# What Can You Do in a .tsx File?

.tsx = TypeScript + JSX  
Used in React apps to combine the power of JSX (HTML in JS) with TypeScript's type safety.

## 1. Write React Components

function Hello() {  
 return <h1>Hello World</h1>;  
}  
✅ You can return JSX (HTML-like syntax)

## 2. Use TypeScript Types

type Props = { name: string };  
  
function Welcome({ name }: Props) {  
 return <h2>Hello, {name}</h2>;  
}  
✅ Type safety = fewer bugs!

## 3. Use Hooks

import { useState } from "react";  
  
function Counter() {  
 const [count, setCount] = useState<number>(0);  
  
 return (  
 <button onClick={() => setCount(count + 1)}>  
 Count: {count}  
 </button>  
 );  
}  
✅ Typed state with hooks

## 4. Add Event Handlers with Types

function Clicker() {  
 const handleClick = (e: React.MouseEvent<HTMLButtonElement>) => {  
 console.log("Button clicked");  
 };  
  
 return <button onClick={handleClick}>Click me</button>;  
}  
✅ Type-safe event handlers

## 5. Use External Components & Props

import MyButton from "./MyButton";  
  
<MyButton label="Click" onClick={() => {}} />  
✅ TypeScript will warn you if props are wrong

## 6. Use Conditional Rendering and Loops

{isLoggedIn ? <p>Welcome</p> : <p>Please log in</p>}  
  
{users.map((user) => (  
 <div key={user.id}>{user.name}</div>  
))}  
✅ Full logic + rendering supported

## 7. Use Context and Redux (Typed)

const user = useContext(UserContext); // typed context  
✅ Better structure than plain JS

## 8. Export / Import Components

export default function Header() {  
 return <header>My App</header>;  
}  
✅ Same import/export as JavaScript

## Summary Table

|  |  |
| --- | --- |
| Feature | Supported in .tsx? |
| React JSX | ✅ Yes |
| Type annotations (props, state) | ✅ Yes |
| useState, useEffect, etc. | ✅ Yes |
| Type-safe events & context | ✅ Yes |
| Exporting/importing components | ✅ Yes |

# Want a Full Real-World Example .tsx File to Study?

Here's a basic but complete example of a .tsx component using TypeScript, JSX, state, props, and types:

import React, { useState } from "react";  
  
type Props = {  
 initialCount?: number;  
};  
  
const Counter: React.FC<Props> = ({ initialCount = 0 }) => {  
 const [count, setCount] = useState<number>(initialCount);  
  
 const increment = () => setCount(prev => prev + 1);  
 const decrement = () => setCount(prev => prev - 1);  
  
 return (  
 <div>  
 <h2>Counter: {count}</h2>  
 <button onClick={increment}>+</button>  
 <button onClick={decrement}>-</button>  
 </div>  
 );  
};  
  
export default Counter;