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:’string’,
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:-

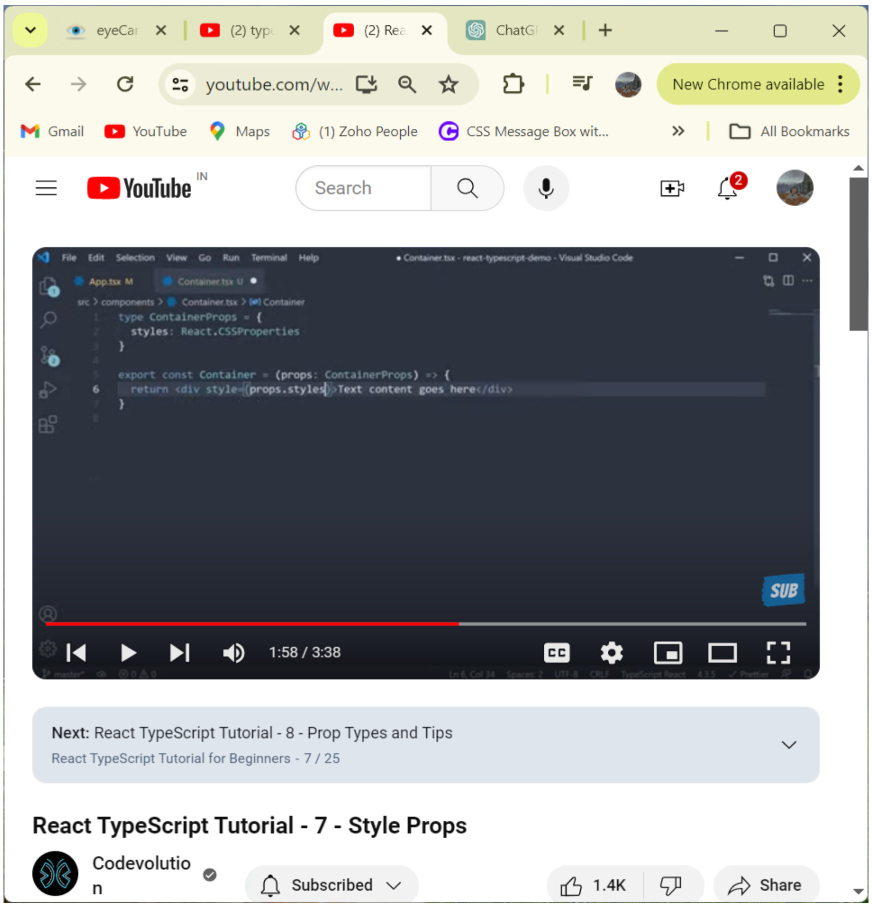
```
Type buttonpros={

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
}
```



```tsx
type ContainerProps = {
  styles: React.CSSProperties
}

export const Container = (props: ContainerProps) => {
  return <div style={props.styles}>Text content goes here</div>
}
```

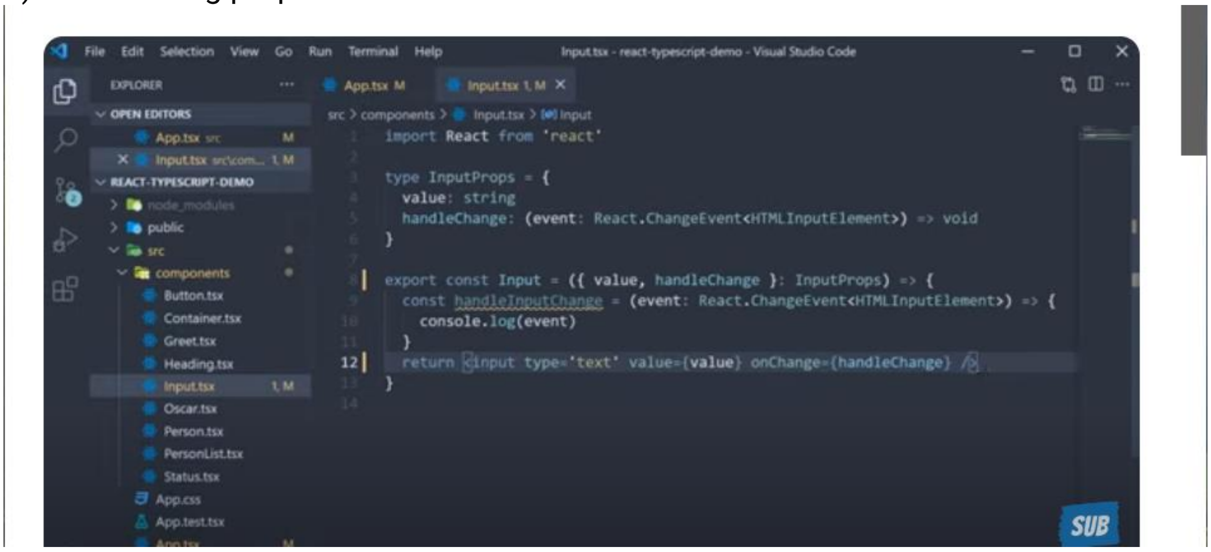
Next: React TypeScript Tutorial - 8 - Prop Types and Tips
React TypeScript Tutorial for Beginners - 7 / 25

React TypeScript Tutorial - 7 - Style Props

Codevolution

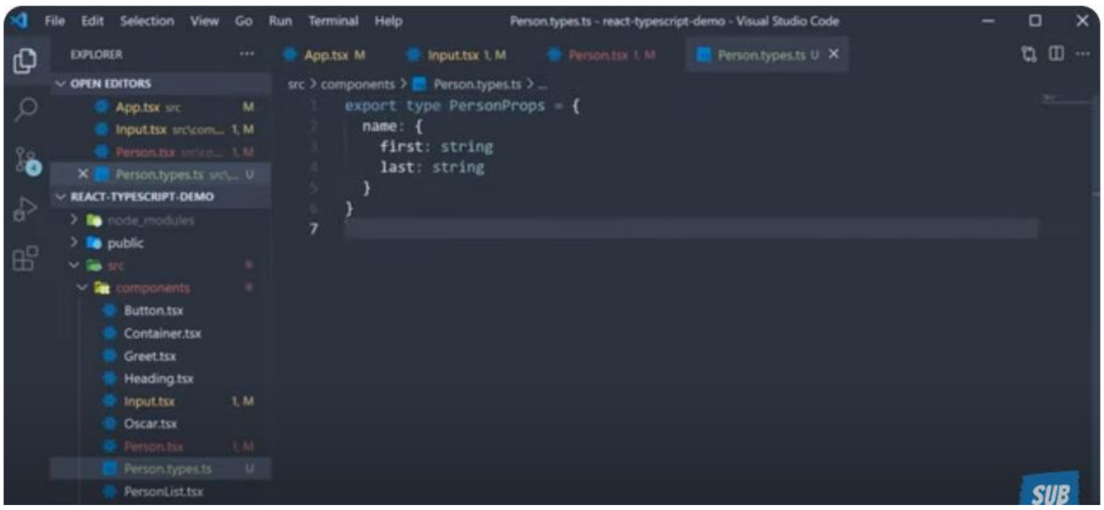
Prop Types and Tips:-

1)destructuring props:-

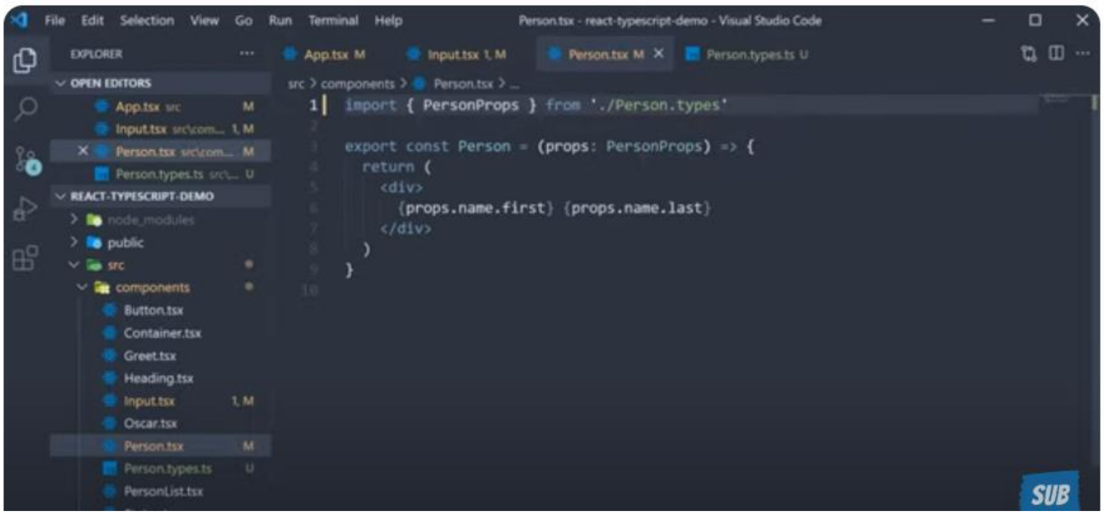


```tsx
import React from 'react'

type InputProps = {
  value: string
  handleChange: (event: React.ChangeEvent<HTMLInputElement>) => void
}

export const Input = ({ value, handleChange }: InputProps) => {
  const handleInputChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    console.log(event)
  }
  return <input type='text' value={value} onChange={handleChange} />
}
```

2)can create types to another file

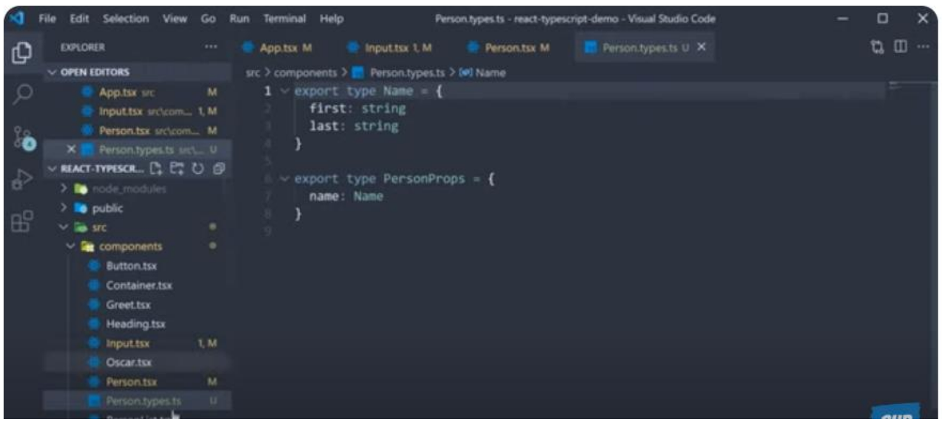


```
export type PersonProps = {
  name: {
    first: string
    last: string
  }
}
```

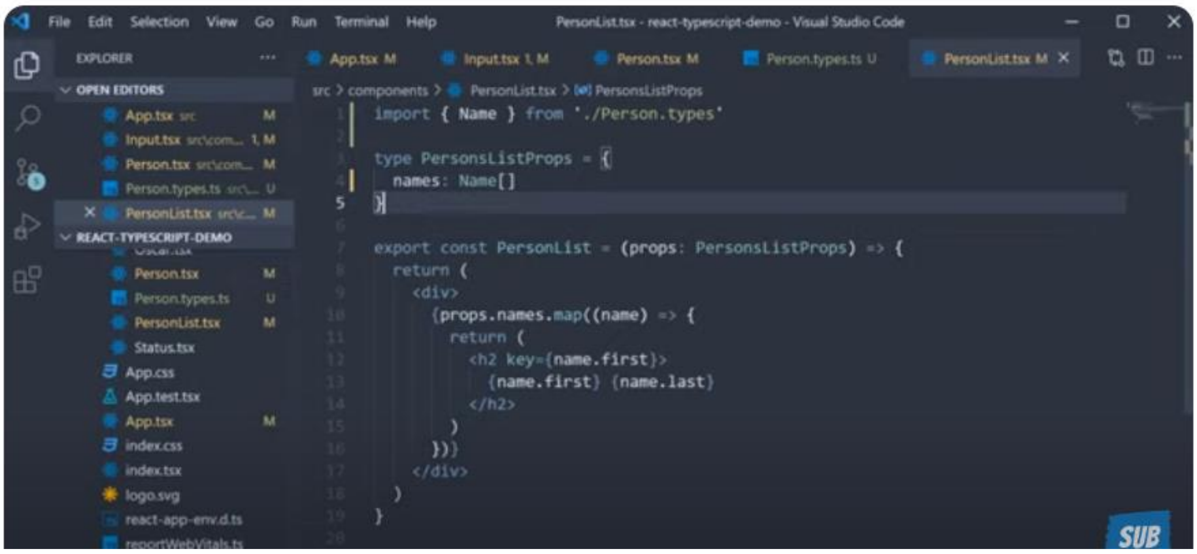


```
import { PersonProps } from './Person.types'

export const Person = (props: PersonProps) => {
  return (
    <div>
      {props.name.first} {props.name.last}
    </div>
  )
}
```

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



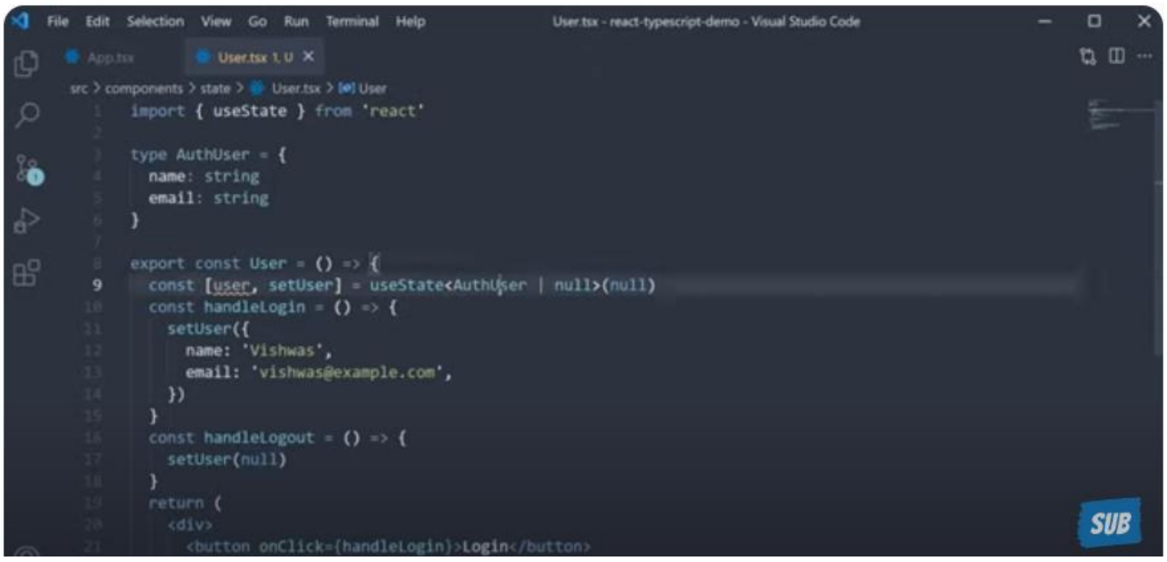
```
export type Name = {
  first: string
  last: string
}

export type PersonProps = {
  name: Name
}
```



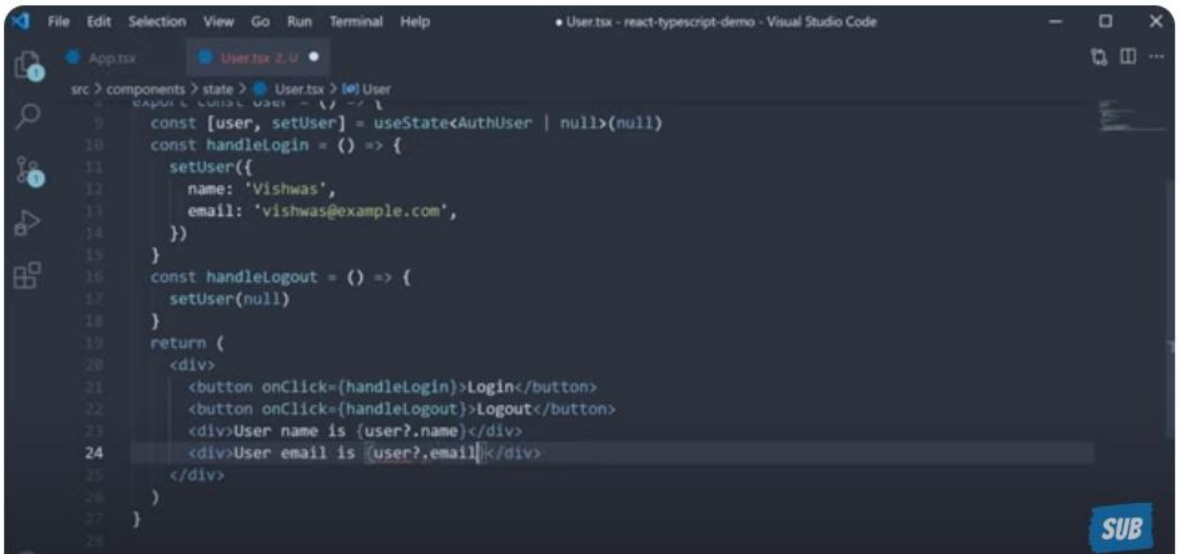
```
import { Name } from './Person.types'

type PersonsListProps = {
  names: Name[]
}

export const PersonList = (props: PersonsListProps) => {
  return (
    <div>
      {props.names.map((name) => {
        return (
          <h2 key={name.first}>
            {name.first} {name.last}
          </h2>
        )
      })}
    </div>
  )
}
```

useState Hook:-

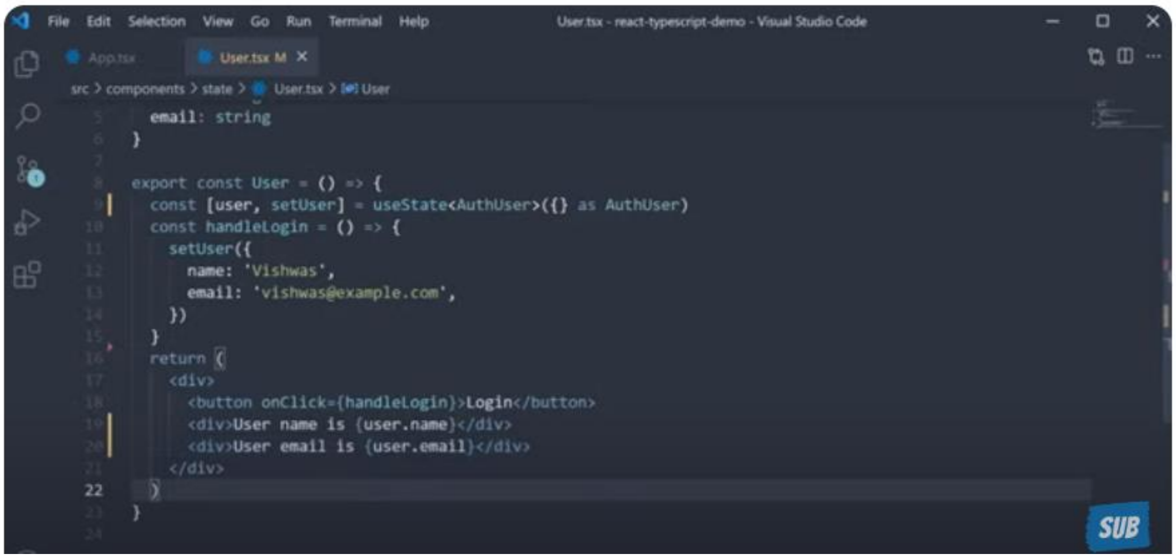
useState Future Value:-



```
import { useState } from 'react'

type AuthUser = {
  name: string
  email: string
}

export const User = () => {
  const [user, setUser] = useState<AuthUser | null>(null)
  const handleLogin = () => {
    setUser({
      name: 'Vishwas',
      email: 'vishwas@example.com',
    })
  }
  const handleLogout = () => {
    setUser(null)
  }
  return (
    <div>
      <button onClick={handleLogin}>Login</button>
```

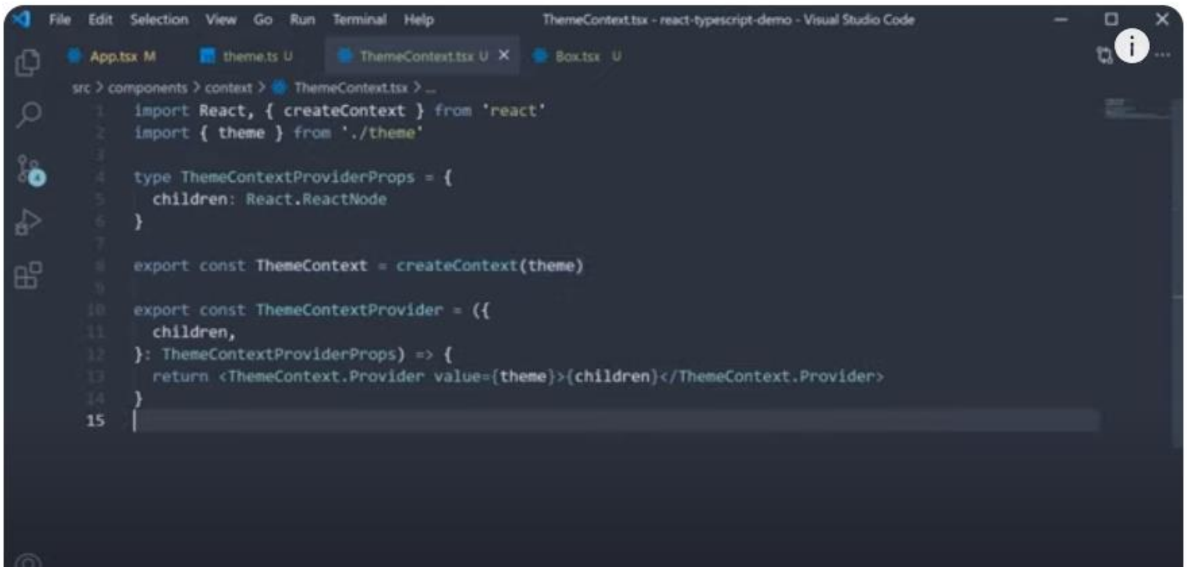

```tsx
  const [user, setUser] = useState<AuthUser | null>(null)
  const handleLogin = () => {
    setUser({
      name: 'Vishwas',
      email: 'vishwas@example.com',
    })
  }
  const handleLogout = () => {
    setUser(null)
  }
  return (
    <div>
      <button onClick={handleLogin}>Login</button>
      <button onClick={handleLogout}>Logout</button>
      <div>User name is {user?.name}</div>
      <div>User email is {user?.email}</div>
    </div>
  )
}
```

useState Type Assertion:-


```tsx
  email: string
}

export const User = () => {
  const [user, setUser] = useState<AuthUser>({} as AuthUser)
  const handleLogin = () => {
    setUser({
      name: 'Vishwas',
      email: 'vishwas@example.com',
    })
  }
  return (
    <div>
      <button onClick={handleLogin}>Login</button>
      <div>User name is {user.name}</div>
      <div>User email is {user.email}</div>
    </div>
  )
}
```

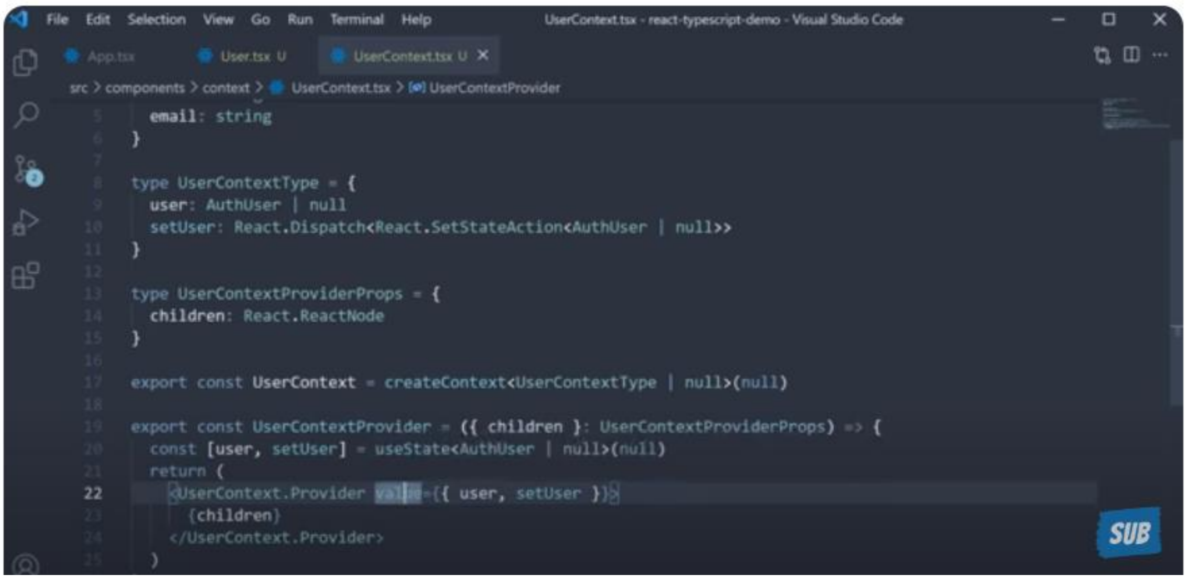
Use Reducer Hook:-

useContext Hook:-


File Edit Selection View Go Run Terminal Help
ThemeContext.tsx - react-typescript-demo - Visual Studio Code
App.tsx M
theme.ts U
ThemeContext.tsx U
Box.tsx U
src > components > context > ThemeContext.tsx > ...

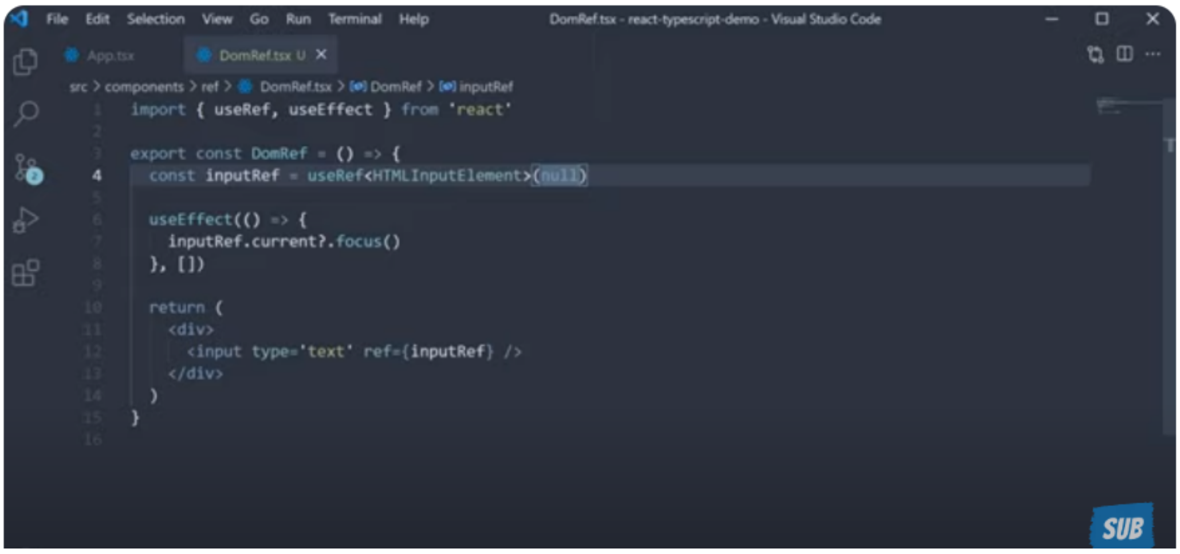
```tsx
import React, { createContext } from 'react'
import { theme } from './theme'

type ThemeContextProviderProps = {
  children: React.ReactNode
}

export const ThemeContext = createContext(theme)

export const ThemeContextProvider = ({
  children,
}: ThemeContextProviderProps) => {
  return <ThemeContext.Provider value={theme}>{children}</ThemeContext.Provider>
}
```

useContext Future Value:-


File Edit Selection View Go Run Terminal Help
UserContext.tsx - react-typescript-demo - Visual Studio Code
App.tsx
User.tsx U
UserContext.tsx U
src > components > context > UserContext.tsx > UserContextProvider
SUB

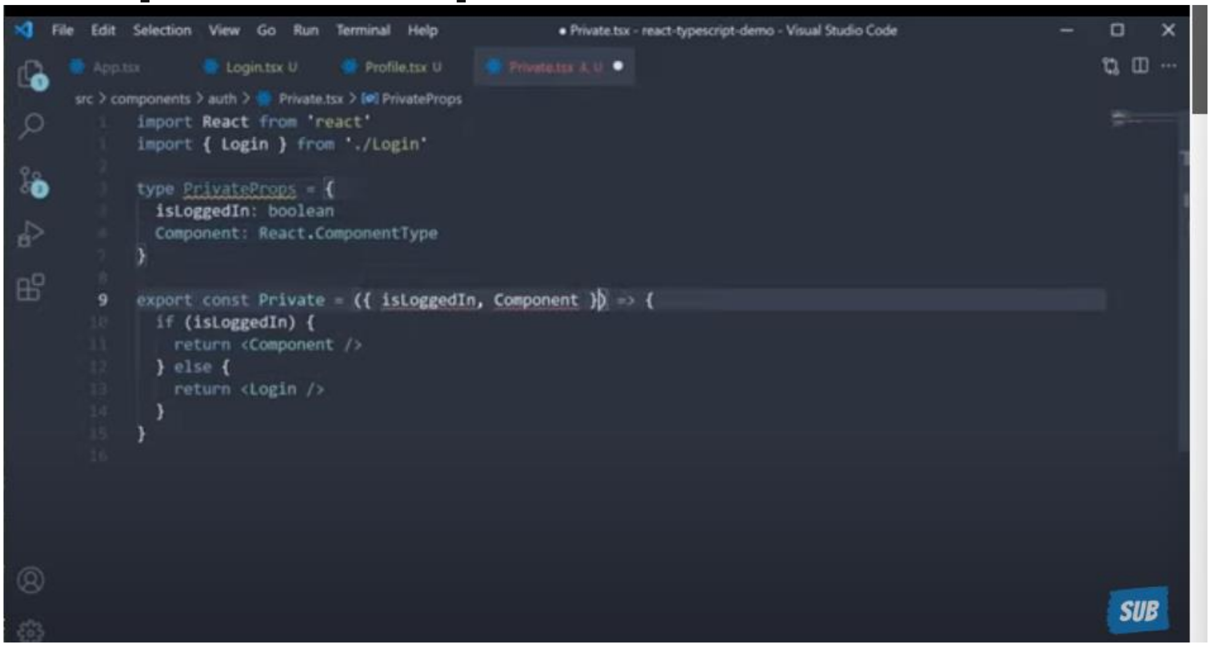
```tsx
  email: string
}

type UserContextType = {
  user: AuthUser | null
  setUser: React.Dispatch<React.SetStateAction<AuthUser | null>>
}

type UserContextProviderProps = {
  children: React.ReactNode
}

export const UserContext = createContext<UserContextType | null>(null)

export const UserContextProvider = ({ children }: UserContextProviderProps) => {
  const [user, setUser] = useState<AuthUser | null>(null)
  return (
    <UserContext.Provider value={{ user, setUser }}>
      {children}
    </UserContext.Provider>
  )
```

useRef Hook:-

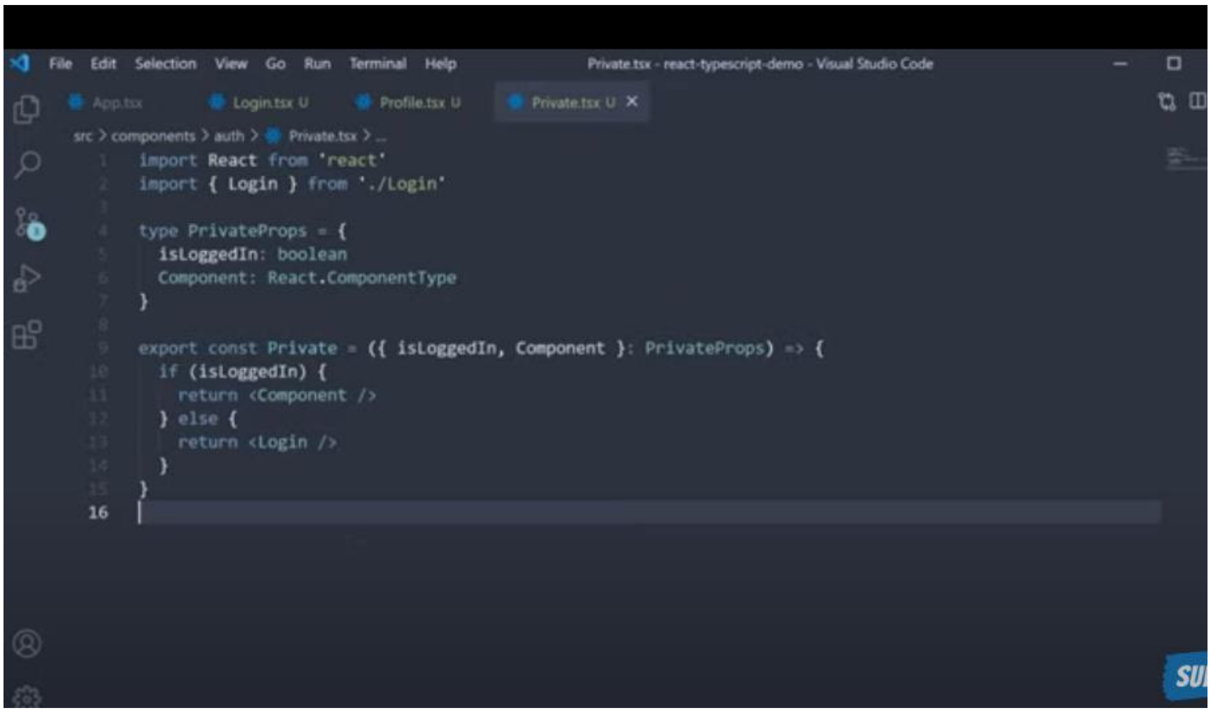


```
import { useRef, useEffect } from 'react'

export const DomRef = () => {
  const inputRef = useRef<HTMLInputElement>(null)

  useEffect(() => {
    inputRef.current?.focus()
  }, [])

  return (
    <div>
      <input type='text' ref={inputRef} />
    </div>
  )
}
```

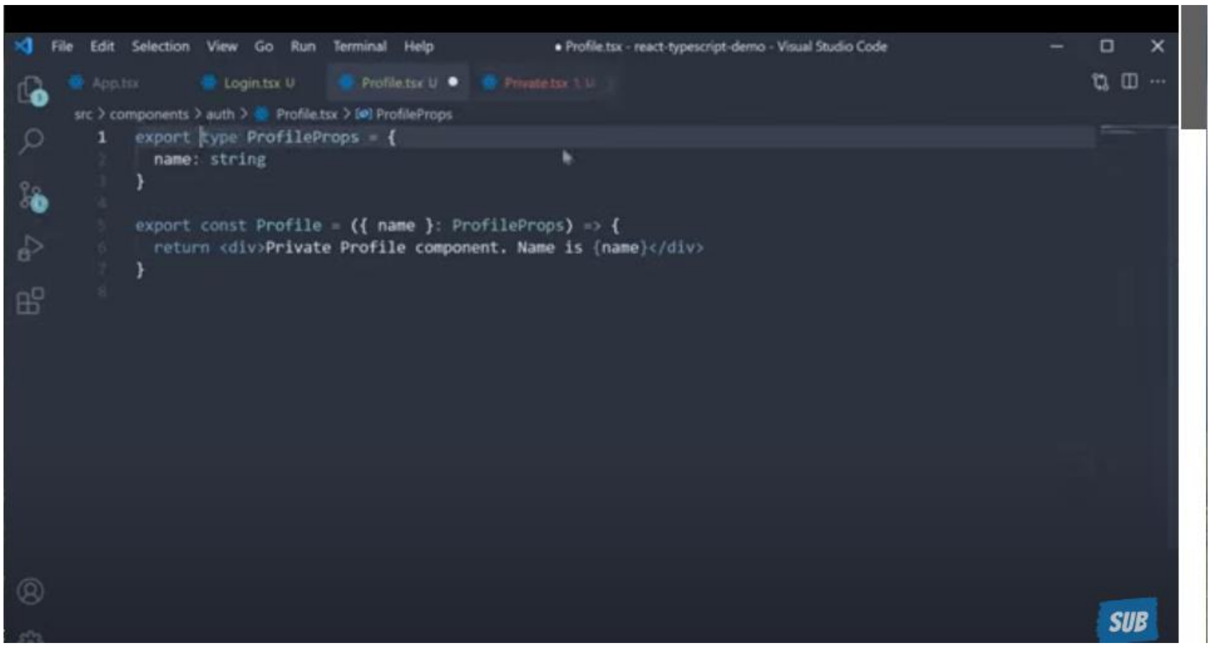
Component Prop:-



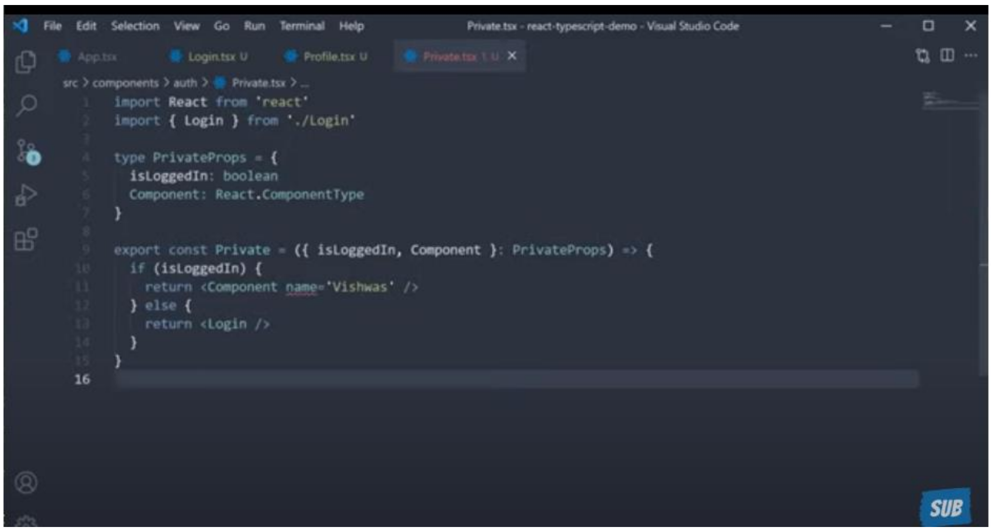
```
import React from 'react'
import { Login } from './Login'

type PrivateProps = {
  isLoggedIn: boolean
  Component: React.ComponentType
}

export const Private = ({ isLoggedIn, Component }) => {
  if (isLoggedIn) {
    return <Component />
  } else {
    return <Login />
  }
}
```


File Edit Selection View Go Run Terminal Help
Private.tsx - react-typescript-demo - Visual Studio Code
App.tsx
Login.tsx U
Profile.tsx U
Private.tsx U
src > components > auth > Private.tsx > ...

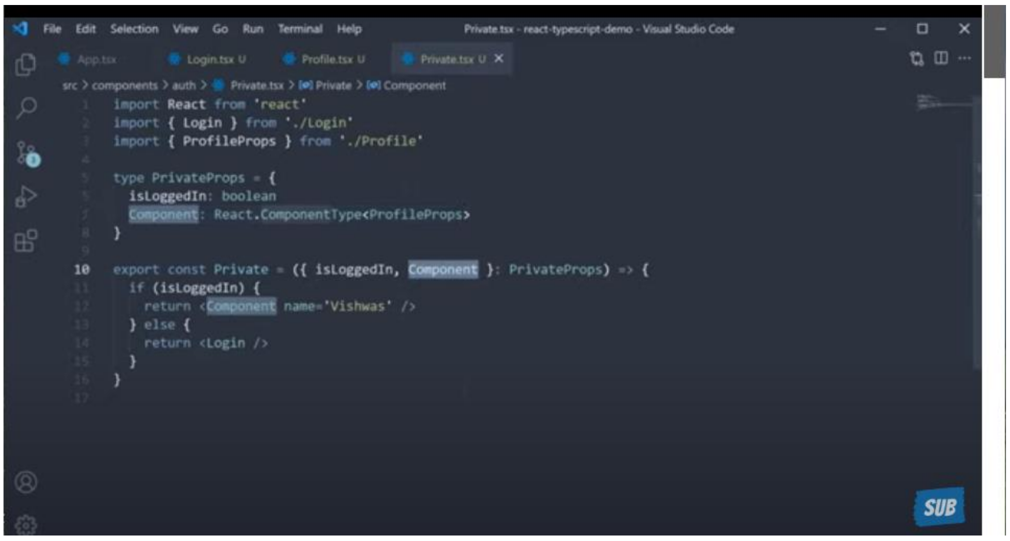
```tsx
import React from 'react'
import { Login } from './Login'

type PrivateProps = {
  isLoggedIn: boolean
  Component: React.ComponentType
}

export const Private = ({ isLoggedIn, Component }: PrivateProps) => {
  if (isLoggedIn) {
    return <Component />
  } else {
    return <Login />
  }
}

```


File Edit Selection View Go Run Terminal Help
● Profile.tsx - react-typescript-demo - Visual Studio Code
App.tsx
Login.tsx U
Profile.tsx U
Private.tsx 1, U
src > components > auth > Profile.tsx > ProfileProps

```tsx
export type ProfileProps = {
  name: string
}

export const Profile = ({ name }: ProfileProps) => {
  return <div>Private Profile component. Name is {name}</div>
}

```

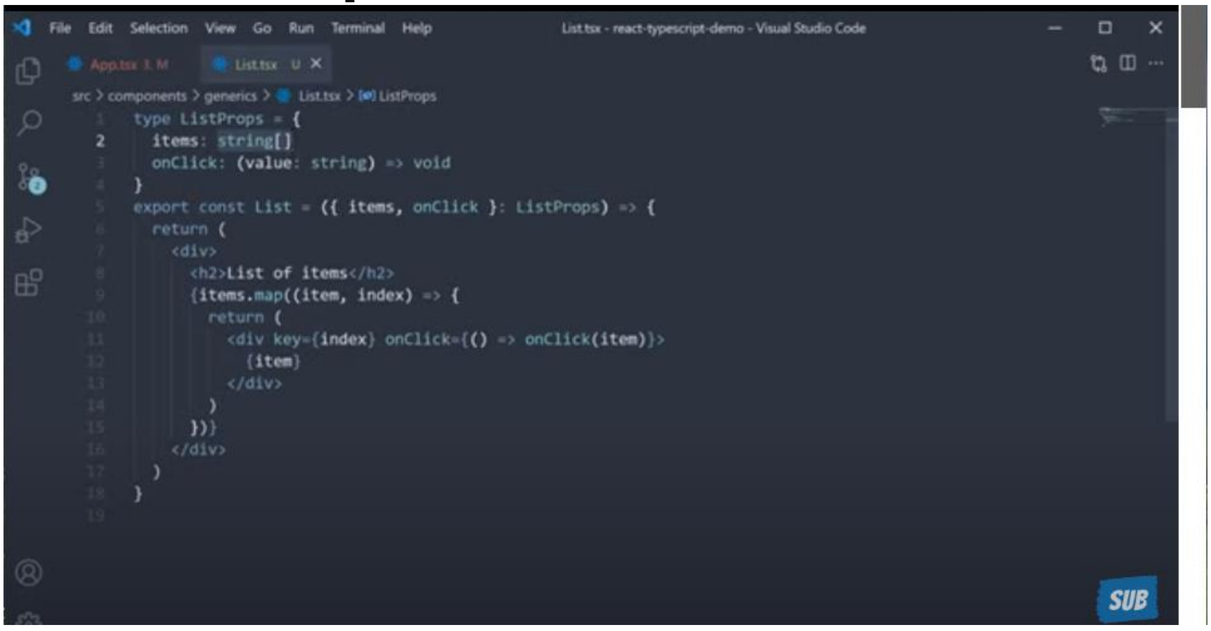


```tsx
import React from 'react'
import { Login } from './Login'

type PrivateProps = {
  isLoggedIn: boolean
  Component: React.ComponentType
}

export const Private = ({ isLoggedIn, Component }: PrivateProps) => {
  if (isLoggedIn) {
    return <Component name='Vishwas' />
  } else {
    return <Login />
  }
}
```

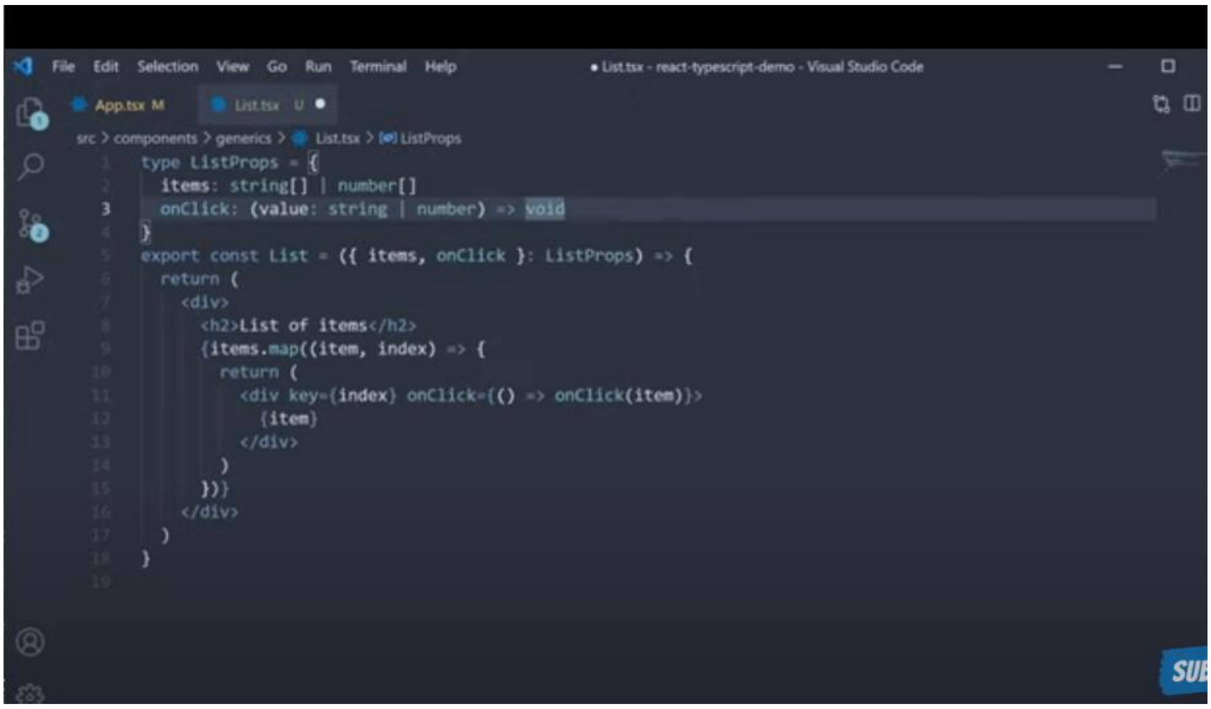


```tsx
import React from 'react'
import { Login } from './Login'
import { ProfileProps } from './Profile'

type PrivateProps = {
  isLoggedIn: boolean
  Component: React.ComponentType<ProfileProps>
}

export const Private = ({ isLoggedIn, Component }: PrivateProps) => {
  if (isLoggedIn) {
    return <Component name='Vishwas' />
  } else {
    return <Login />
  }
}
```

Generic Props:-

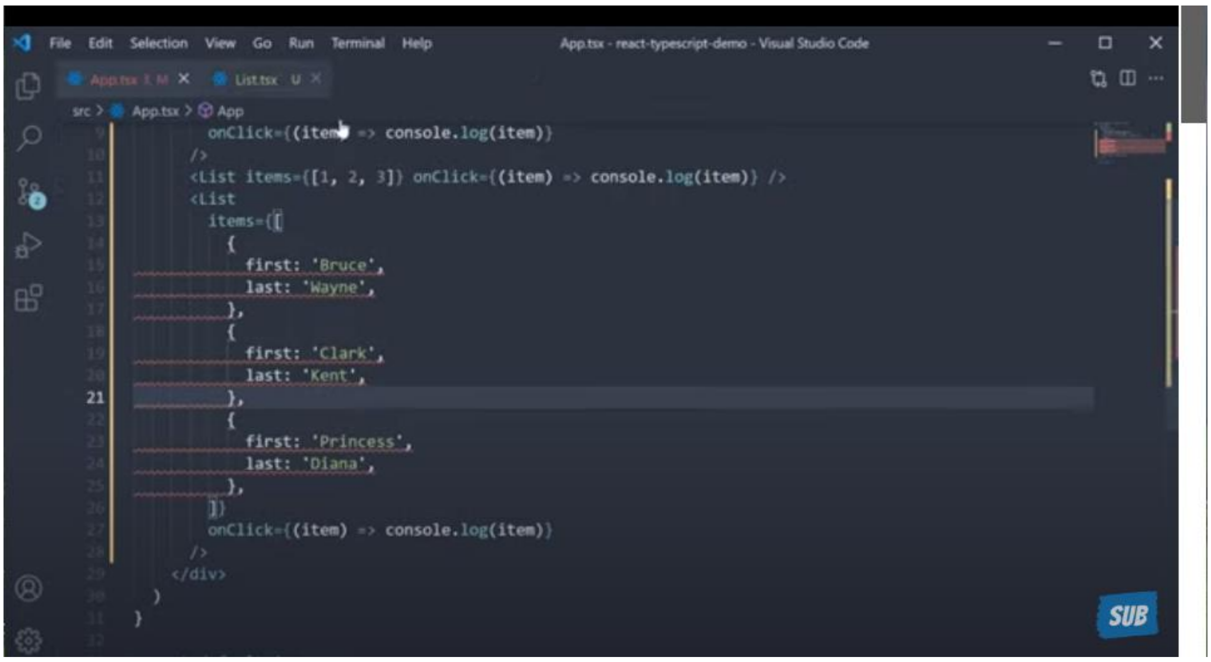


```tsx
type ListProps = {
  items: string[]
  onClick: (value: string) => void
}
export const List = ({ items, onClick }: ListProps) => {
  return (
    <div>
      <h2>List of items</h2>
      {items.map((item, index) => {
        return (
          <div key={index} onClick={() => onClick(item)}>
            {item}
          </div>
        )
      })}
    </div>
  )
}
```



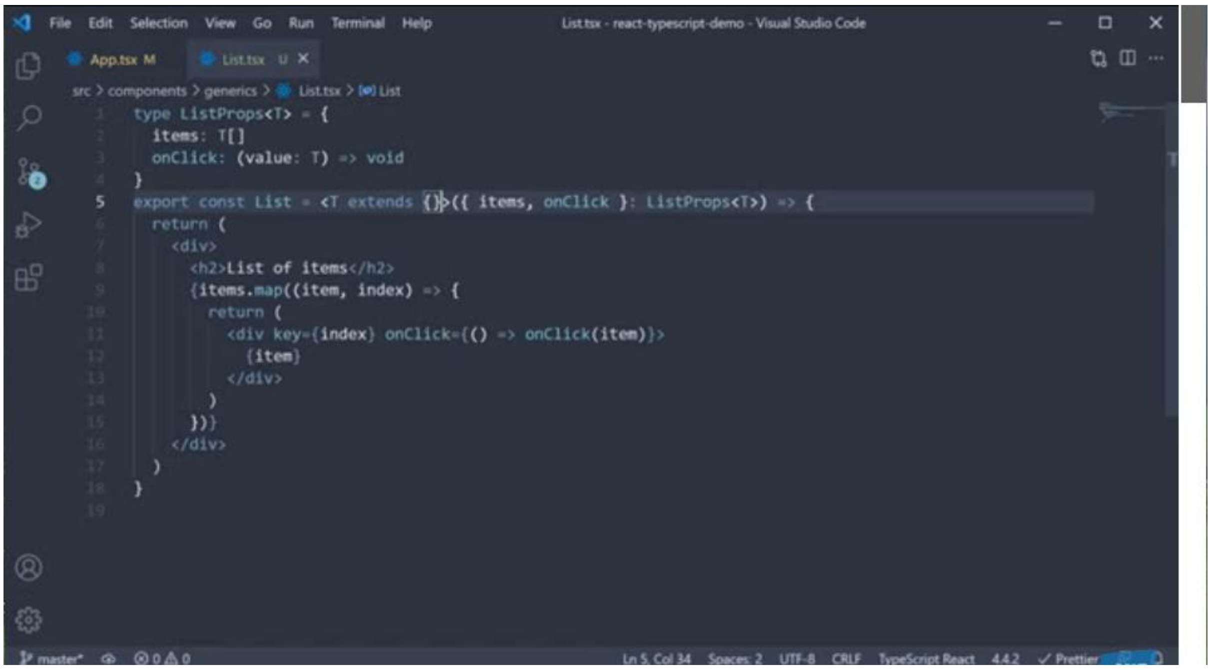
File Edit Selection View Go Run Terminal Help
• List.tsx - react-typescript-demo - Visual Studio Code
App.tsx M
List.tsx U
src > components > generics > List.tsx > ListProps

```
type ListProps = {
  items: string[] | number[]
  onClick: (value: string | number) => void
}
export const List = ({ items, onClick }: ListProps) => {
  return (
    <div>
      <h2>List of items</h2>
      {items.map((item, index) => {
        return (
          <div key={index} onClick={() => onClick(item)}>
            {item}
          </div>
        )
      })}
    </div>
  )
}
```

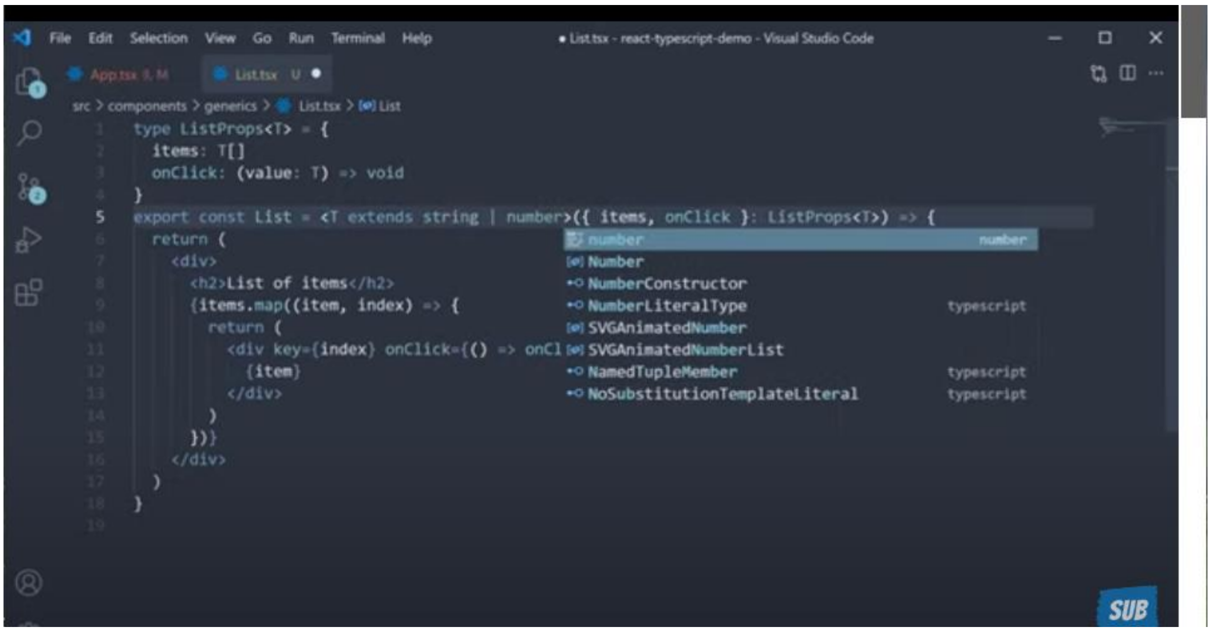


File Edit Selection View Go Run Terminal Help
App.tsx - react-typescript-demo - Visual Studio Code
App.tsx 3, M
List.tsx U
src > App.tsx > App

```
        onClick={(item) => console.log(item)}
      />
      <List items={[1, 2, 3]} onClick={(item) => console.log(item)} />
      <List
        items={[
          {
            first: 'Bruce',
            last: 'Wayne',
          },
          {
            first: 'Clark',
            last: 'Kent',
          },
          {
            first: 'Princess',
            last: 'Diana',
          },
        ]}
        onClick={(item) => console.log(item)}
      />
    </div>
  )
}
```

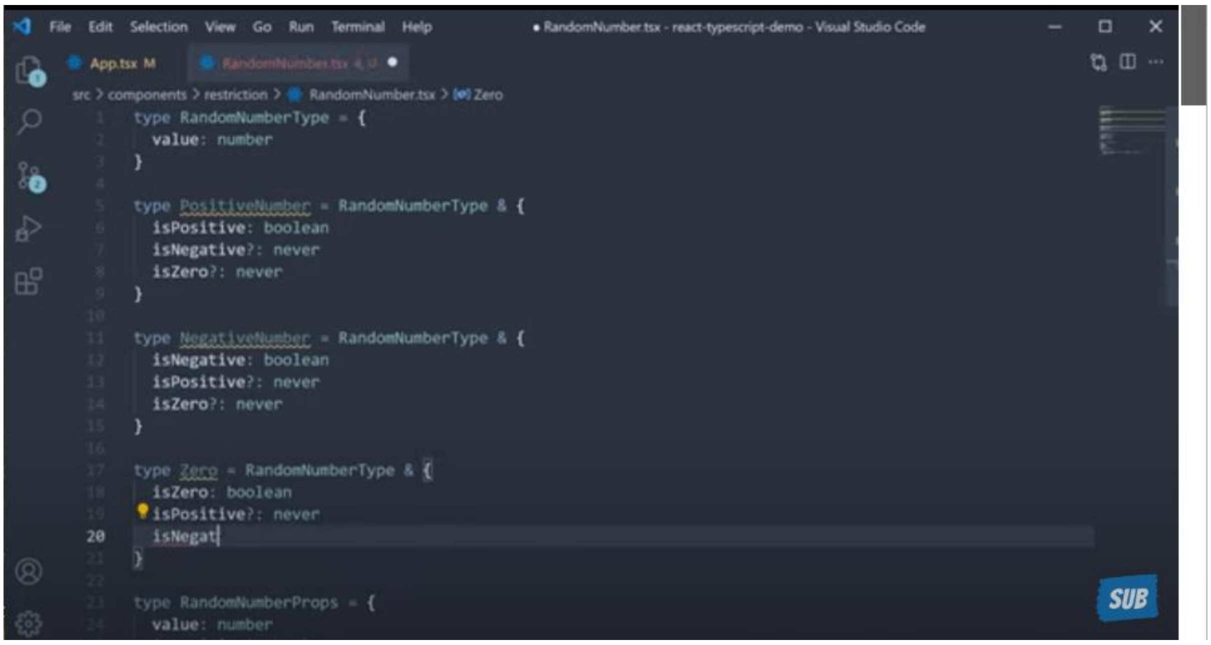


```
type ListProps<T> = {
  items: T[]
  onClick: (value: T) => void
}
export const List = <T extends {}>({ items, onClick }: ListProps<T>) => {
  return (
    <div>
      <h2>List of items</h2>
      {items.map((item, index) => {
        return (
          <div key={index} onClick={() => onClick(item)}>
            {item}
          </div>
        )
      })}
    </div>
  )
}
```

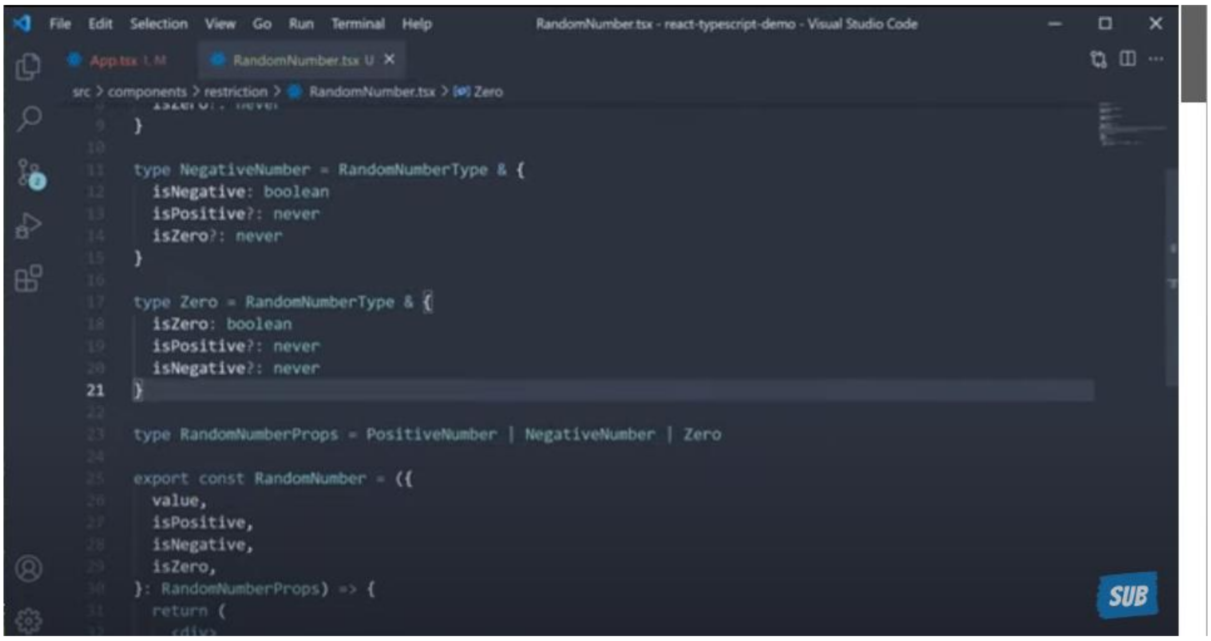


```
type ListProps<T> = {
  items: T[]
  onClick: (value: T) => void
}
export const List = <T extends string | number>({ items, onClick }: ListProps<T>) => {
  return (
    <div>
      <h2>List of items</h2>
      {items.map((item, index) => {
        return (
          <div key={index} onClick={() => onCl
            {item}
          </div>
        )
      })}
    </div>
  )
}
```

Restricting Props:-

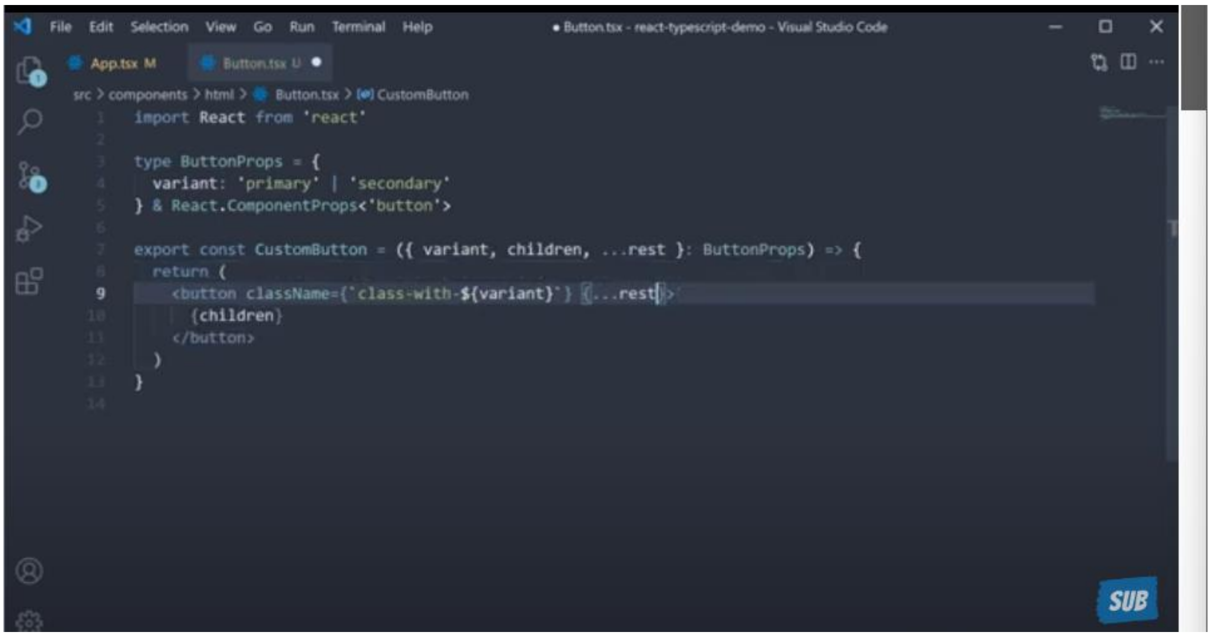

File Edit Selection View Go Run Terminal Help
• RandomNumber.tsx - react-typescript-demo - Visual Studio Code
App.tsx M
RandomNumber.tsx 4, U
src > components > restriction > RandomNumber.tsx > Zero
SUB

```
type RandomNumberType = {
  value: number
}

type PositiveNumber = RandomNumberType & {
  isPositive: boolean
  isNegative?: never
  isZero?: never
}

type NegativeNumber = RandomNumberType & {
  isNegative: boolean
  isPositive?: never
  isZero?: never
}

type Zero = RandomNumberType & {
  isZero: boolean
  isPositive?: never
  isNegat
}

type RandomNumberProps = {
  value: number
```

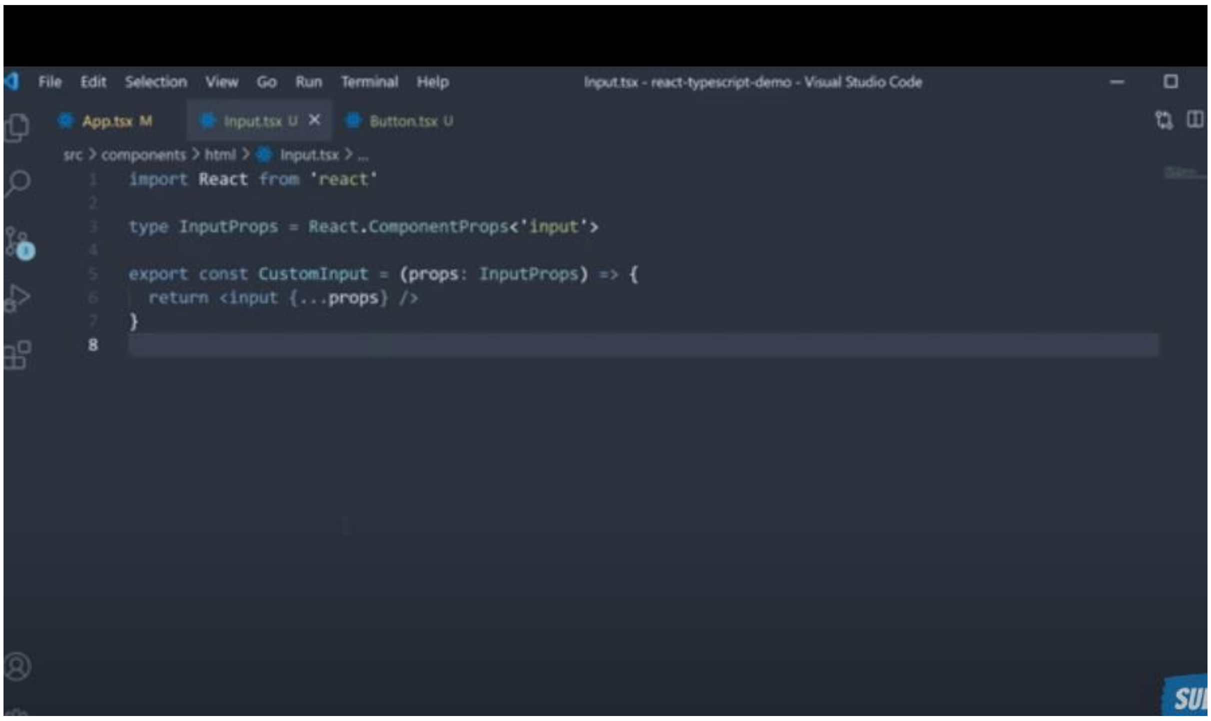

File Edit Selection View Go Run Terminal Help
RandomNumber.tsx - react-typescript-demo - Visual Studio Code
App.tsx 1, M
RandomNumber.tsx U
src > components > restriction > RandomNumber.tsx > Zero
SUB

```
}

type NegativeNumber = RandomNumberType & {
  isNegative: boolean
  isPositive?: never
  isZero?: never
}

type Zero = RandomNumberType & {
  isZero: boolean
  isPositive?: never
  isNegative?: never
}

type RandomNumberProps = PositiveNumber | NegativeNumber | Zero

export const RandomNumber = ({
  value,
  isPositive,
  isNegative,
  isZero,
}: RandomNumberProps) => {
  return (
```

Wrapping HTML Elements:-

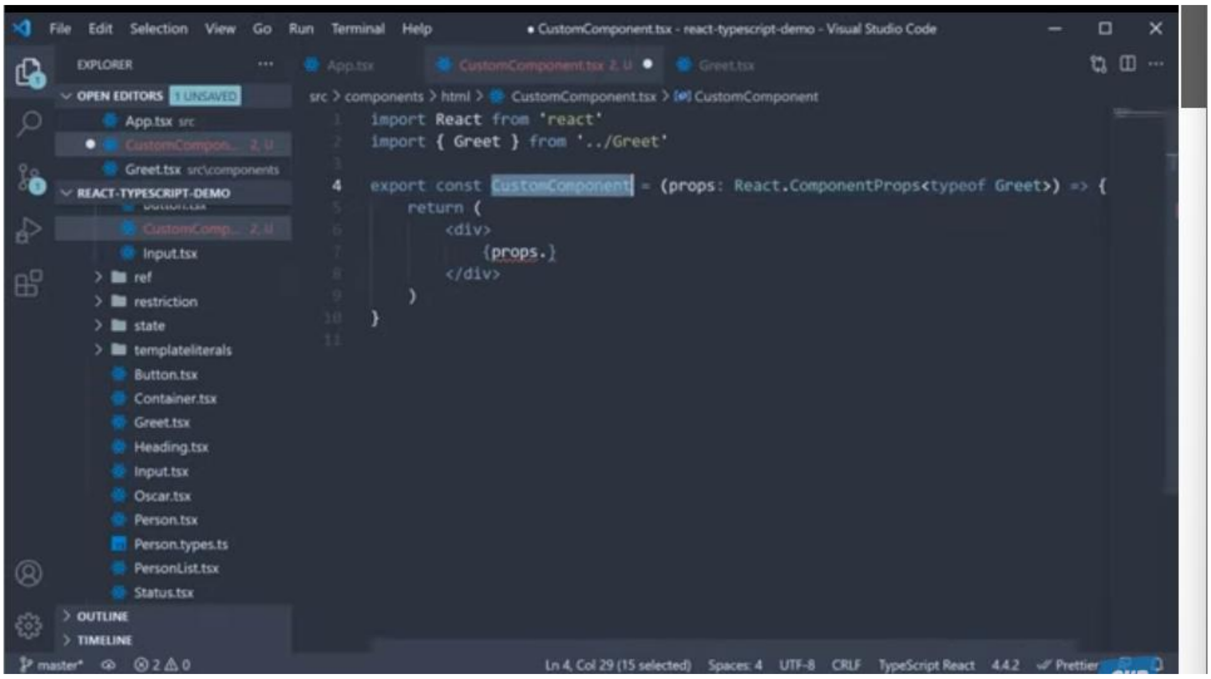


```tsx
import React from 'react'

type ButtonProps = {
  variant: 'primary' | 'secondary'
} & React.ComponentProps<'button'>

export const CustomButton = ({ variant, children, ...rest }: ButtonProps) => {
  return (
    <button className={`class-with-${variant}`} {...rest}>
      {children}
    </button>
  )
}
```



```tsx
import React from 'react'

type InputProps = React.ComponentProps<'input'>

export const CustomInput = (props: InputProps) => {
  return <input {...props} />
}
```

Extracting a Components Prop Types:-



```
import React from 'react'
import { Greet } from '../Greet'

export const CustomComponent = (props: React.ComponentProps<typeof Greet>) => {
    return (
        <div>
            {props.}
        </div>
    )
}
```