Configurando PropTypes para validaciones en las propiedades

En esta sección, se realizara una configuración en la validación de los valores definidos en las propiedades de un componente utilizando la biblioteca PropTypes.

La biblioteca PropTypes

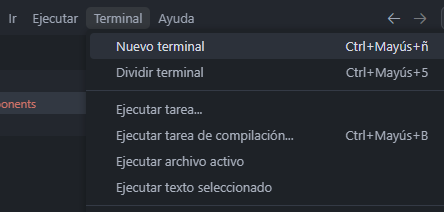
PropTypes es una herramienta invaluable para administrar propiedades en React. A diferencia de simplemente establecer valores por defecto, PropTypes proporciona una solución más sólida y desacoplada para validar las props recibidas y definir valores predeterminados cuando sea necesario.

Esta biblioteca permite especificar los tipos de datos esperados para las propiedades definidas, lo que ayuda a garantizar que las aplicaciones funcionen de manera más predecible y resistente a errores. Además, se puede aplicar que ciertas propiedades sean obligatorias, lo que mejora la robustez y claridad del código.

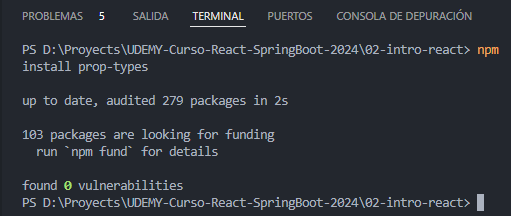
Instalar la dependencia prop-types

Para empezar a usar PropTypes, se procede a instalar la dependencia **prop-types** en el proyecto. Es importante tener en cuenta que Vite no incluye PropTypes por defecto, razón por la cual se añade manualmente al archivo **package.json**.

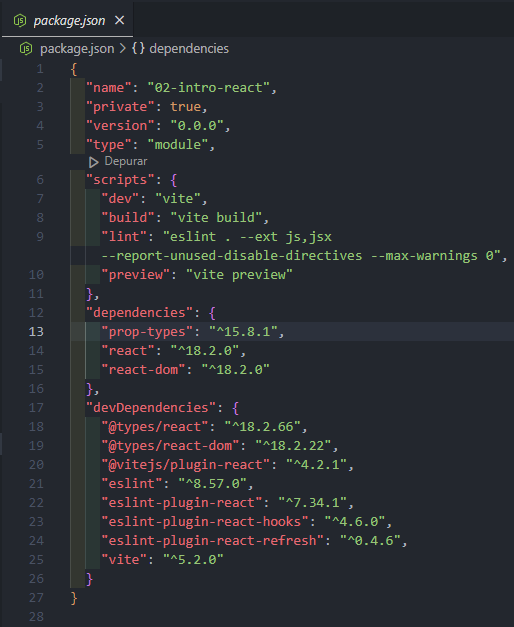
Para hacerlo, abre una nueva terminal en Visual Studio Code o en cualquier otra herramienta como Powershell, pero asegúrate de estar en la raíz del proyecto (para aquello se ejecuta el comando **cd** y la ruta hacia el proyecto; además se puede listar los archivos y carpetas mediante el comando **ls**).



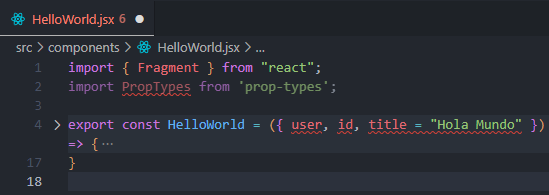
Luego, ejecuta el comando **npm install prop-types**.



Una vez instalado, cierra la consola y verifica en el archivo **package.json** que **prop-types** esté incluido como una dependencia.



Luego, ve al componente **HelloWorld** e importa **PropTypes** manualmente mediante un **import** de la siguiente manera como se muestra en la imagen:

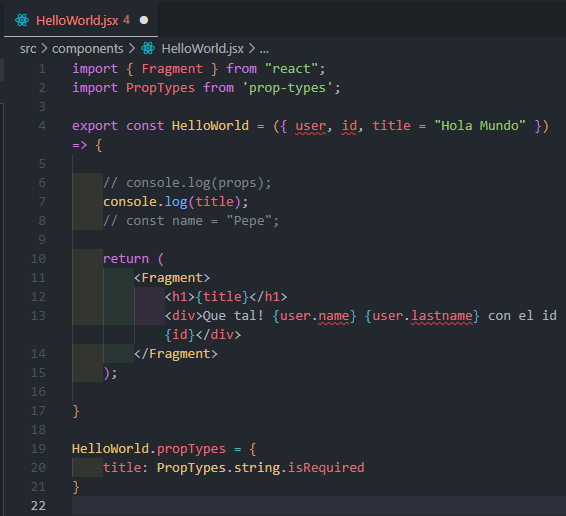


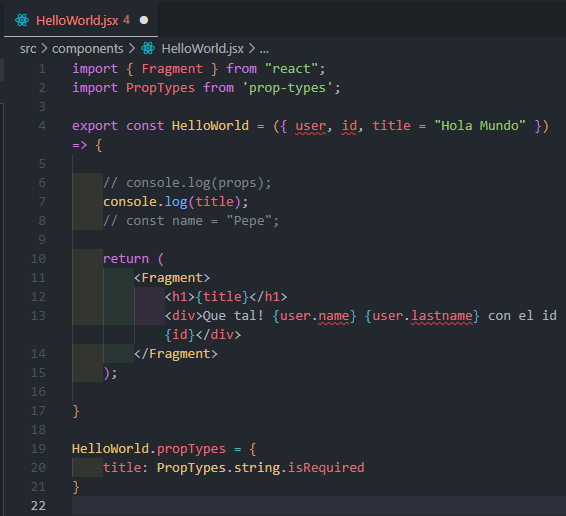
Ten en cuenta que con PropTypes, se puede establecer un contrato claro entre los componentes, mejorando la calidad y la mantenibilidad del código en las aplicaciones React.

Validar el tipo de dato y hacerlo obligatorio

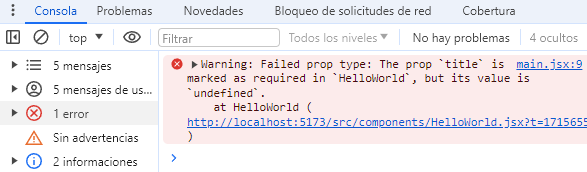
Ahora que se ha importado **PropTypes** en el componente **HelloWorld**, se puede definir claramente el tipo de datos esperado para cada propiedad utilizando **PropTypes**. Además, se puede hacer que ciertas propiedades sean obligatorias utilizando la función **isRequired**.

Por ejemplo, en el componente **HelloWorld**, se define que la prop **title** sea de tipo string y que sea requerida utilizando **PropTypes.string.isRequired**:



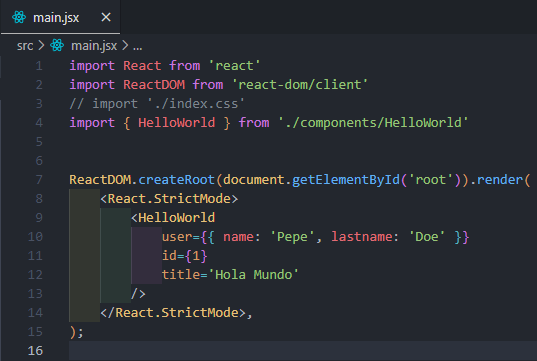


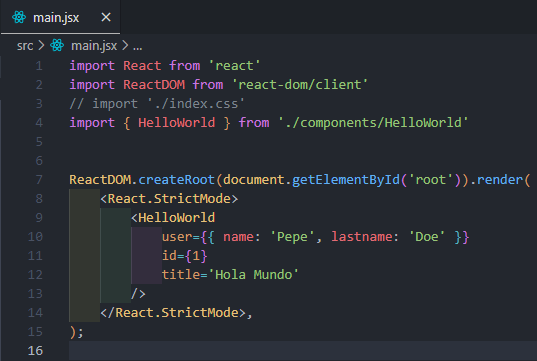
Al guardar estos cambios y ejecutar la aplicación, a plena vista se ve que funciona bien la página web. Sin embargo, es posible que haya un error en la consola.



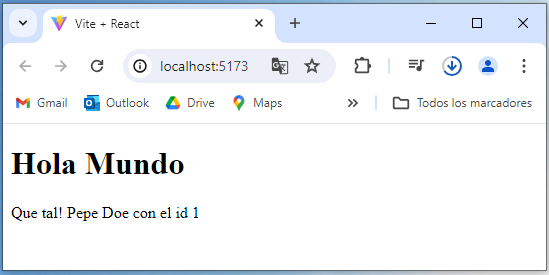
Este error indicará que la prop **title** ha fallado la validación, ya que está marcada como requerida pero se pasa como **undefined**. Esto puede ocurrir si no se proporciona la propiedad **title** en el componente principal (**main.jsx**), a pesar de que se ha establecido su valor por defecto en **HelloWorld.jsx**.

Para resolver este problema, asegurate de proporcionar un valor para la prop **title** en el componente principal. Por ejemplo:



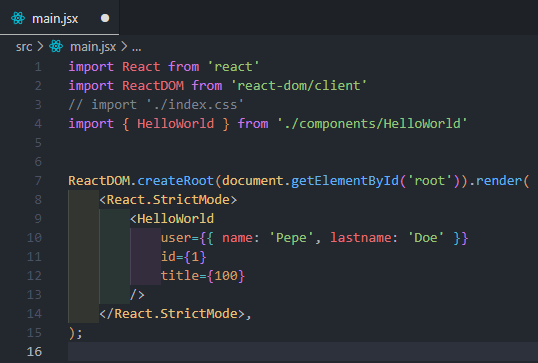


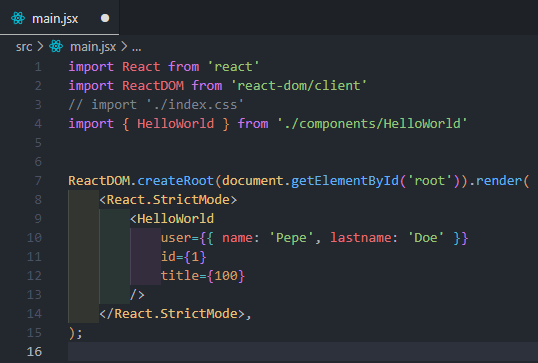
Guarda los cambios y verifica que el error en la consola haya desaparecido. Ahora la validación de **PropTypes** se realiza correctamente y la aplicación web funciona sin problemas.

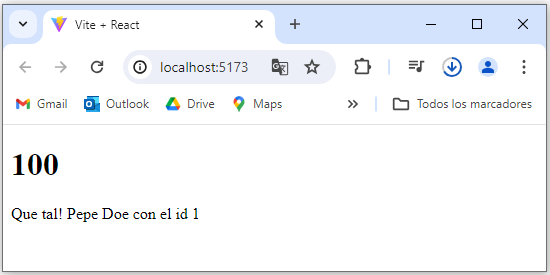


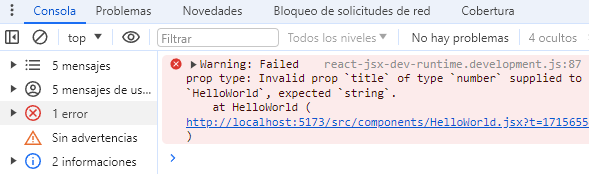
Problemas con los tipos de datos

Al intentar pasar un título como un número en lugar de una cadena de texto, se obtiene un error en la consola a pesar de que la aplicación se visualiza correctamente en el navegador (recuerda que el valor se coloca entre llaves para validar el tipo de dato).



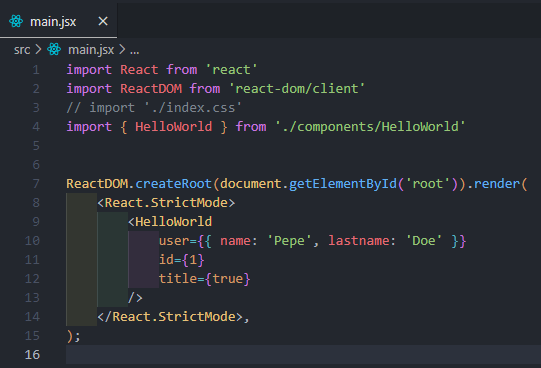


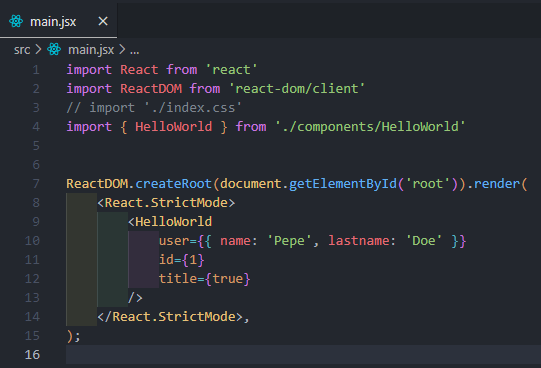


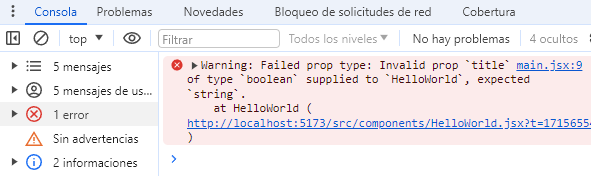


El error indicará que el tipo de dato es inválido, ya que el título es de tipo number pero se esperaba que fuera de tipo String. Esto demuestra cómo se puede ser estricto en los tipos de datos que se aceptan y que ciertas propiedades sean obligatorias.

Otro escenario común es pasar un valor booleano como título, por ejemplo, asignando el valor **true**:

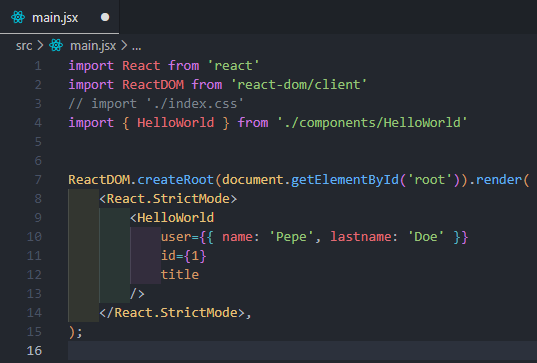


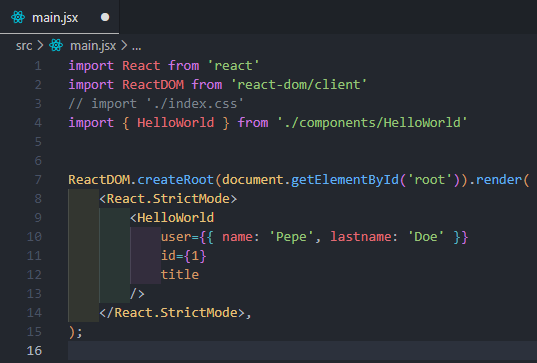




En este caso, la consola mostrará un error indicando que el tipo de dato es inválido, ya que el **prop.title** fue manejado en el componente **HelloWorld** y se esperaba una cadena de texto en lugar de un booleano.

Incluso al utilizar un atajo como definir solamente **title** sin un valor asignado, que es equivalente a **title={true}**, se obtiene el mismo error.

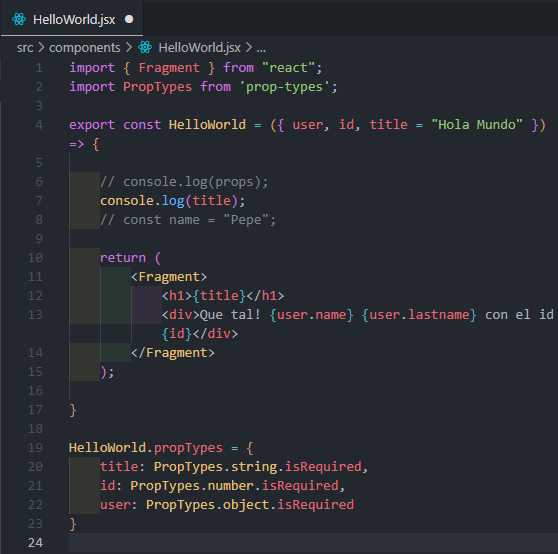




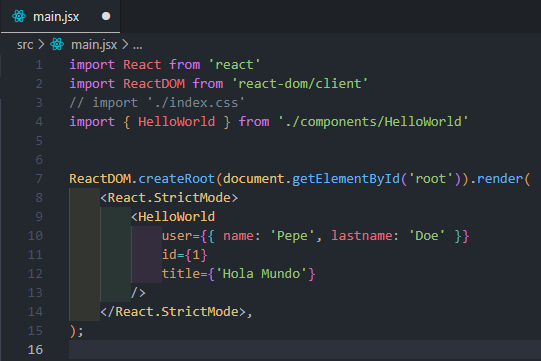
Estos errores muestran como PropTypes sirve para ayudar a garantizar la integridad de los datos y mejorar la robustez de las aplicaciones React.

Manejo de varios tipos de datos con PropTypes

Es esencial definir los PropTypes para cada propiedad en los componentes React. Por ejemplo, para el **id**, se puede asegurar de que sea un número y, opcionalmente, que sea requerido. Por otro lado, para el **user**, debe ser un objeto.

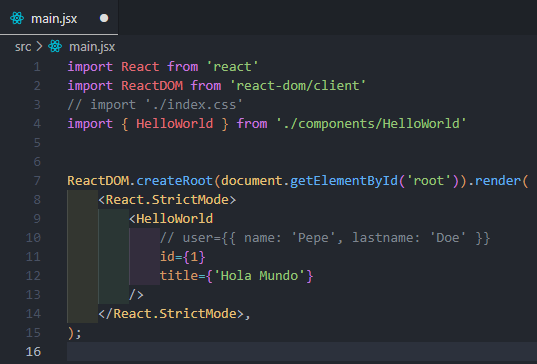


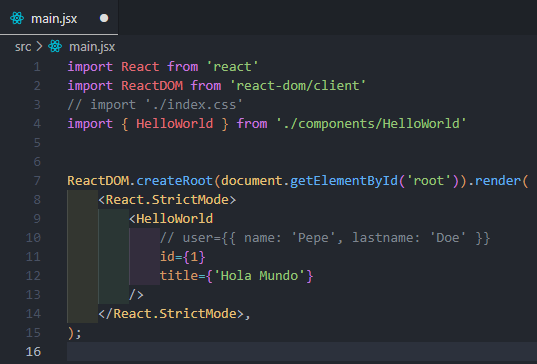
En el componente **main.jsx**, se puede ver cómo se pasan estas props (se restaura el valor de **title**):

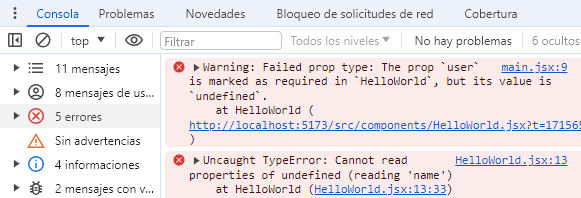


Es importante recordar que si se omite una propiedad requerida entonces se recibe un error en la consola, el cual indica que propiedad falta.

Por ejemplo, al eliminar (o comentar) la prop **user** en **main.jsx**, se recibe un error indicando que la prop **user** es requerida en el componente **HelloWorld**, pero su valor no está definido.







Procede a restaurar el último cambio del archivo **main.jsx** para la siguiente sección.

