# Housekeeping:

Sample project available at: https://github.com/ParasDalal/reacttraining.git

See all the commits

git log --oneline

git checkout <GitId>

Will not cover any advanced things

Project created with TypeScript

npm install -g npm

npx create-react-app sample --template typescript

npm install react-router-dom @types/react-router-dom

# General rules of the Hooks

Hooks are just functions

Pre-defined hooks are for very general use cases

There are custom hook libraries available for various complex situations

We can write our own hooks very easily

<https://reactjs.org/docs/hooks-rules.html>

# useState

maintain the state information

Local variables don’t persist between renders

React don’t trigger automatic rendering when data changes for local variable

# useEffect

DOM rendering and after effect

Can be used for fetching data

Will need the cleanup function as "return" in case it is used for some subscription or event hookup. E.g. if you subscribe to window.setTimeout, then you will need to do window.clearTimeout

The second parameter that has the dependency is very important. If that parameter is not supplied, it will be called only once.

This is client side only (so, will not work if the page is processed on server)

# useRef

Assign the reference from any DOM element

It allows direct access to an element in DOM

# useContext

Create interface to explain the type of data that will be available

Can create an interface with actual data handling methods

Data can be handled as State

Wrap the component tree node in <Provider> under which the data will be available

Data should go from top component to all children

Changes should propagate up to the main provider and that will refresh all the dependent places

If we change data using "Form" submit, then may need to do preventdefault()

# useReducer

If you need multiple type of data based on action, you need to define “Action type”

If the state has multiple pieces of data and the changes to them depend on the kind of “action”, useReducer is a way to handle it.

The “reducer” function will take the state (data), will apply action on it and then will return the revised state (data)

Since the reducer is maintaining the state, we don’t need to use useState

Since after every “dispatch” of the action, the component is rendered again, we don’t need the “useEffect”

If you have an array of objects as the “data” that you are changing in the reducer, you may need to return the clone of the array for React to trigger re-render

# Useful things/URLs:

## Typescript types:

interface Props{

children: React.ReactNode;

}

## To add Typescript to any existing project

npm add typescript @types/react @types/react-dom @types/node

## URLs

<https://usehooks-ts.com/> (Many ready-made hooks using react/typescript)

<https://beta.reactjs.org/> (New documentation for React)

<https://www.w3schools.com/howto/howto_js_topnav.asp> (Navigation bar)

React training repo (7 different courses and 2 different applications):

<https://github.com/ReactTraining/react-workshop.git>

<https://dmitripavlutin.com/react-usereducer/> (Explanation of useReducer Hook)

<https://www.codegrepper.com/code-examples/javascript/usereducer+not+rerendering+react> (Few answers on this site are to the point)

<https://www.telerik.com/kendo-react-ui/components/grid/editing/editing-inline/> (GRID samples from Kendo UI)

<https://quicktype.io/> (Convert JSON into type safe code)

## Other options to create a new React application

### create-next-app

### create-snowpack-app

### Nx (nx.dev)

### Gatsby (gatsbyjs.com)

### TSDX (tsdx.io) - may be worth exploring

## To add any local project to GitHub

Download GitHub command like (cli.github.com)

gh auth login (To authorize to GitHub)

Git (to put it under source control – local project)

git init -b main

Go to GitHub and create a new repository (Do not add the default license/readme etc.)

- gh auth login (To authorize your local machine to GitHub)

- git remote add origin https://github.com/ParasDalal/reacttraining.git

- git remote -v (to check the remote connection)

- git add . (Add local files)

- git commit -m "Initial"

- git push -u origin main

## Check out:

react-error-boundary (package for controlled display of errors)

## Update any project to the latest version of packages

npm install -g npm-check-updates

ncu -u (This will update the versions in the json file)

npm install

## If npm install does not work for you on work computer

npm config set strict-ssl false

## Explicitly defining the return types in TypeScript

We can define the return types as “const” to return only the typed values  
e.g. return [loading, setAuthInfo] as const

## PluralSight courses

Using React Hooks by Peter Kellner