Assignment Nine

Please read:

- This is your ninth assignment which you need to submit on the LMS. This is an assignment on Typescript.
- Work on the questions given below and be ready with your solution in CodeSandbox. You have to submit your CodeSandbox link below on this page.
- Late Submission: The deadline to submit this assignment is 19th February 2024, 6:00PM IST.

Important Instructions:

1. Make sure that you follow the typescript rules as it will help you build your typescript muscle.
2. Do not copy from someone else as that would be cheating.

challenge

https://codesandbox.io/s/mailbox-psq6so

You can watch the video for reference:

ex01: add typescript to the project

If you are using codesandbox, add these packages in "add dependencies" or run the following command.

```
yarn add --dev typescript @types/react @types/react-dom
```

COPY

ex02: create a tsconfig.json file:

In your project's root directory, create a `tsconfig.json` file. This file will configure typescript for your project. You can create a basic configuration to start with:

```
{
  "compilerOptions": {
    "target": "esnext",
    "module": "esnext",
    "strict": true,
    "jsx": "react-jsx",
    "esModuleInterop": true,
    "moduleResolution": "node"
  }
}
```

COPY

ex03: rename .js and .jsx files to .ts and .tsx:

challenge

Start with the `MailCard` component. Your task is to add TypeScript types to the `MailCard` component. This component displays an individual mail card and provides various actions that can be taken on the mail.

1. Define a type or interface named `MailCardProps` that represents the props for the `MailCard` component. Include the following prop types:
   - `mId`: string
   - `subject`: string
   - `content`: string
   - `isStarred`: boolean
   - `unread`: boolean
   - `isInSpam?`: boolean (optional)
   - `isInTrash?`: boolean (optional)
   - `noDetail?`: boolean (optional)
2. Use the `MailCardProps` type/interface to annotate the props of the `MailCard` component.

ex03.1: rename pages in the pages folder - home

challenge

Your task is to add TypeScript types to the `Home` page component, which displays a list of mail cards based on the user's preferences and filters.

1. Add type annotations to the `Home` component.
2. Replace any instances of the `Mail` type with the appropriate type. Utilize the `MailCardProps` here. You can rename the `Mail` can create a common type.
3. Ensure that all props, states, and functions are properly typed.

ex03.2: rename pages in the pages folder - maildetail

challenge

The task is to rename the `maildetail.js` file to `maildetail.tsx` in the pages folder and add TypeScript types to the `MailDetail` component. This component displays the details of a specific mail based on its ID.

1. Rename the file from `maildetail.js` to `maildetail.tsx`.
2. Add TypeScript type annotations to the `MailDetail` component's props and other variables. Utilize the `Mail` type here.
3. Ensure that the component uses the types correctly throughout the code.

ex03.3: rename pages in the pages folder - spam

challenge

Your task is to add TypeScript types to the `Spam` component. This component displays a list of spam mails and their details.

1. Add TypeScript type annotations to the `Spam` component's props and other variables. Utilize the `Mail` type here.
2. Use the types correctly throughout the code to ensure proper type checking.

ex03.4: rename pages in the pages folder - trash

challenge

Your task is to rename the `trash.js` file to `trash.tsx` in the pages folder and add typescript types to the `Trash` component. This component displays a list of mails that are in the trash.

1. Add TypeScript type annotations to the `Trash` component's props and other variables. Utilize the `Mail` type here.

2. Use the types correctly throughout the code to ensure proper type checking.

ex04: convert index.js to index.tsx

challenge

Your task is to convert the `index.js` file to `index.tsx` in the root of your project. The `index.tsx` file is the entry point of your application and initializes the rendering of your app.

📌 You will encounter an error related to the `rootElement` when using the `createRoot` function. Consider how you're using `getElementById("root")` and whether you're ensuring it's not `null` before using it with `createRoot`.

ex05: convert app.js to app.tsx

challenge

Your task is to modify the `App` component to use TypeScript for type annotations. This component represents the main structure of your application and defines the navigation links for different sections.

1. Import necessary types and components from the `react-router-dom` library.
   ```
   **@types/react-router-dom**
   ```
   COPY
2. Add TypeScript type annotations to the `getActiveStyle` function and other variables.

3. Update the return type of the `App` component to `JSX.Element`.

ex06: add types to data

challenge

Your task is to add typescript types to the `mails` array in the provided code. This array contains a list of mail objects with different properties such as `mId`, `unread`, `isStarred`, `subject`, and `content`.

ex07: add types to reducer

challenge

Your task is to add typescript types to the reducer logic for handling mail actions. This includes defining the `Mail` type and the `MailAction` union type, as well as using these types in the `mailReducer` function.

1. Define the `Mail` interface based on the properties of a mail object.
2. Define the `MailAction` union type that includes all the possible action types and their payloads.
3. In the `mailReducer` function, add TypeScript type annotations to the function parameters and return type to ensure correct typing.

ex08: add types to context

challenge

Your task is to add TypeScript types to the context setup, including the `MailProvider` component and the `useMail` hook. This involves defining proper types for the context value and ensuring type safety throughout the context.

1. Define the `MailProviderProps` interface with a `children` property of type `ReactNode`.
2. Define the `MailContextValue` interface based on the `MailState` type, adding the `dispatch` property of type `React.Dispatch<MailAction>`.
3. Create the `MailContext` using `createContext` with an initial value of `null` and the type parameter of `MailContextValue | null`.
4. Inside the `MailProvider` component, add type annotations to the function parameters and return type.
5. Inside the `MailProvider` component, add proper TypeScript types to the `contextValue`.
6. Define the `useMail` hook with the correct return type of `MailContextValue`