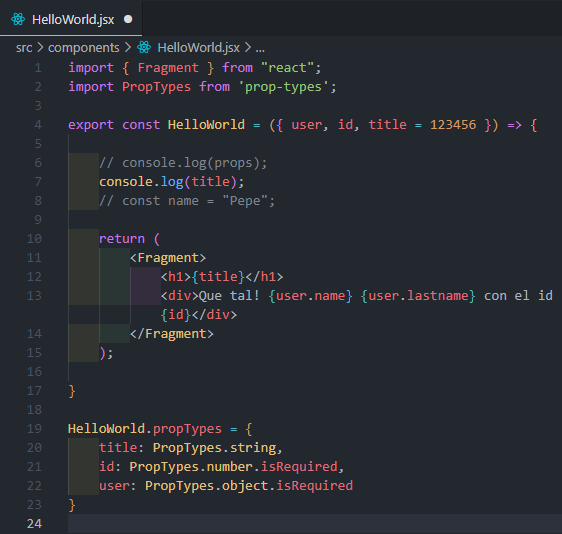
Las propiedades por defecto o Default Props

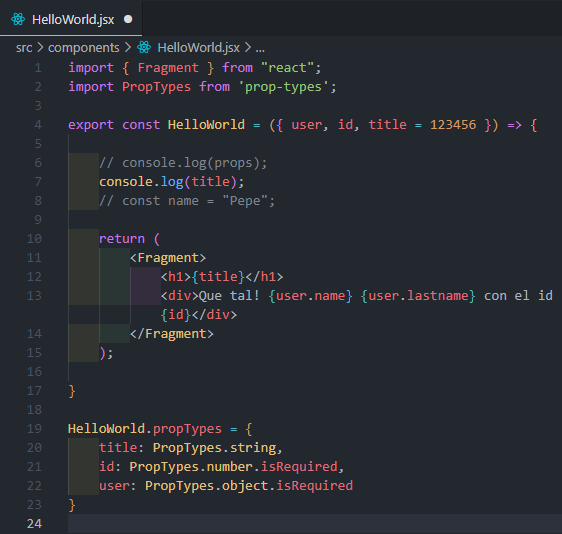
En esta sección vamos a establecer los valores por defecto para las propiedades o props definidas en un componente con defaultProps, también veremos algunos problemas que suelen pasar al definir los defaultProps.

Valor por defecto en el argumento de la función

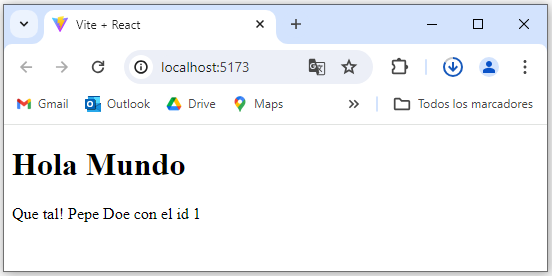
Una práctica común en React es proporcionar valores por defecto para las propiedades en caso de que no se pasen desde el componente padre. Esto garantiza un comportamiento más predecible y reduce la posibilidad de errores.

Primero, eliminamos la validación **isRequired** del **propTypes** para **title** y lo definimos como un tipo de dato string. Luego, establecemos un valor por defecto de tipo numérico para **title** en el parámetro de la función principal del componente **HelloWorld**.

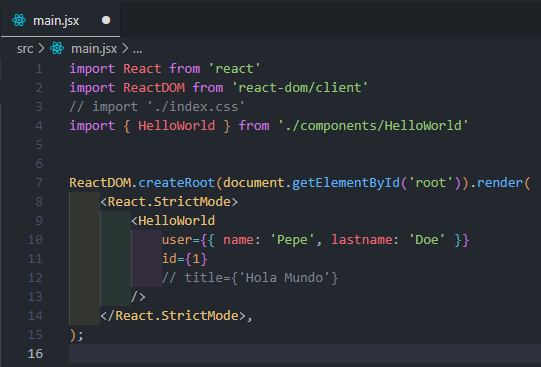


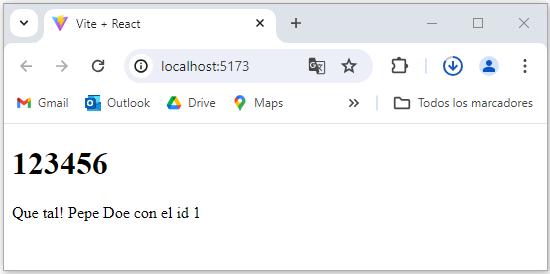


Si ejecutamos la aplicación, veremos que la prop **title** utiliza el valor por defecto establecido en el componente padre **main.jsx** y no el valor por defecto establecido en **HelloWorld.jsx**.

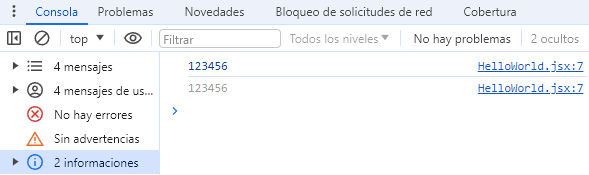


Para mostrar el valor numérico en el título, comentamos o eliminamos el valor por defecto para la prop **title** definido en el componente padre **main.jsx**





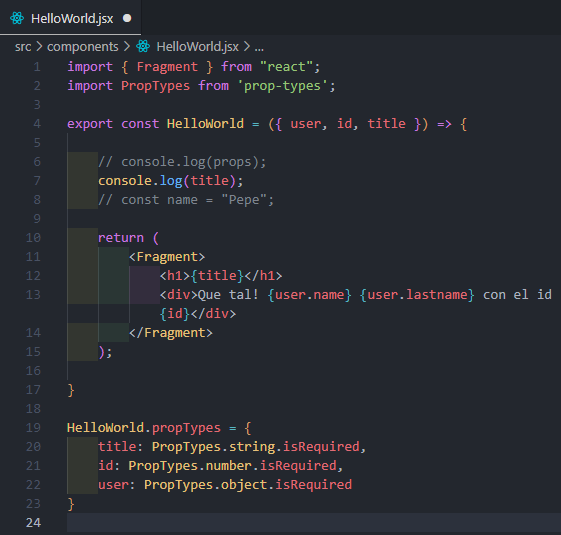
En la consola, no se mostrarán errores ya que hemos definido el valor por defecto directamente en el parámetro de la función (atributo desestructurado) y no hemos marcado la prop **title** como obligatoria en **propTypes**.

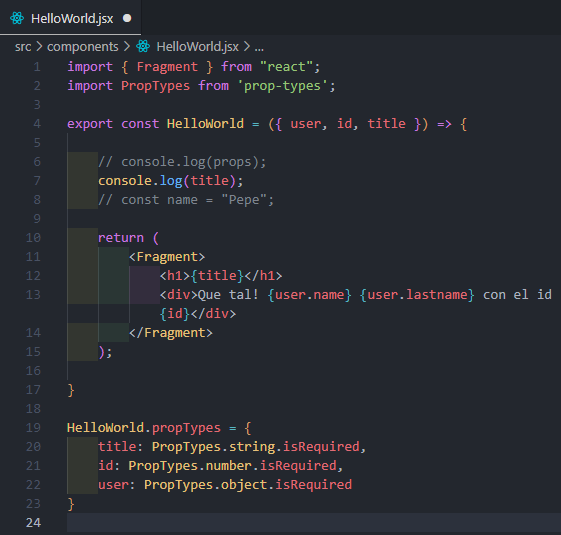


Uso de defaultProps para definir el valor por defecto

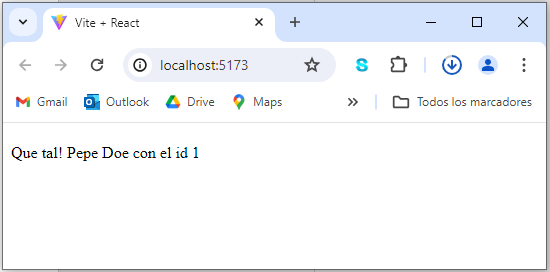
Una forma más desacoplada y mejorada de manejar valores por defecto en las propiedades es mediante el uso de **defaultProps**. Esto nos permite establecer valores predeterminados para las props en caso de que no se pasen desde el componente padre.

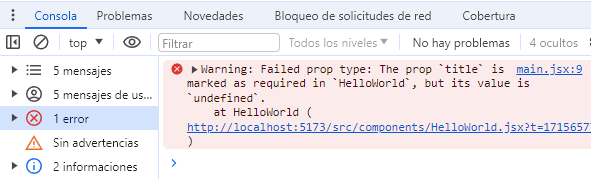
Primero, restauramos el componente **HelloWorld** eliminando el valor por defecto para **title** y marcando la prop cómo requerida en **propTypes**.



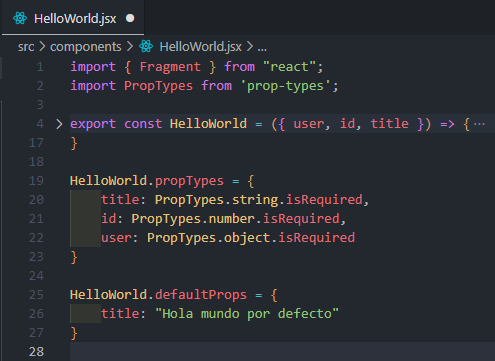


Si ejecutamos la aplicación sin pasar la prop **title** desde el componente padre, veremos que no se muestra el título en la página web, además se produce un error en la consola indicando que la prop es requerida y que su valor es **undefined**.

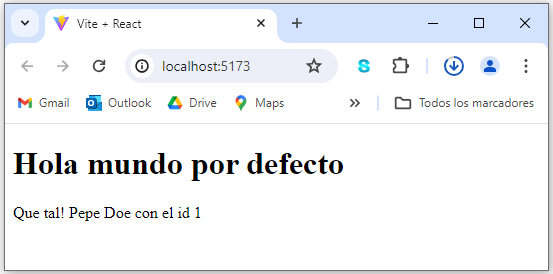




Para resolver esto, podemos definir un valor por defecto para **title** utilizando **defaultProps**. Colocamos la definición de **defaultProps** debajo del bloque de propTypes (sigue la siguiente lógica: primero se define el tipo de dato con **propTypes** y luego se establece su valor con **defaultProps**).



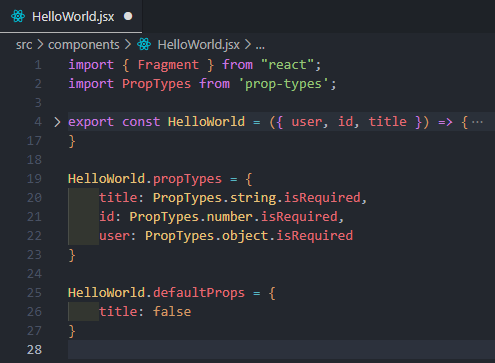
Ahora, si ejecutamos la aplicación sin pasar la prop **title** desde el componente padre, el componente **HelloWorld** utilizará el valor por defecto definido en **defaultProps** en lugar de lanzar un error.

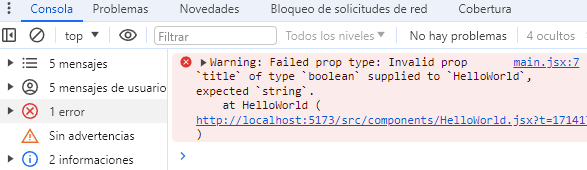


Esta forma de asignar valores por defecto es una práctica recomendada y desacoplada, mejorando la legibilidad y la robustez del código en nuestras aplicaciones React.

El problema con el tipo de dato no válido

Si definimos un tipo de dato no válido para la prop **title**, como un booleano, se producirá un error en la consola. Esto ocurre porque estamos violando la definición de **propTypes**, que espera que **title** sea de tipo string. Incluso si establecemos un valor por defecto de tipo booleano utilizando **defaultProps**, el error persistirá porque PropTypes aún espera un tipo de dato específico.

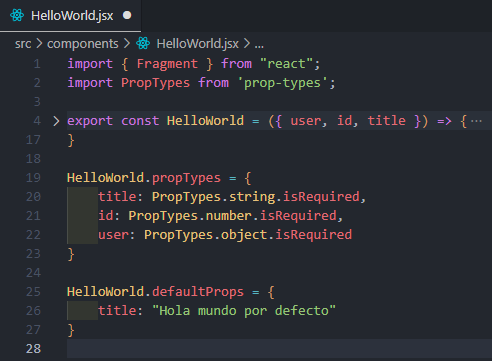




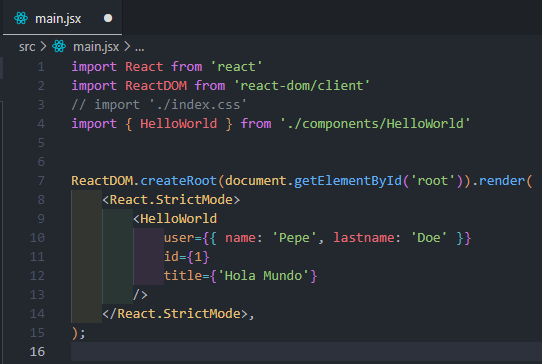
Para corregir este problema, debemos asegurarnos de que el valor por defecto establecido en **defaultProps** coincida con el tipo de dato esperado definido en **propTypes**. En este caso, debemos corregir el valor por defecto de **title** para que sea una cadena de texto en lugar de un booleano.

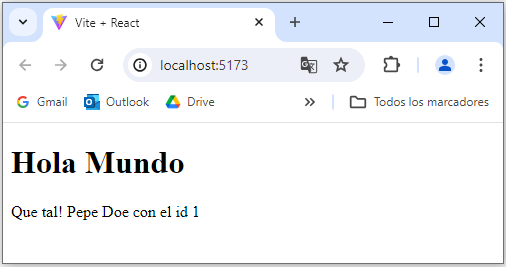
El problema de definir 2 valores por defecto para cada componente

Si definimos un valor por defecto para la prop **title** tanto en el componente **HelloWorld** mediante **defaultProps** como en el componente padre, **main.jsx**, mediante un valor como un argumento, el valor definido en el componente padre prevalecerá. Esto se debe a que los valores por defecto definidos en **defaultProps** se aplican sólo si la prop no se proporciona desde el componente padre.



Por lo tanto, en el ejemplo dado, el valor **"Hola Mundo"** que se pasa desde **main.jsx** se utilizará como el valor de **title**, y el valor definido en **defaultProps** en **HelloWorld** será ignorado.

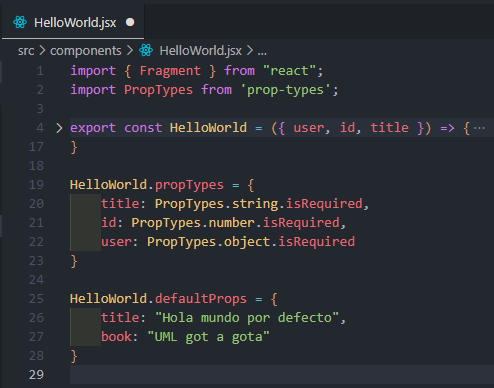




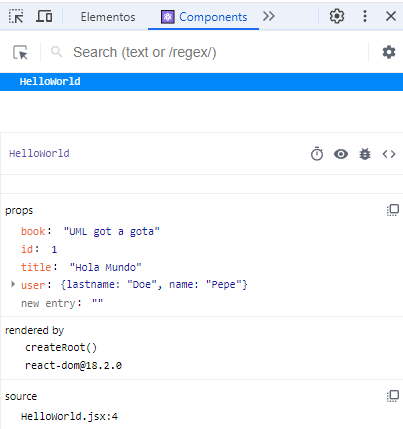
Uso de la consola de Chrome para visualizar las propiedades no definidas

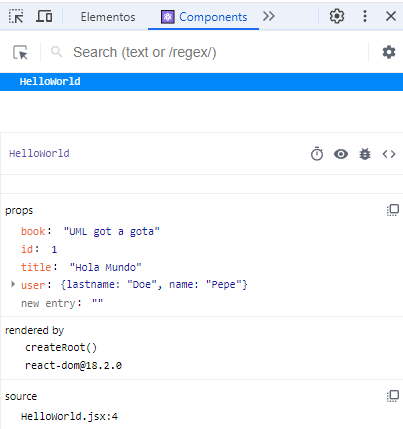
Una práctica útil al desarrollar aplicaciones React es utilizar la herramienta de desarrollo de Chrome para inspeccionar y depurar nuestros componentes. Esto nos permite visualizar props no definidas y comprender mejor cómo se están pasando los datos entre los componentes.

Se procede a añadir una nueva prop llamada **book** en los **defaultProps** del componente **HelloWorld**. Aunque esta prop no se pasa desde el componente padre, se genera automáticamente como una nueva prop en el componente **HelloWorld**.



Al abrir la consola de Chrome y navegar a la sección **Components**, encontraremos nuestro componente **HelloWorld** listado allí. Al hacer clic en él, podemos ver todas las props que se pasan al componente, incluida la prop **book** con su valor por defecto.





Para visualizar la prop **book** en el navegador, primero debemos asegurarnos de definir su párametro (en este caso como una propiedad desestructurada) en la función principal del componente. Luego, podemos renderizar su valor dentro de las etiquetas **<div>** del componente.





