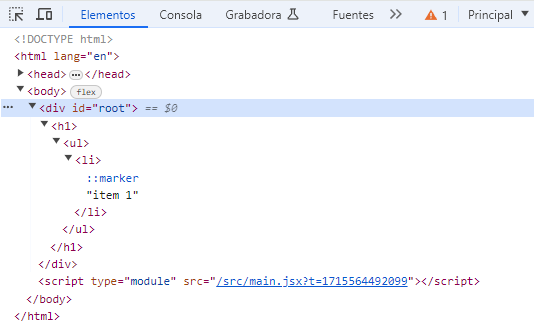
En React, puedes anidar elementos HTML de manera similar como se hacía en HTML puro, pero utilizando la función createElement de React para cada elemento anidado.

A continuación, se muestra cómo crear un componente **<h1>** que contiene un **<ul>** con un **<li>** anidado:





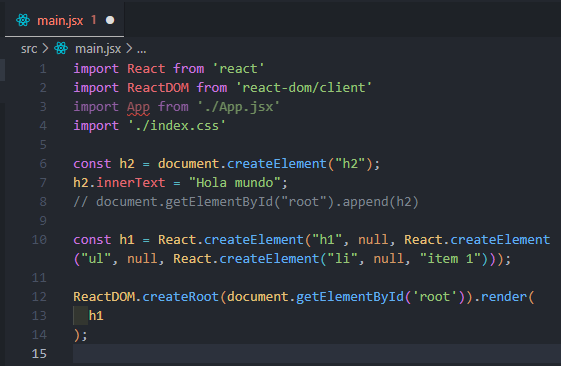
Al renderizar este componente en el navegador, verás que se muestra un encabezado **<h1>** que contiene una lista **<ul>** con un ítem **<li>**. Esto se puede comprobar abriendo la consola de Chrome, en la sección Elementos:

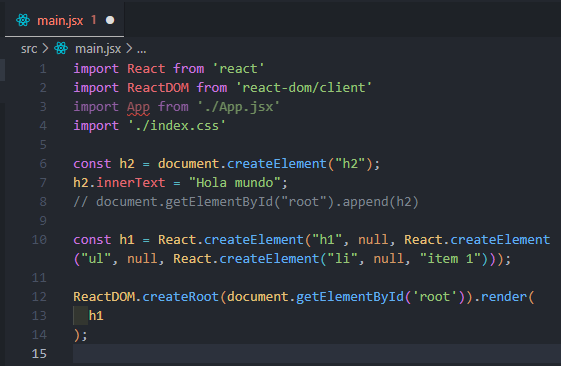


React vs JavaScript para crear un elemento HTML

Existen diferentes formas de crear elementos HTML en una aplicación web. Anteriormente se había utilizando JavaScript puro, y la otra más común actualmente es utilizando la función **createElement** de React.

A continuación, se muestra cómo crear un elemento **<h2>** con el texto **"Hola mundo"** utilizando JavaScript puro. Es importante destacar que, debido a que se trabaja con React para renderizar los componentes, el elemento **<h2>** creado con JavaScript puro no se mostrará en el navegador:





En el ejemplo mostrado:

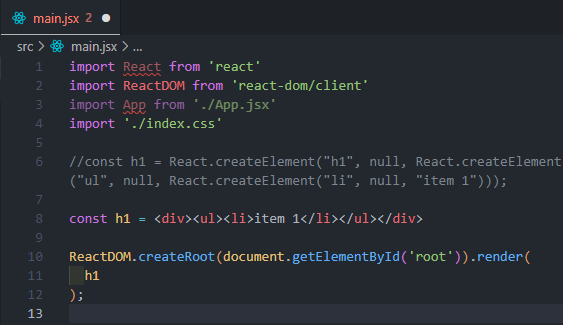
* El componente **<h2>** se podría crear utilizando JavaScript puro con **document.createElement** y el texto se puede establecer utilizando **innerText** (no funciona en archivos jsx).
* El componente **<h1>** se crea utilizando la función **createElement** de React.

Ambos enfoques logran el mismo resultado en el código, pero el uso de React simplifica la creación y el anidamiento de elementos HTML, haciendo que el código sea más legible y mantenible.

Utilizar las etiquetas HTML en React

Hacer el uso de las etiquetas HTML en React es más sencillo y legible gracias al uso de JSX. JSX permite escribir código que parece HTML pero que se convierte en llamadas a funciones de JavaScript en tiempo de compilación.

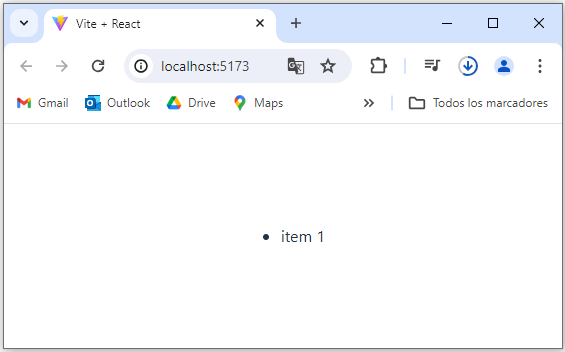
A continuación, se muestra cómo crear un elemento anidado con las etiquetas **<div>**, **<ul>** y **<li>** respectivamente; y su contenido utilizando JSX en lugar de la función **createElement** de React:



En el ejemplo mostrado:

* Se utiliza JSX para definir el elemento **<div>** y su contenido.
* JSX se convierte en llamadas a funciones de JavaScript en tiempo de compilación, lo que hace que el código sea más sencillo y legible.

El resultado final en el navegador será el mismo que en el ejemplo anterior, pero el código es mucho más claro y fácil de entender gracias al uso de JSX.



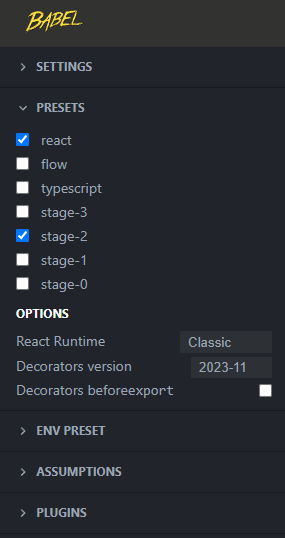
Convertir etiquetas HTML en JSX con Babel JS

El uso de etiquetas HTML directamente en React puede parecer extraño al principio, ya que parece que se están mezclando dos lenguajes diferentes. Sin embargo, esto es posible gracias a JSX, una extensión de JavaScript que permite escribir código que se parece a HTML pero que se convierte en llamadas a funciones de JavaScript durante la compilación.

Babel es una herramienta que nos permite transformar el JavaScript moderno, incluido el JSX, en un código que pueda ser interpretado por todos los navegadores.

Para probar y convertir tu código JSX en JavaScript utilizando Babel, puedes visitar BabelJS.io (<https://babeljs.io/>). Sigue estos pasos:

1. Ve a la opción **Try it out** (se encuentra en la barra de menú) para probar y convertir el JSX y las etiquetas HTML.
2. En la sección **Preset**, marca la opción **React** y en **React runtime**, elige **Classic**.



Se muestra cómo se ve el código JSX y su equivalente en JavaScript puro utilizando Babel:

Definir un elemento <container> en JSX:

<container>

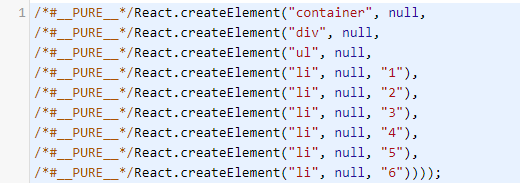
</container>

/*#__PURE__*/React.createElement("container", null);

Anidar elementos en JSX:

<container>
  <div>
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
      <li>6</li>
    </ul>
  </div>
</container>

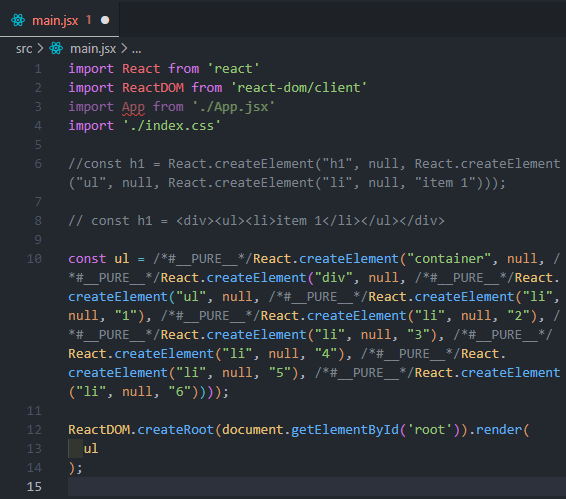
<container>
  <div>
    <ul>
      <li>1</li>
      <li>2</li>
      <li>3</li>
      <li>4</li>
      <li>5</li>
      <li>6</li>
    </ul>
  </div>
</container>

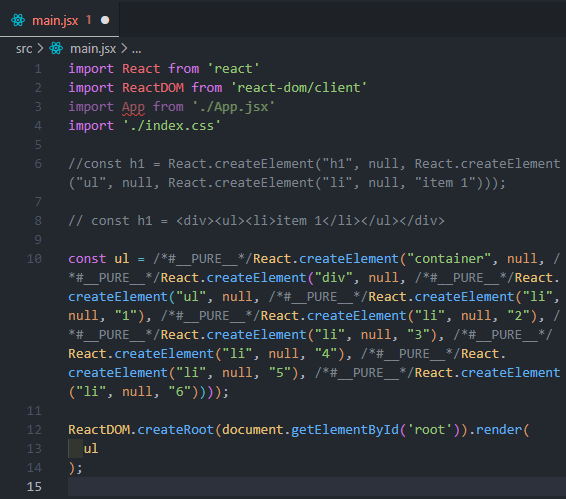


En este ejemplo:

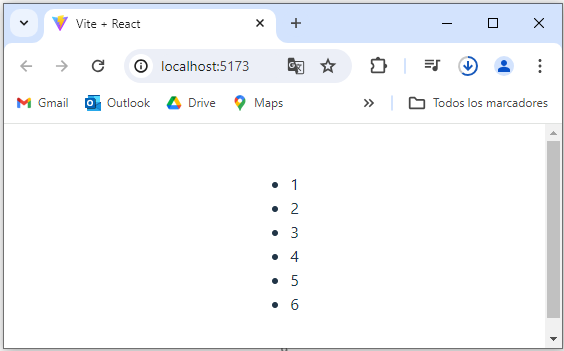
* Se utiliza JSX para definir los elementos y su anidamiento.
* Babel se encarga de transformar el código JSX en llamadas a funciones de JavaScript (**React.createElement**) que manipulan el DOM para crear los elementos HTML.

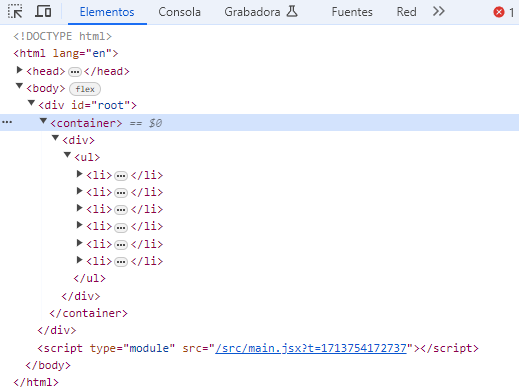
Para aplicar este ejemplo, puedes copiar y pegar el código de React proporcionado como resultado de la constante **ul** y especificarlo dentro de **render**.

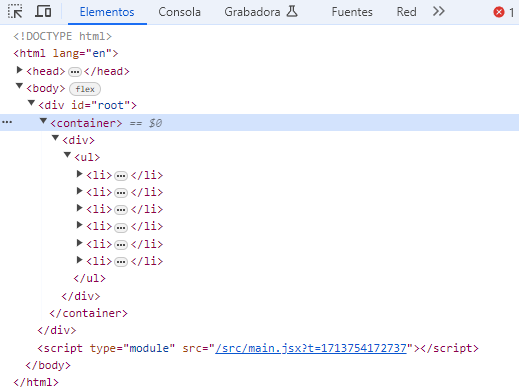




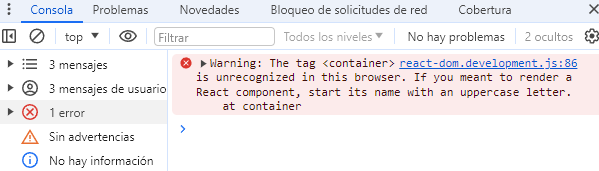
El resultado final en el navegador será el mismo en ambos casos, pero el uso de JSX hace que el código sea más claro y fácil de entender, especialmente cuando se trabaja con anidamiento de elementos.

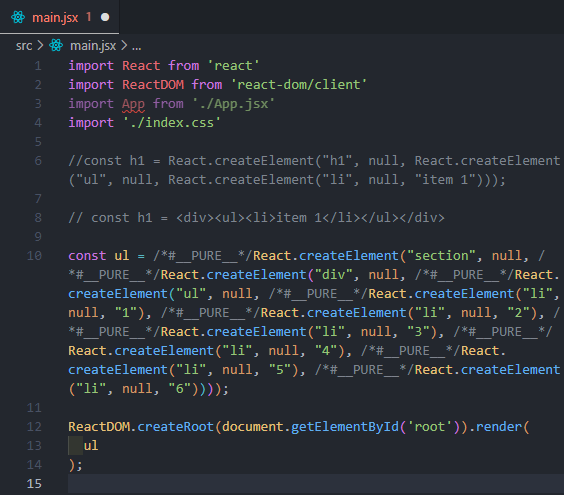






Asimismo se puede notar un error en la consola, el cual indica que no se reconoce el elemento **<container>** en HTML, para solucionarlo se reemplaza con un elemento **<section>**. Para que no de el error.





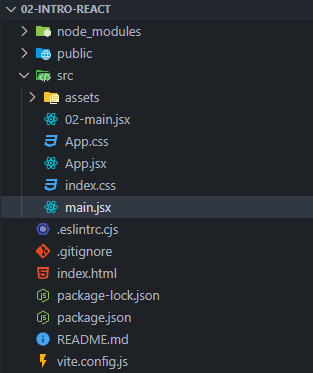
Antes de pasar al siguiente tema

Realiza estas acciones en Visual Studio Code:

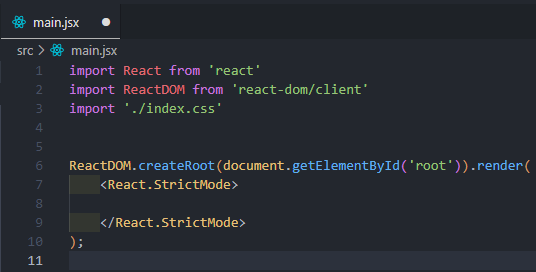
1. Abre el panel izquierdo y busca el archivo **main.js**.
2. Haz clic derecho en el archivo y selecciona **Cambiar nombre**.
3. Cambia el nombre del archivo a **02-main.jsx** y presiona **Enter**.
4. Ahora, crea un nuevo archivo, dentro la carpeta raíz **src**, haciendo clic en el botón **Nuevo archivo** en la parte superior del panel izquierdo.



1. Nombre el nuevo archivo como **main.jsx** y presiona **Enter**.



1. Copie y pegue todo el código fuente contenido en **02-main.jsx** a **main.jsx**. Elimine todo el código relacionado a la manipulación de DOM.
2. Agrega las etiquetas **<React.StrictMode></React.StrictMode>** dentro del método **render** de tal manera que el código quede así:



A partir de este momento, se realizarán cambios en el archivo **main.jsx** que has creado.