# JSX

JSX stands for JavaScript XML and it allows us to write HTML in React.

All the components of react app must have extension jsx.

# VITE

Vite optimizes code compilation and execution, which can result in a better end-user experience due to faster loading times and lower resource usage. It is super fast.

Step 1: npm create vite@5.4.0

Project name:

Select a framework: react

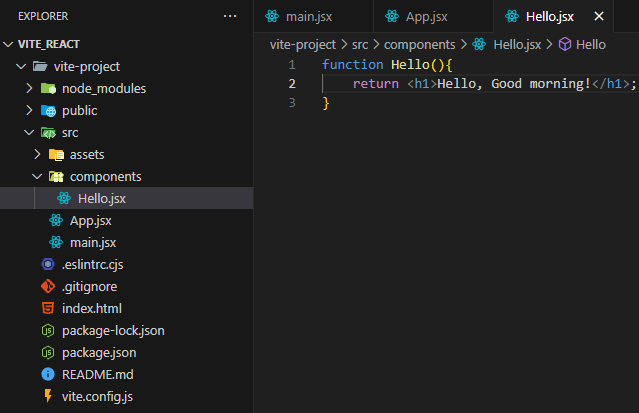
Select a variant: javascript

Step 2: cd directory

Step 3: npm install

Step 4: npm run dev

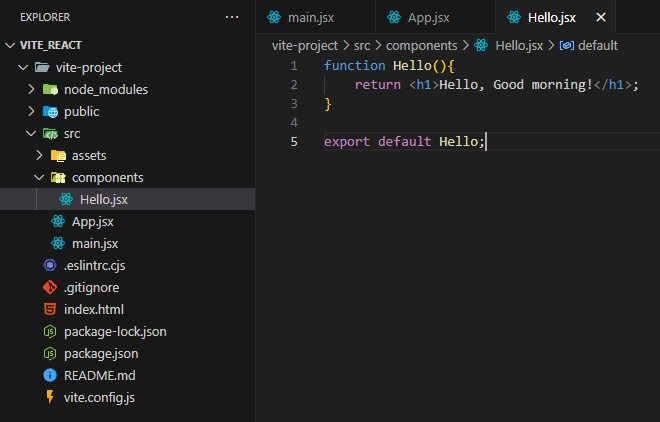
Step 5: ctrl + C to stop the server.

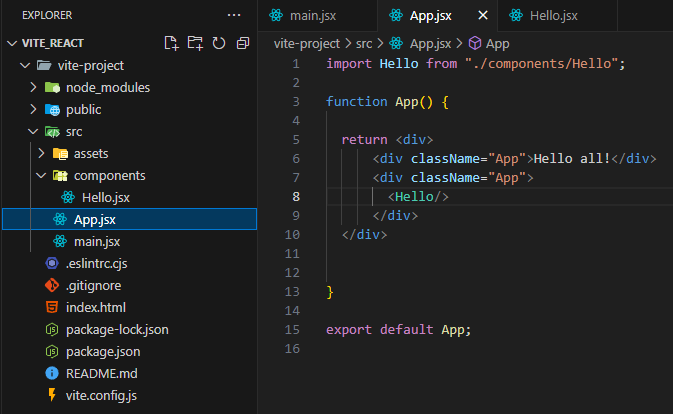


Create a component directory and manage all the components in the same directory.

The component name must be started with a capital letter and also the function inside the file must be of the same name as of the file.

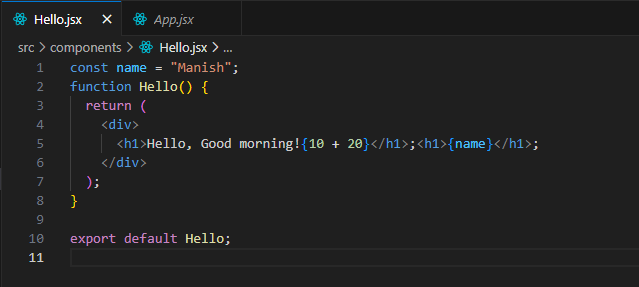
We have to export the function before we import in some file.





# Use of jsx.

JSX allows us to write javascript code inside the html. And using jsx, we can only return 1 element. ie. We use div containers.



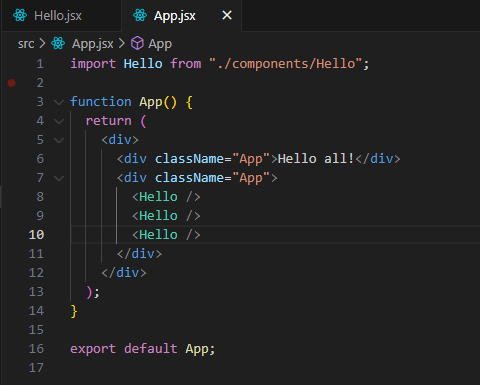
# Reusability of component

It helps in making the application modular.

We can manage each component individually and use any where needed for multiple times.

Abstraction: Hiding away the complexity of codes.

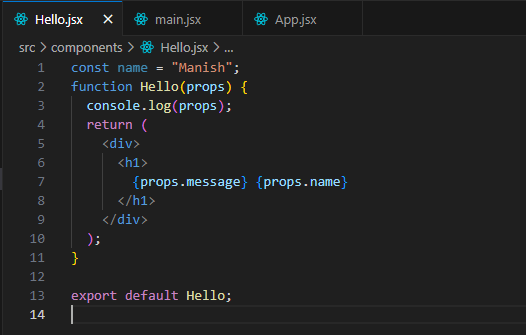
We can use the components as shown below in our application.

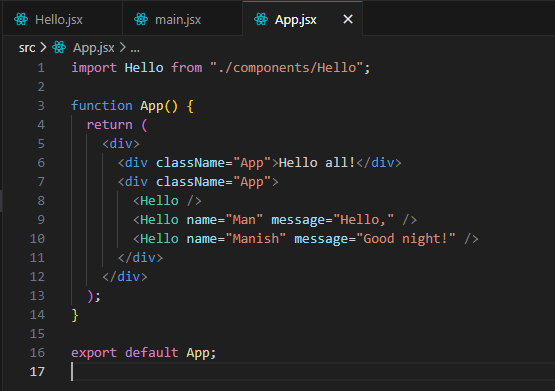


# What are props in React?

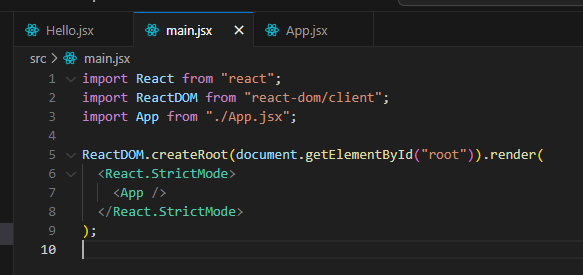
Props are arguments passed into React components. Props are passed to components via HTML attributes from one component to another. Props stands for properties. It helps use of dynamic components to get different results.

We can pass any data types using props.





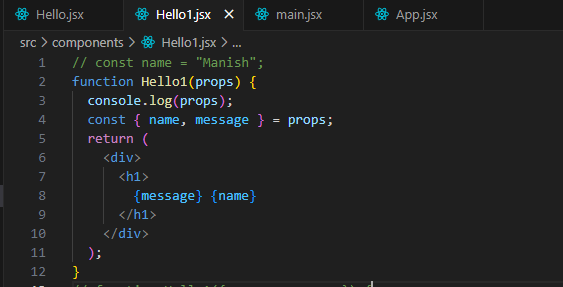
# Strict mode in react



Strict mode is used in dev mode which runs the application 2 times for safety purpose.

It is removed while application goes to production.

# Destructuring props



We can directly store the arguments to the variables and use it in the component by destructuring the props.

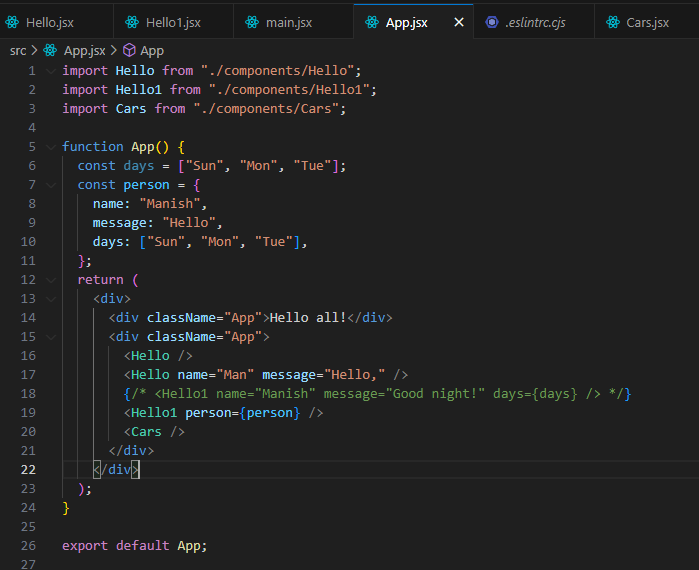


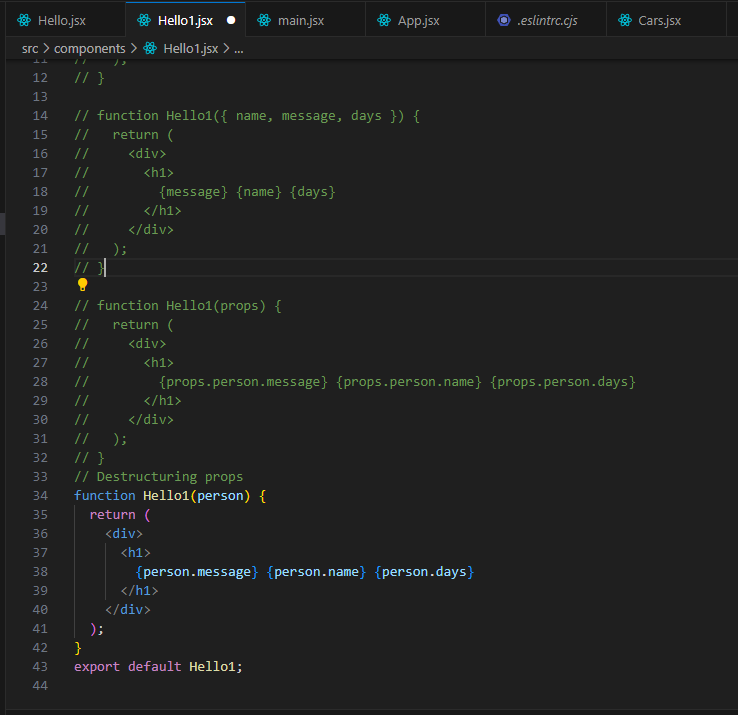
We can also directly destructure the props up in the function argument. All the methods give the same result.

# Immutability of props

We cannot change the value or reassign value of the variables used to store argument in the components.

# Passing Arrays and Object to Components using Props





# Using of map function

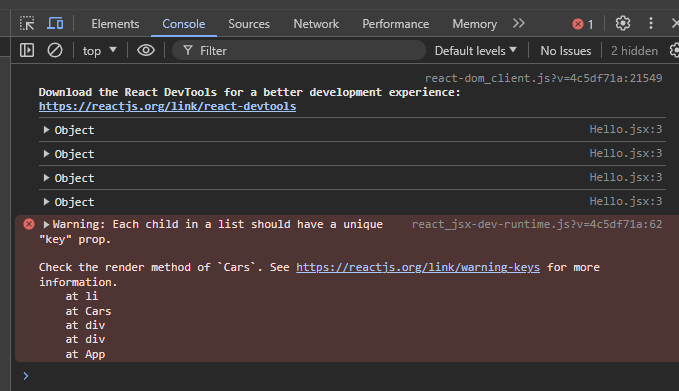
const numbers = [4, 9, 16, 25];  
const newArr = numbers.map(Math.sqrt)

const numbers = [65, 44, 12, 4];  
const newArr = numbers.map(myFunction)  
  
function myFunction(num) {  
  return num \* 10;  
}

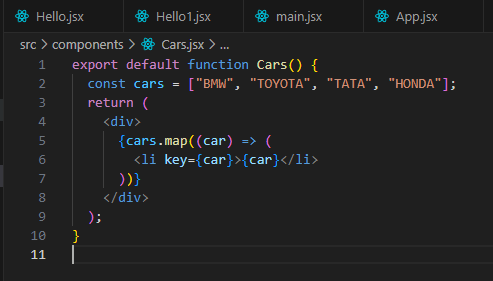
map() creates a new array from calling a function for every array element.

map() does not execute the function for empty elements.

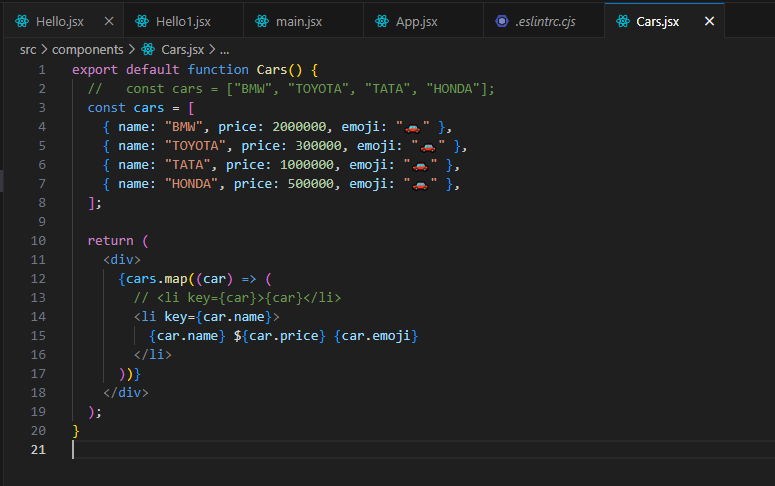
map() does not change the original array.



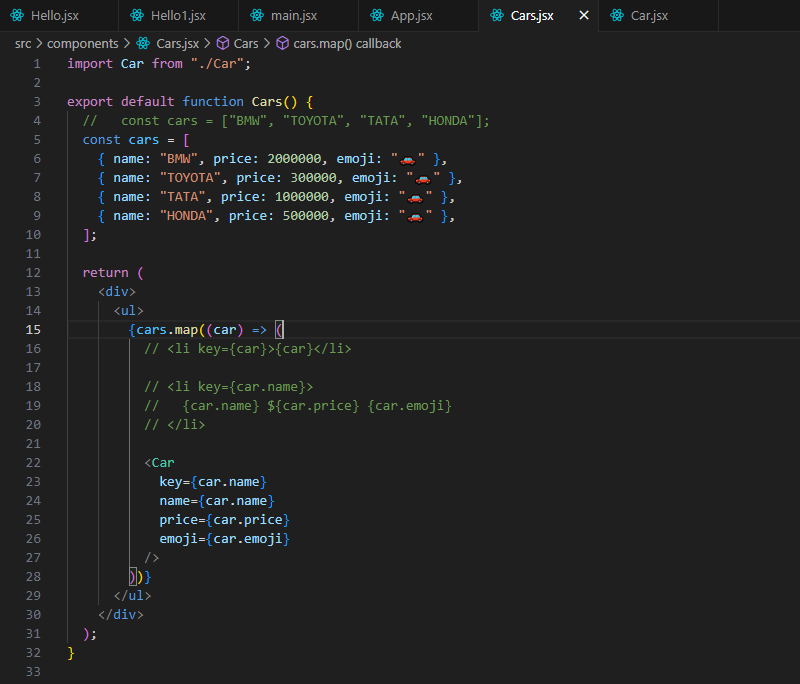
We can just make ach array item unique by using key as follows:

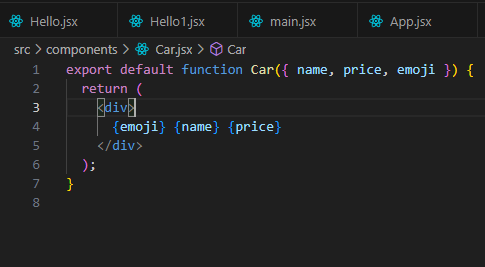


# Rendering array of Objects

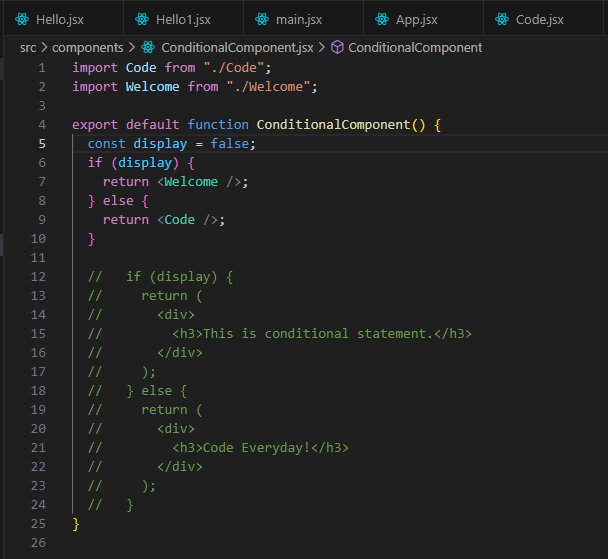


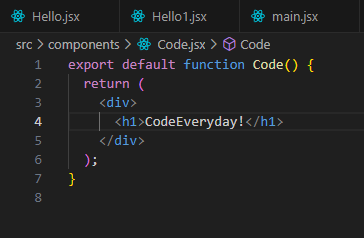
# Rendering components inside a loop





# Conditionally Rendering JSX & Components

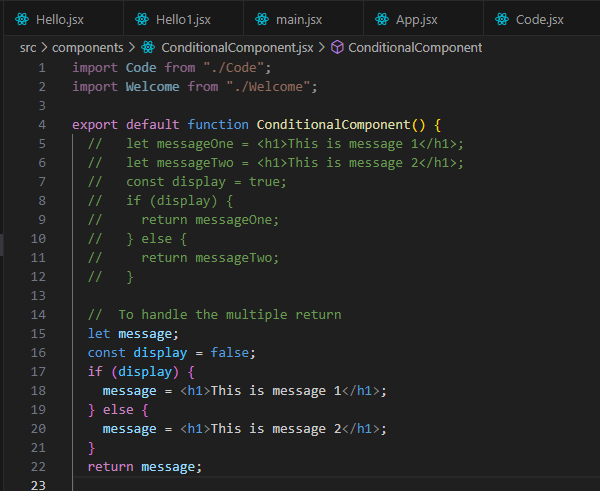






It’s not good to have wo return statements in a single component.

# Conditional Rendering using Element Variables



Always use element variables to conditionally render JS elements.

# Ternary Operators

// Using ternary operator

  const display = false;

  return display ? <h1>Message 1</h1> : <h1>Message 2</h1>;

Or we can also use components.

  // Using ternary operator

  const display = true;

  return display ? <Welcome /> : <Code />;