

Why TypeScript?

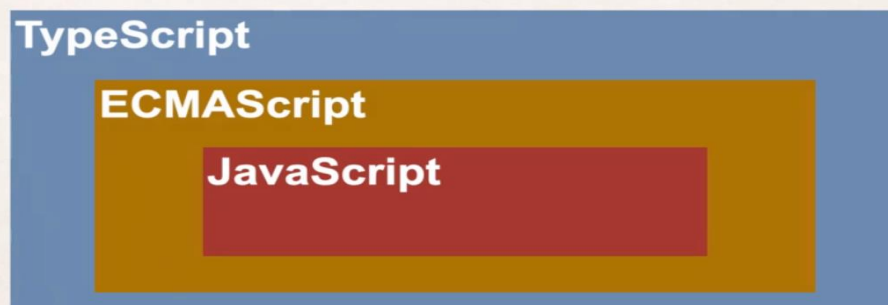
React Development

- For React development, we can develop using various languages
 - **JavaScript**: extremely popular programming language
 - **ECMAScript**: standardized version of JavaScript (ES6, ES9, ...)
 - **TypeScript**: adds optional types to JavaScript
- **TypeScript** is becoming extremely popular for React

The relationship between Typescript, JavaScript and ECMAScript

Relationships

- TypeScript is a superset of JavaScript and ECMAScript



TypeScript

- **FAQ: Why do many React developers use TypeScript?**
- Strongly-typed language with compile time checking and IDE support
- Increased developer productivity and efficiency
- Docs, online blogs and tutorials use TypeScript for coding examples

Convert JavaScript to TypeScript

- To fully see the power of TypeScript running on a React app..
- Let's change our todo application from a JavaScript project to a TypeScript project.

JavaScript

`todo-app.js`



TypeScript

`todo-app.tsx`

Install TypeScript:

Step 1: Install TypeScript

- We will first need to download TypeScript to an already created React Project.

Project Terminal

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

Create tsconfig file:

Step 2: Create a tsconfig file

- We will need a tsconfig.json file to use with our TypeScript.
- Specifies the root of the project.
- Specifies compiler options that are required to properly build a TS project.

Project Terminal

```
npx tsc --init
```

Add Types to param and variable:

Step 3: Add types to each param and variable

- TypeScript needs to validate each param and variable within our app

App.js

```
const addToDo (
  description,
  assigned
)
```

App.tsx

```
const addToDo (
  description: string,
  assigned: string
)
```

Create TodoModel.ts

Step 4: Create TodoModel.ts

- There is a new type we need to create: TodoModel.

```
Class TodoModel {
  rowNumber: number;
  rowDescription: string;
  rowAssigned: string;

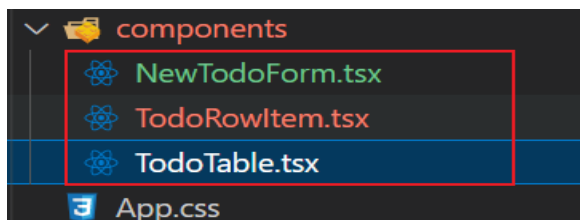
  constructor (
    rowNumber: number,
    rowDescription: string,
    rowAssigned: string
  ) {
    this.rowNumber = rowNumber;
    this.rowDescription = rowDescription;
    this.rowAssigned = rowAssigned;
  }
}
```

TypeScript Installation

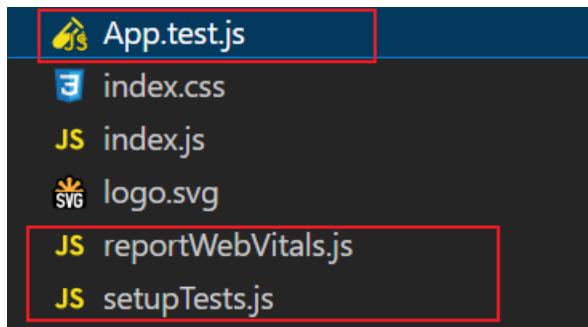
Open a new terminal, run this command:

```
PS C:\react-to-demo\react-todos> npm install --save typescript @types/node @types/react @types/react-dom @types/jest
```

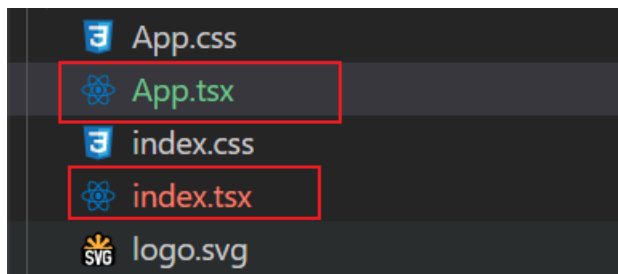
Change all the components js file name to tsx.



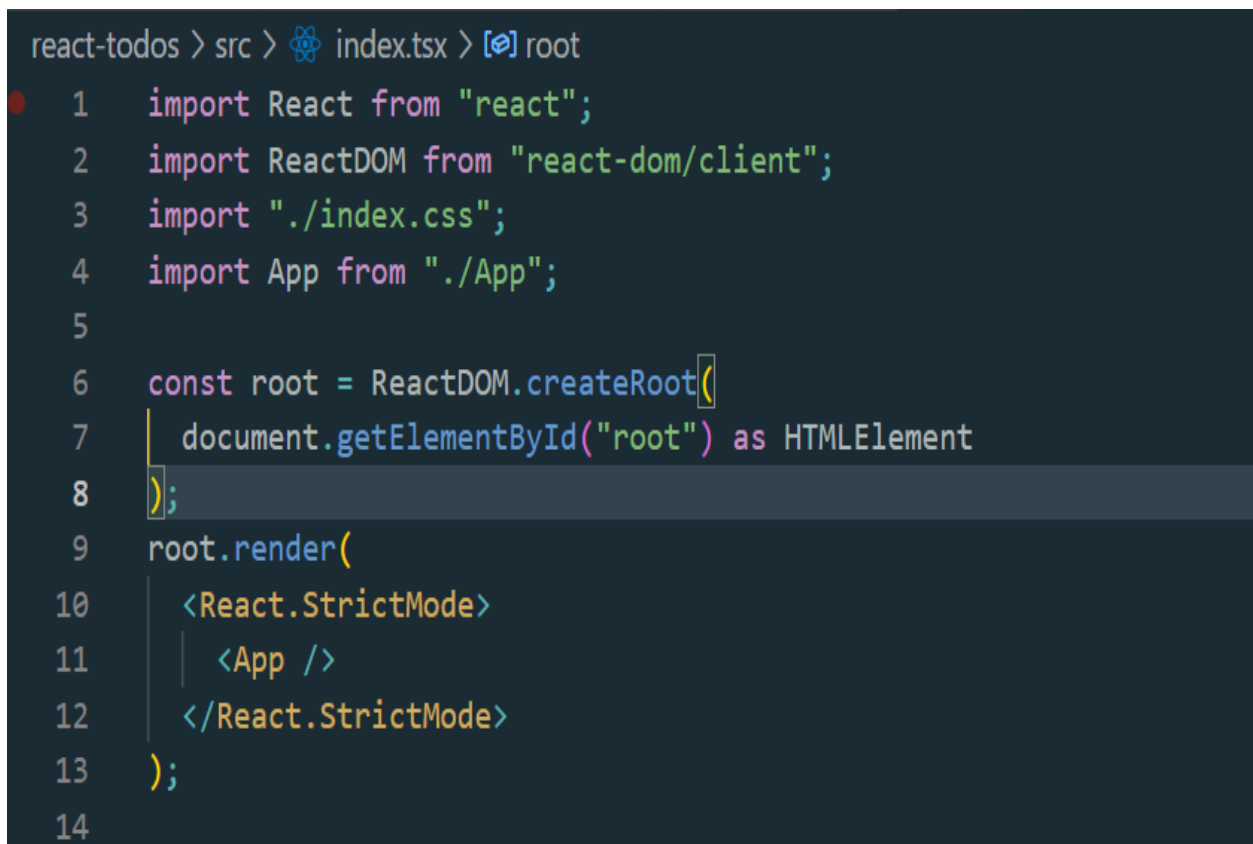
Delete these three js file



Also change these two file name from js to tsx



Modify the index.tsx,



Fix the issue by adding an import statement in TodoTable.tsx

```
App.tsx U  NewTodoForm.tsx U  TodoTable.tsx U X  index.tsx U
react-todos > src > components > TodoTable.tsx > TodoTable
1  import React from "react";
2  import TodoRowItem from "../TodoRowItem";
3
4  function TodoTable(props) {
5    return (
6      <table className="table table-hover">
7        <thead>
8          <tr>
9            <th scope="col">#</th>
10           <th scope="col">Description</th>
11           <th scope="col">Assigned</th>
12           <th scope="col">Actions</th>
13         </tr>
14       </thead>
15       <tbody>
16         {props.todos.map((todo) => (
17           <TodoRowItem
18             key={todo.rowNumber}
19             rowNumber={todo.rowNumber}
20             rowDescription={todo.rowDescription}
21             rowAssigned={todo.rowAssigned}
22             deleteTodo={props.deleteTodo}
23           />
24         ))}
25       </tbody>
26     </table>

```

Do the same thing in TodoRowItem.tsx

```
react-todos > src > components > TodoRowItem.tsx > TodoRowItem
1  import React from "react";
2
3  function TodoRowItem(props) {
4    return (
5      <tr>
6        <th scope="row">{props.rowNumber}</th>
7        <td>{props.rowDescription}</td>
8        <td>{props.rowAssigned}</td>
9        <td>
10          <button
11            className="btn btn-danger"
12            onClick={() => props.deleteTodo(props.rowNumber)}
13          >
14            Delete
15          </button>
16        </td>
17      </tr>
18    </>;
19  }
20
21  export default TodoRowItem;
22

```

Try to run the application using “npm start” command in the terminal, we’ll get an error message

Compiled with problems:

ERROR in ./src/index.tsx 7:0-24

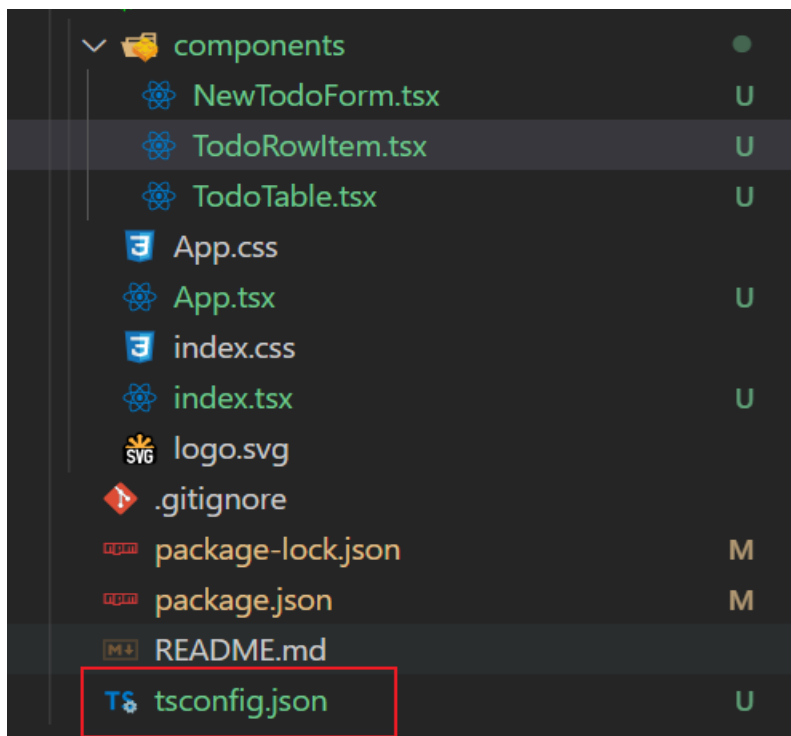
Module not found: Error: Can't resolve './App' in 'C:\react-to-demo\react-todos\src'

Stop the Server, run the command npx tsc --init

Terminate batch job (Y/N)? Y

PS C:\react-to-demo\react-todos> npx tsc --init

Then we can get a new file “tsconfig.json”



Open this file.

```
/* Language and Environment */
"target": "es2016",
// "lib": [],
// "jsx": "preserve",
// "experimentalDecorators": true,
// "emitDecoratorMetadata": true,
// "jsxFactory": "",
// "jsxFragmentFactory": "",
// "jsxImportSource": "",
// "reactNamespace": "",
// "noLib": true,
// "useDefineForClassFields": true,
// "moduleDetection": "auto",

/* Set the JavaScript language version for emitted JavaScript and include compatible library de
/* Specify a set of bundled library declaration files that describe the target runtime environm
/* Specify what JSX code is generated. */
/* Enable experimental support for TC39 stage 2 draft decorators. */
/* Emit design-type metadata for decorated declarations in source files. */
/* Specify the JSX factory function used when targeting React JSX emit, e.g. 'React.createElem
/* Specify the JSX Fragment reference used for fragments when targeting React JSX emit e.g. 'Re
/* Specify module specifier used to import the JSX factory functions when using 'jsx: react-js
/* Specify the object invoked for 'createElement'. This only applies when targeting 'react' JSX
/* Disable including any library files, including the default lib.d.ts. */
/* Emit ECMAScript-standard-compliant class fields. */
/* Control what method is used to detect module-format JS files. */
```


Uncomment this line, and change “preserve” to “react-jsx”

```
"jsx": "react-jsx"
```

Then save it, and run the application, we'll get a bunch of error.

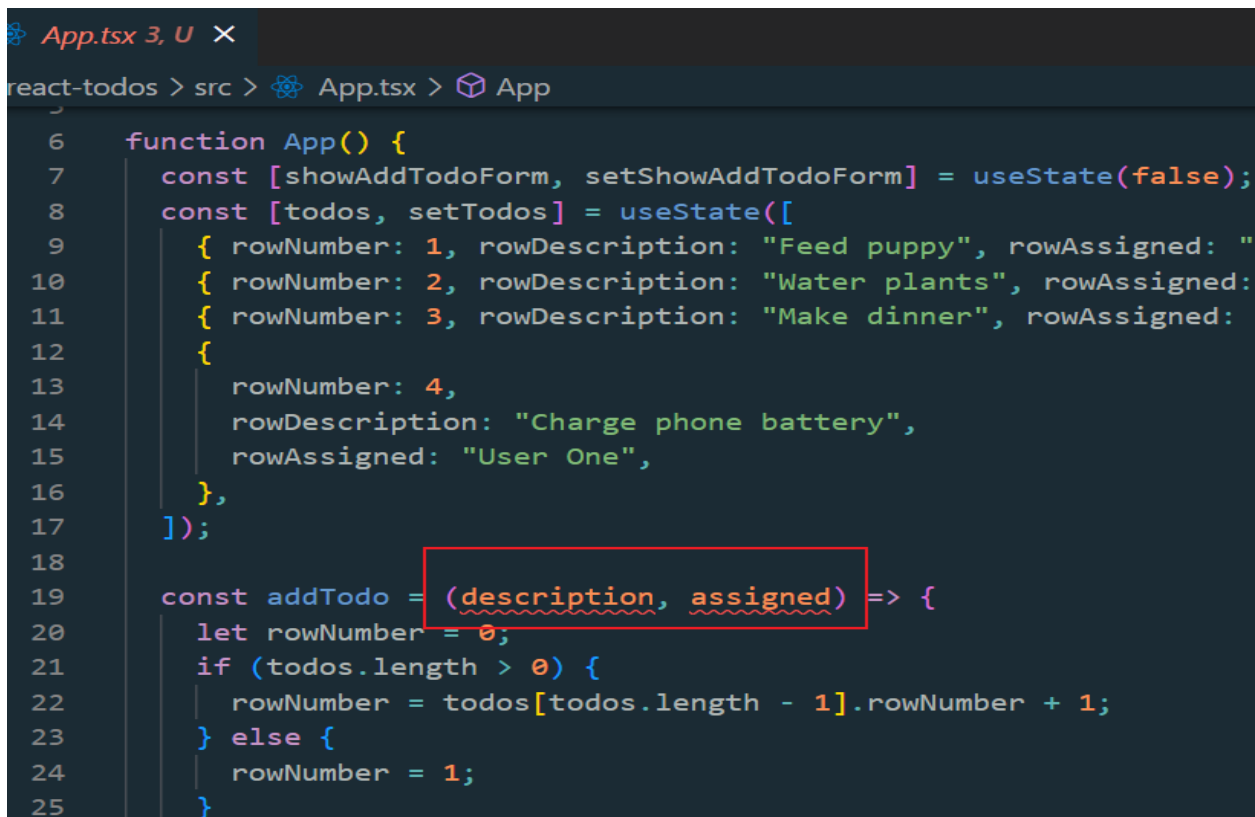


The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The error message is: **ERROR in src/components/ToDoTable.tsx:4:20**
TS7006: Parameter 'props' implicitly has an 'any' type.
The code snippet shows the beginning of the `ToDoTable` function, which takes `props` as an argument. The error points to the `props` parameter.

```
5 |     <tr>
6 |       <th scope="row">{props.rowNumber}</th>

ERROR in src/components/ToDoTable.tsx:4:20
TS7006: Parameter 'props' implicitly has an 'any' type.
2 | import TodoRowItem from "../TodoRowItem";
3 |
> 4 | function ToDoTable(props) {
    |                        ^^^^^^
5 |   return (
6 |     <table className="table table-hover">
7 |       <thead>
```

Open the app.tsx file. We can find some error



The screenshot shows the VS Code interface with the `App.tsx` file open. The code defines the `App` function, which uses `useState` to manage `showAddTodoForm` and `todos`. The `addTodo` function is defined as an arrow function, and the parameters `description` and `assigned` are highlighted with a red box.

```
App.tsx 3, U X
react-todos > src > App.tsx > App

6 | function App() {
7 |   const [showAddTodoForm, setShowAddTodoForm] = useState(false);
8 |   const [todos, setTodos] = useState([
9 |     { rowNumber: 1, rowDescription: "Feed puppy", rowAssigned: "
10 |    { rowNumber: 2, rowDescription: "Water plants", rowAssigned:
11 |    { rowNumber: 3, rowDescription: "Make dinner", rowAssigned:
12 |    {
13 |      rowNumber: 4,
14 |      rowDescription: "Charge phone battery",
15 |      rowAssigned: "User One",
16 |    },
17 |   ]);
18 |
19 |   const addTodo = (description, assigned) => {
20 |     let rowNumber = 0;
21 |     if (todos.length > 0) {
22 |       rowNumber = todos[todos.length - 1].rowNumber + 1;
23 |     } else {
24 |       rowNumber = 1;
25 |     }
26 |   }
27 | }
```

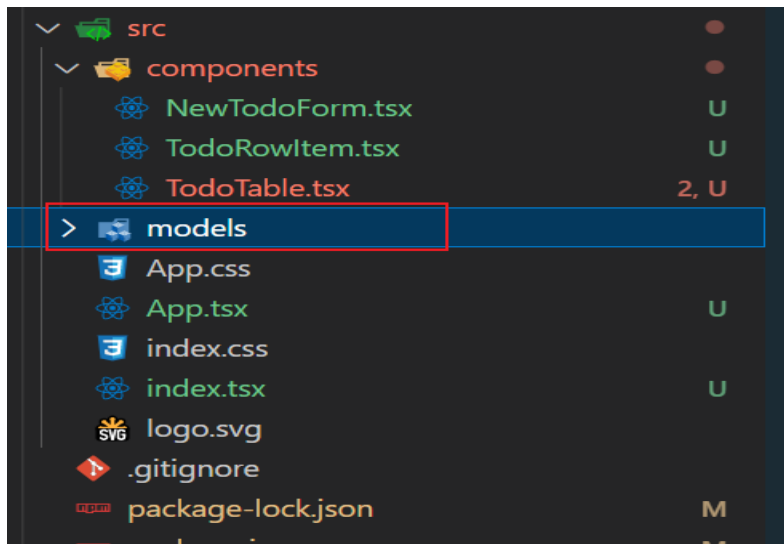
Give these variables types.

```
const addTodo = (description: string, assigned: string) => {  
  let rowNumber = 0;  
  if (todos.length > 0) {  
    rowNumber = todos[todos.length - 1].rowNumber + 1;  
  } else {  
    rowNumber = 1;  
  }  
  const newTodo = {  
    rowNumber: rowNumber,
```

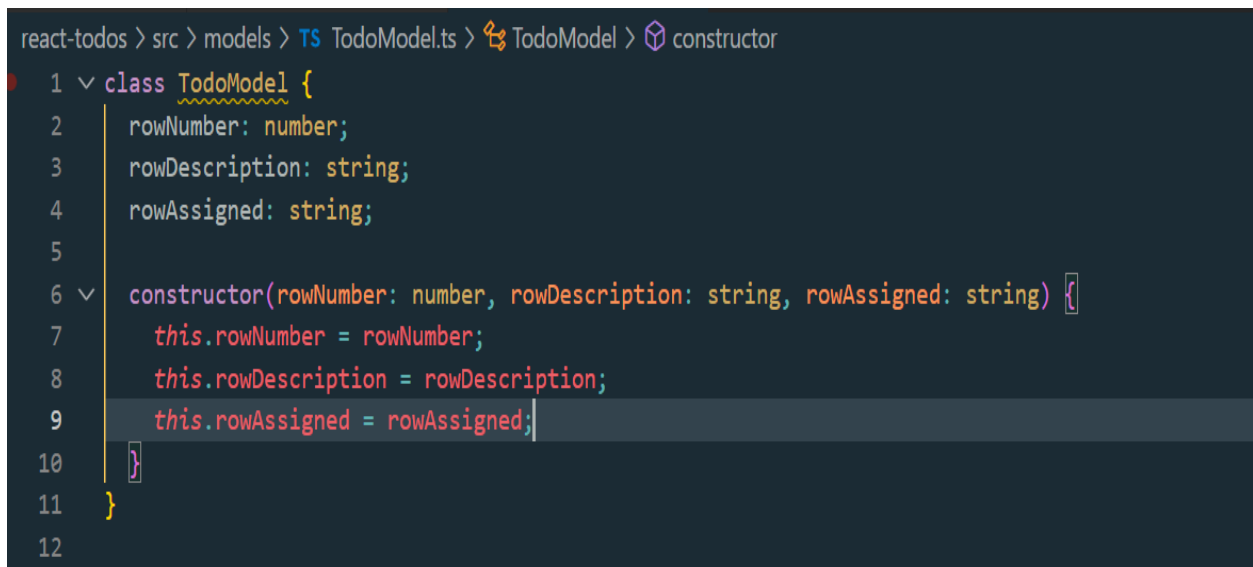
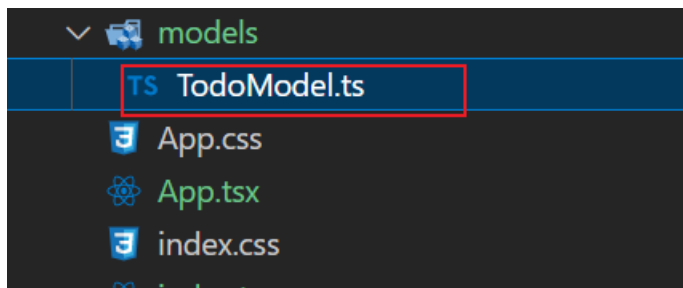
Then we open TodoTable.tsx. We still find some errors.

```
App.tsx U  TodoTable.tsx 2, U X  
react-todos > src > components > TodoTable.tsx > TodoTable > props.todos.map() callback  
1  import React from "react";  
2  import TodoRowItem from "../TodoRowItem";  
3  
4  function TodoTable(props) {  
5    return (  
6      <table className="table table-hover">  
7        <thead>  
8          <tr>  
9            <th scope="col">#</th>  
10           <th scope="col">Description</th>  
11           <th scope="col">Assigned</th>  
12           <th scope="col">Actions</th>  
13         </tr>  
14       </thead>  
15       <tbody>  
16         {props.todos.map((todo) => (  
17           <TodoRowItem  
18             key={todo.rowNumber}  
19             rowNumber={todo.rowNumber}  
20             rowDescription={todo.rowDescription}  
21             rowAssigned={todo.rowAssigned}  
22             deleteTodo={props.deleteTodo}  
23           </>  
24         )]}  
25       </tbody>  
26     </table>  
27   )
```


We need to create a model fold in the src directory



Under the models folder, we create a TodoModel.ts file



After create this TodoModel class, then we go to the TodoRowItem.tsx, modify it.

```
function TodoRowItem(props: {  
  rowNumber: number;  
  rowDescription: string;  
  rowAssigned: string;  
  deleteTodo: Function;  
}) {  
  return (  
    <tr>  
      <th scope="row">{props.rowNumber}</th>  
      <td>{props.rowDescription}</td>  
      <td>{props.rowAssigned}</td>  
      <td>  
        <button  
          className="btn btn-danger"  
          onClick={() => props.deleteTodo(props.rowNumber)}  
        >  
          Delete  
        </button>  
      </td>  
    </tr>  
  )  
}
```

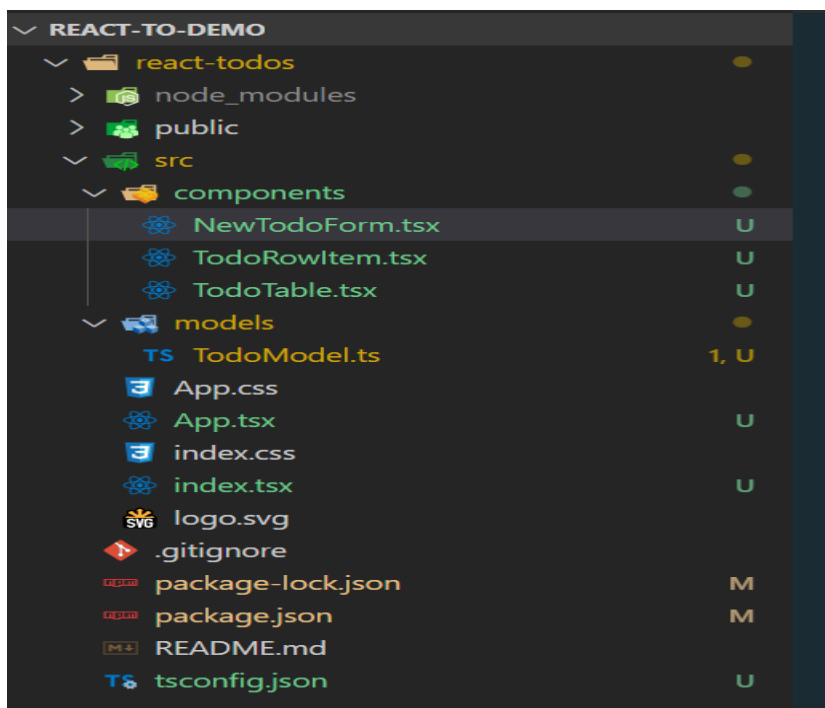
Then we need to modify the TodoTable.tsx

```
react-todos > src > components > TodoTable.tsx > TodoTable  
1 import React from "react";  
2 import TodoRowItem from "../TodoRowItem";  
3  
4 function TodoTable(props: { todos: TodoModel[]; deleteTodo: Function }) {  
5   return (  
6     <table className="table table-hover">  
7       <thead>  
8         <tr>  
9           <th scope="col">#</th>  
10          <th scope="col">Description</th>  
11          <th scope="col">Assigned</th>  
12          <th scope="col">Actions</th>  
13        </tr>  
14      </thead>  
15      <tbody>  
16        {props.todos.map((todo) => (  
17          <TodoRowItem
```

Also modify the NewTodoForm.tsx

```
react-todos > src > components > NewTodoForm.tsx > NewTodoForm
1  import React, { useState } from "react";
2
3  function NewTodoForm(props: { addToDo: Function }) {
4      const [description, setDescription] = useState("");
5      const [assigned, setAssigned] = useState("");
6
7      const submitTodo = () => {
8          if (description !== "" && assigned !== "") {
9              props.addToDo(description, assigned);
10             setAssigned("");
11             setDescription("");
12         }
13     }
14 }
```

So now there's no issue any more.




We can run the application, and get the page like this.

localhost:3000

Your Todo's			
#	Description	Assigned	Actions
1	Feed puppy	User One	<button>Delete</button>
2	Water plants	User Two	<button>Delete</button>
3	Make dinner	User One	<button>Delete</button>
4	Charge phone battery	User One	<button>Delete</button>

New Todo

Let's open the `TodoRowItem.tsx`. Change the function to `const`, add the `export` before it, and delete the `export` statement at the bottom.

```
react-todos > src > components >  TodoRowItem.tsx > ...
1   import React from "react";
2
3   export const TodoRowItem = (props: {
4     rowNum: number;
5     rowDescription: string;
6     rowAssigned: string;
7     deleteTodo: Function;
8   }) => {
9     return (
10      <tr>
11        <th scope="row">{props.rowNum}</th>
12        <td>{props.rowDescription}</td>
13        <td>{props.rowAssigned}</td>
14        <td>
15          <button
16            className="btn btn-danger"
17            onClick={() => props.deleteTodo(props.rowNum)}
18          >
19            Delete
20          </button>
21        </td>
22      </tr>
23    );
24  };
25
```

And change the import statement in TodoTable.tsx

```
react-todos > src > components > TodoTable.tsx > ...
1 import React from "react";
2 import { TodoRowItem } from "../TodoRowItem";
3
4 function TodoTable(props: { todos: TodoModel[]; deleteTodo: Function }) {
5   return (
6     <table className="table table-hover">
7       <thead>
8         <tr>
9           <th scope="col">#</th>
10          <th scope="col">Description</th>
11          <th scope="col">Assigned</th>
12          <th scope="col">Actions</th>
13        </tr>
14      </thead>
15      <tbody>
16        {props.todos.map((todo) => (
17          <TodoRowItem
18            key={todo.rowNumber}
19            rowNumber={todo.rowNumber}
20            rowDescription={todo.rowDescription}
21            rowAssigned={todo.rowAssigned}
22            deleteTodo={props.deleteTodo}
23          />
24        ))}
25      </tbody>
26    </table>
27  );
28 }
```

Modify the TodoRowItem.tsx

```
react-todos > src > components > TodoRowItem.tsx > [?] TodoRowItem
1 import React from "react";
2
3 export const TodoRowItem: React.FC<{
4   rowNumber: number;
5   rowDescription: string;
6   rowAssigned: string;
7   deleteTodo: Function;
8 }> = (props) => {
9   return (
10     <tr>
11       <th scope="row">{props.rowNumber}</th>
12       <td>{props.rowDescription}</td>
13       <td>{props.rowAssigned}</td>
14       <td>
15         <button
16           className="btn btn-danger"
17           onClick={() => props.deleteTodo(props.rowNumber)}
18         >
19           Delete
20         </button>
21       </td>
22     </tr>
23   );
24 };
25
```

Just do the same thing in NewTodoForm.tsx

```
react-todos > src > components > NewTodoForm.tsx > NewTodoForm
1  import React, { useState } from "react";
2
3  export const NewTodoForm: React.FC<{ addTodo: Function }> = (props) => {
4    const [description, setDescription] = useState("");
5    const [assigned, setAssigned] = useState("");
6
7    const submitTodo = () => {
8      if (description !== "" && assigned !== "") {
9        props.addTodo(description, assigned);
10       setAssigned("");
11       setDescription("");
12     }
13   };
14
15   return (
```

And do same thing in TodoTable.tsx

```
react-todos > src > components > TodoTable.tsx > TodoTable
1  import React from "react";
2  import { TodoRowItem } from "../TodoRowItem";
3
4  export const TodoTable: React.FC<{
5    todos: TodoModel[];
6    deleteTodo: Function;
7  }> = (props) => {
8    return (
9      <table className="table table-hover">
10        <thead>
11          <tr>
12            <th scope="col">#</th>
13            <th scope="col">Description</th>
14            <th scope="col">Assigned</th>
15            <th scope="col">Actions</th>
16          </tr>
17        </thead>
```


Then Fix all the issues, and run the application.

localhost:3000

Your Todo's			
#	Description	Assigned	Actions
1	Feed puppy	User One	Delete
2	Water plants	User Two	Delete
3	Make dinner	User One	Delete
4	Charge phone battery	User One	Delete
New Todo			