**Why typescript+react:-**

With static type checking you get to learn about potential bugs as you are typing code, then heading to the browser and figuring out at runtime

npx create-react-app my-app --template typescript

npm install --save typescript @types/node @types/react @types/react-dom @types/jest

You might need to configure TypeScript according to your project's needs. You can do this by creating a **tsconfig.json** file in the root of your project and adding necessary configurations.

**{**

**"compilerOptions": {**

**"target": "es5",**

**"lib": ["dom", "dom.iterable", "esnext"],**

**"allowJs": true,**

**"skipLibCheck": true,**

**"esModuleInterop": true,**

**"allowSyntheticDefaultImports": true,**

**"strict": true,**

**"forceConsistentCasingInFileNames": true,**

**"module": "esnext",**

**"moduleResolution": "node",**

**"resolveJsonModule": true,**

**"isolatedModules": true,**

**"noEmit": true,**

**"jsx": "react"**

**},**

**"include": ["src"]**

**}**

components are defined in .tsx file extension

**basic prop types:-**

type GreetProps ={

name:’sting’,

messageCount:number,

isLoggedIn:boolean

}

Export const Greet=(props:GreetProps)=>[

}

Use types when building applications and interfaces when building libraries

Const personname={

First:’bruce’,

Last:’wayne’

}

type PersonProps={

Name:{

First:string,

Last:string,

}

Const namelist=[

{first:’bruce,

Last:’wayne’}

]

Type personListsProps={

Names:{

First:string,last:string}

}[]

Advanced types:-

Type statusProps={

status:”loading’|”success”|”error”

}

Type oscarprops={

Children:React.ReactNode

}

Optional props

Type greetprops={

Name:string,

Messagecount?:number,

Isloggeinin?:Boolean

}

Const {messagecount=0}=props

Event props:-

Type buttonpros={

Handleclick : ()=>void

}

Type buttonpros={

Handleclick : (event:React.MouseEvent<HTMLButtonElement>,id:number)=>void

}

type inputprops={

Value:string,

handleChange : event:React.ChangeEvent<HTMLElement>)=>void

}

Const handleInputChange=(event:React.ChangeEvent<HTMLInputElement>)=>{

Console.log(event)

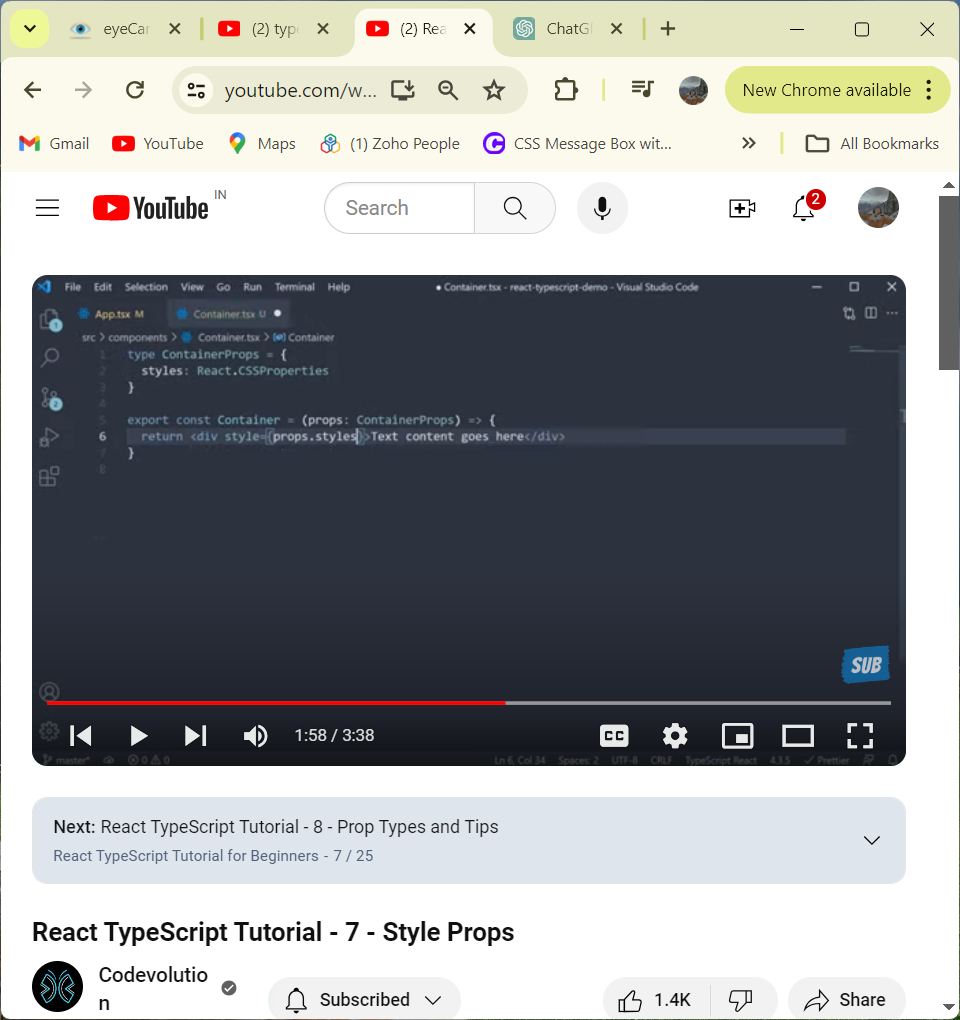
}

**Style Props:-**

Type containerprops={

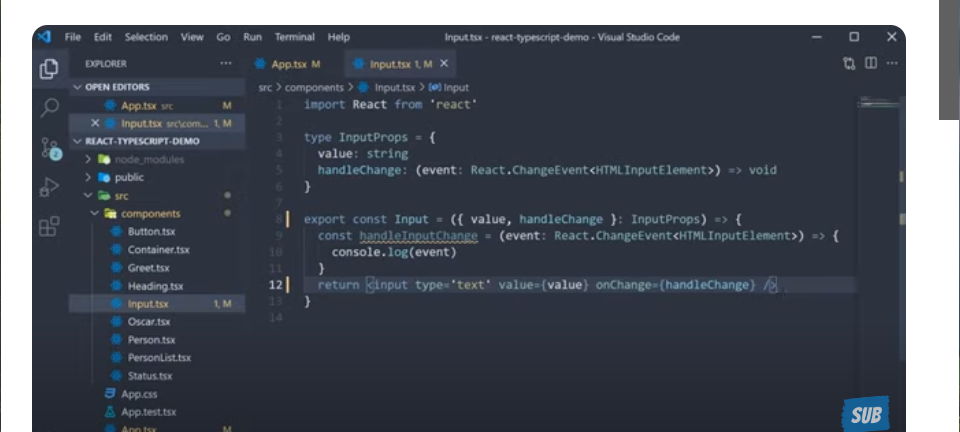
Styles:React.CSSProperties

}



**Prop Types and Tips:-**

1)destructuring props:-



2)can create types to another file

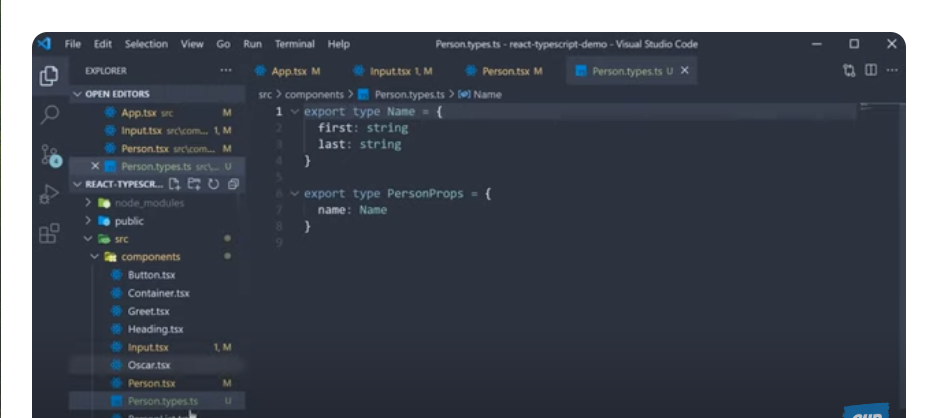
A screenshot of a computer

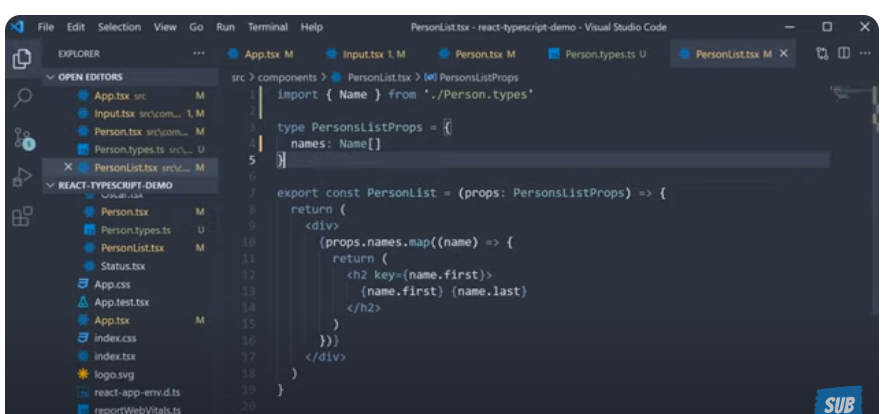
Description automatically generated

A screenshot of a computer program

Description automatically generated

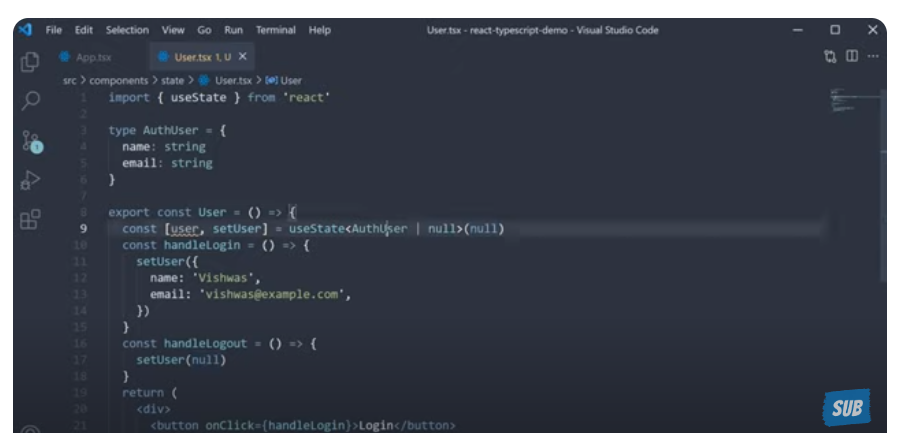
3)it is possible to use a type in multiple places

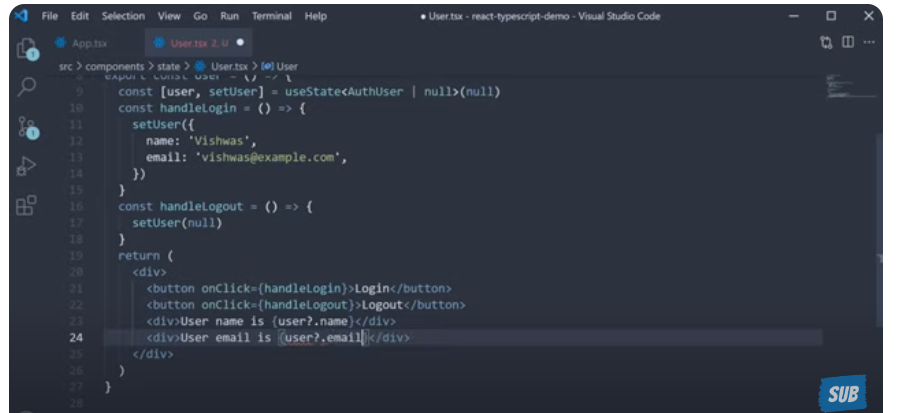




**useState Hook:-**

**useState Future Value:-**



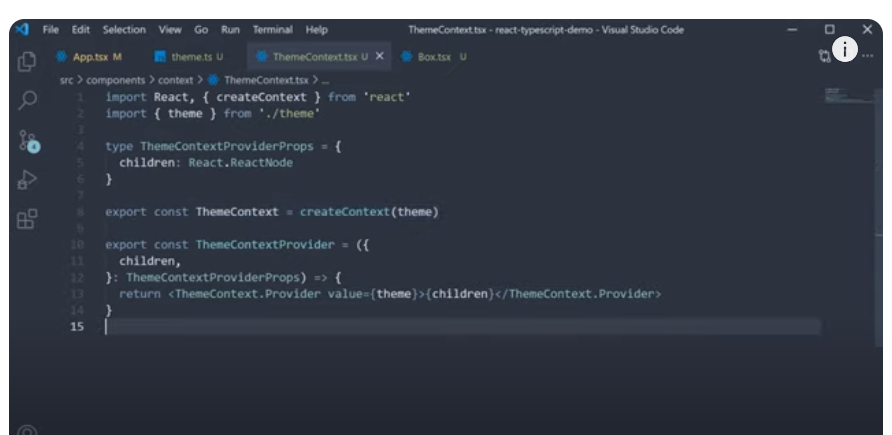


**useState Type Assertion:-**

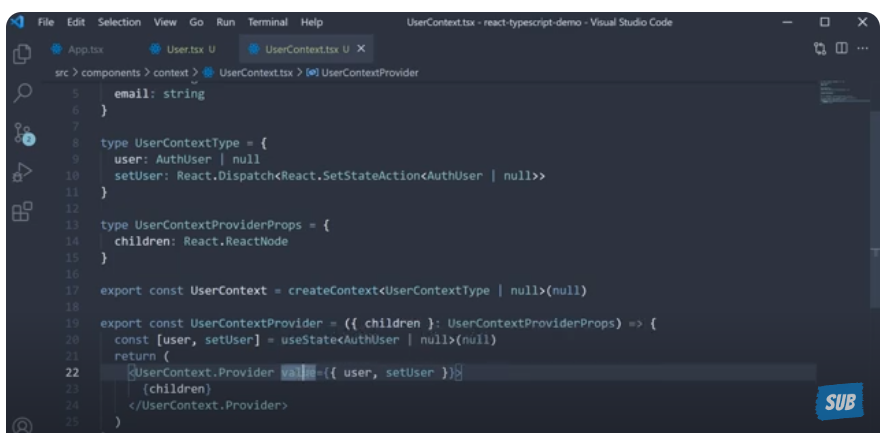


**Use Reducer Hook:-**

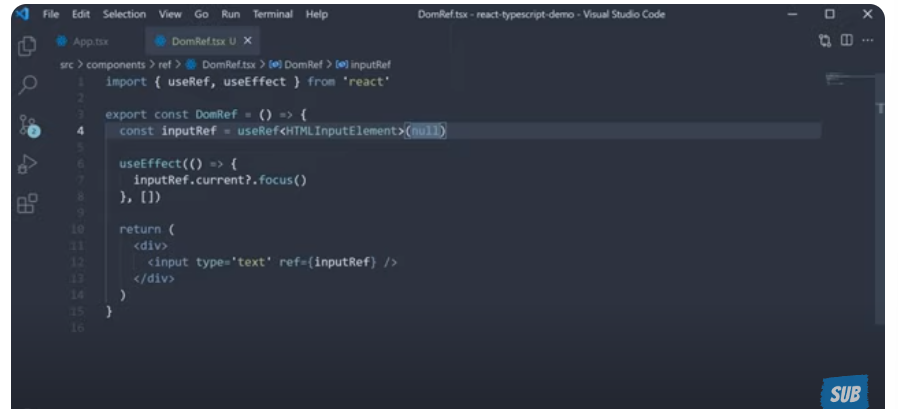
**useContext Hook:-**

****

**useContext Future Value:-**



**useRef Hook:-**



**Component Prop:-**

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

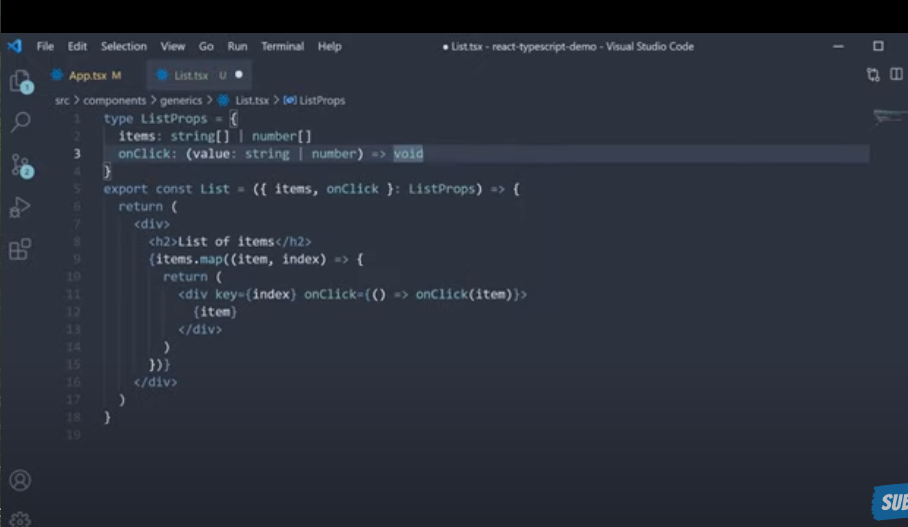
A computer screen shot of a program

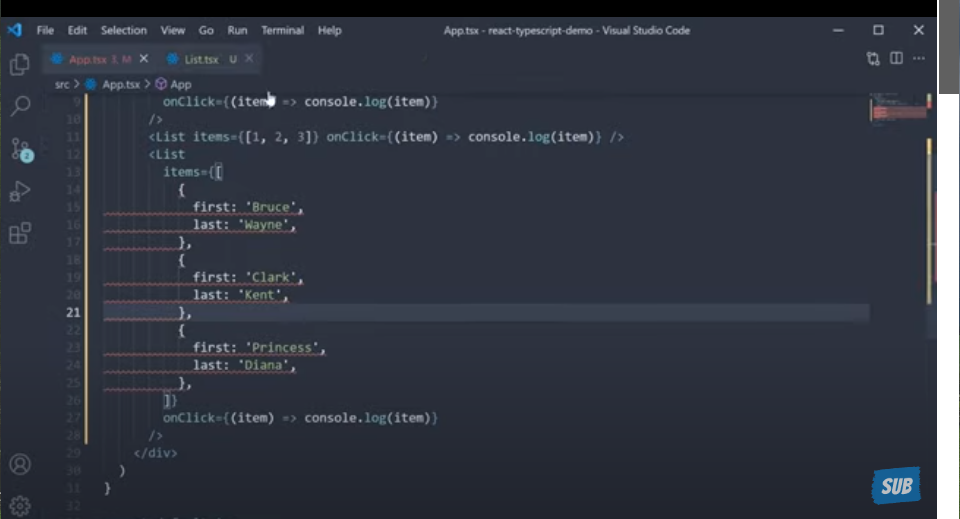
Description automatically generated

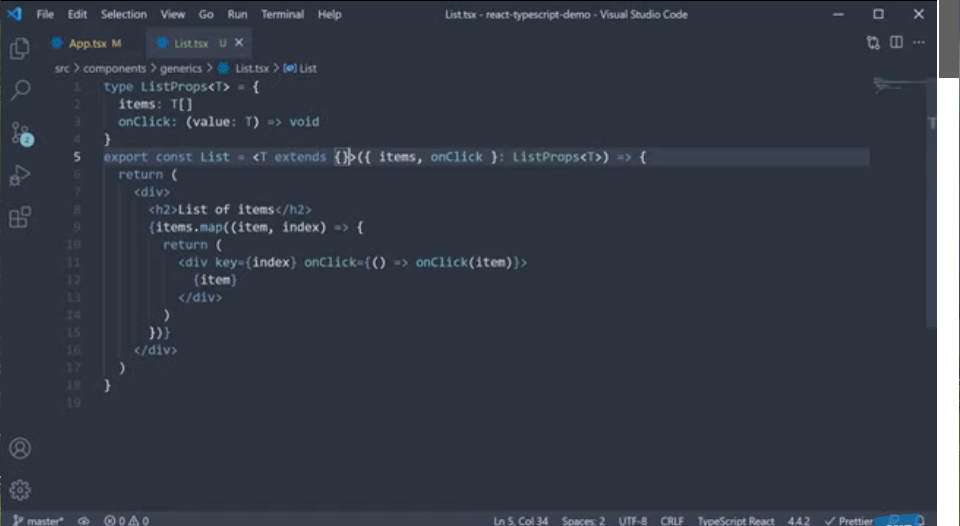
**Generic Props:-**

**A screenshot of a computer

Description automatically generated**

****

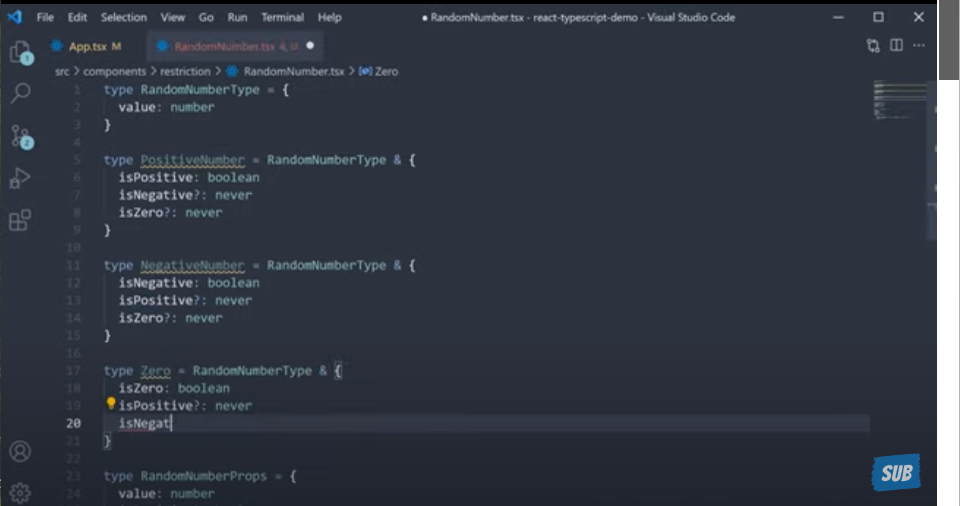
****

****

**A screenshot of a computer

Description automatically generated**

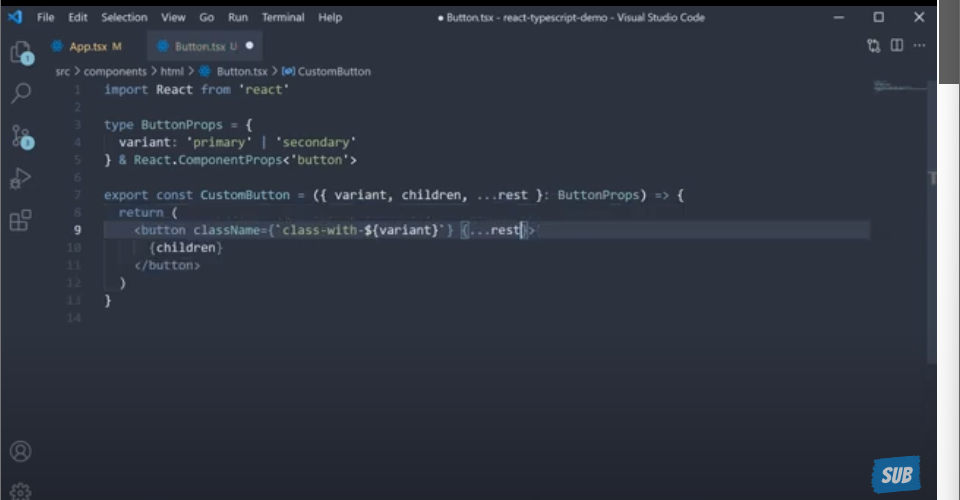
**Restricting Props:-**

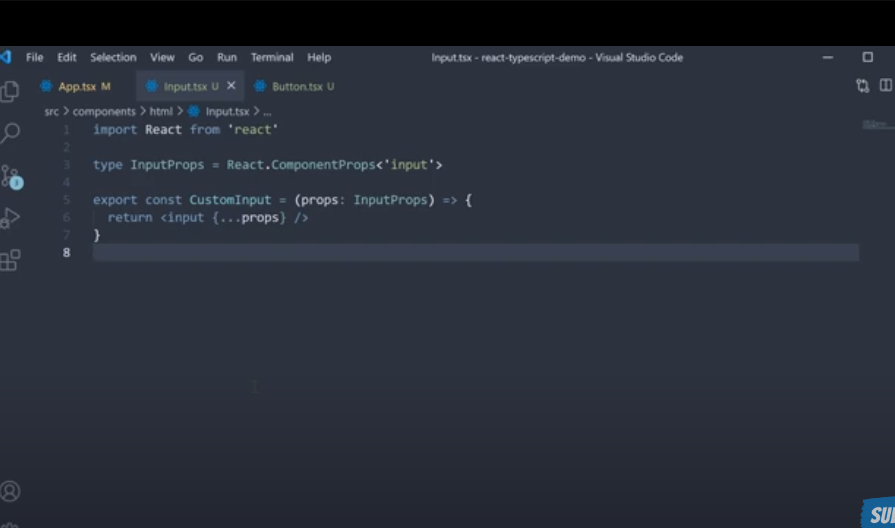
****

**A screenshot of a computer

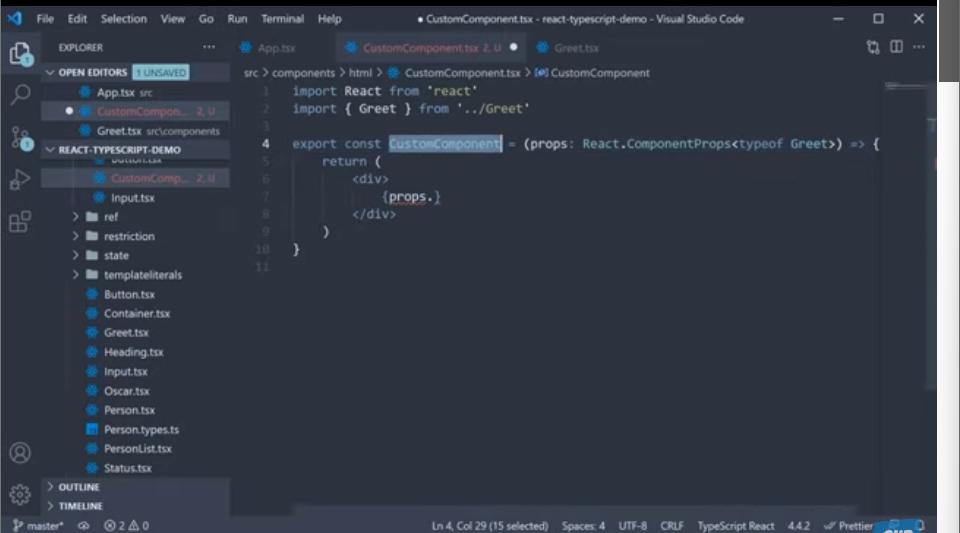
Description automatically generated**

**Wrapping HTML Elements:-**

****

****

**Extracting a Components Prop Types:-**

****