# Supplementary notes REACT-JUN25

## Create a react app in Vite

- [ ] npx create-react-app@latest my-app
- [ ] https://create-react-app.dev/docs/getting-started

*Using **npm create-react-app** relies on create-react-app being globally installed, potentially leading to version conflicts.*

*By contrast, **npx create-react-app** mitigates version conflicts by fetching and executing create-react-app locally, ensuring project integrity and stability.*

## VSC extensions used in course

- [ ] Live server
- [ ] Auto tag rename
- [ ] (Optionally) Prettier code formatter

## Node deps used in project

- [ ] Vite
- [ ] React and react-dom
- [ ] Axios
- [ ] Uuid
- [ ] React-browser-router
- [ ] TailwindCSS

## Browser tools for Chrome

- [ ] React dev tools
- [ ] Redux dev tools

## VS Code shortcuts - JS environment

| | |
|---|---|
| CTRL + B | toggle explorer pane / full screen |
| CTRL + +/- | up/down font size |
| CTRL + / | toggle comment on/off |
| **ALT/OPTION + CLICK** | **multi-cursor select** |
| ALT + SHIFT + DOWN/UP | duplicate line down/up |

### *find and kill running servers:*

*After a long session working in React it is quite possible you may have dev servers running even after you have closed down panes in VS Code. Use these commands to search for likely port numbers (5173 or greater for Vite, 3000 for json-server, 5500 for live server etc) and kill them.*

*ON MAC OS:*
**sudo lsof -i :<PORT>**
(Replace <PORT> with the port number you want, but keep the colon)
*// this will print out stuff including PID*
**kill -9 <PID>** *// without angle brackets*

*ON WINDOWS:*
**netstat -ano | findstr :<PORT>**
(Replace <PORT> with the port number you want, but keep the colon)
**netstat -ano | findstr :***
(ALL ports)
**taskkill /PID <PID> /F**
(No colon this time)

*ON LINUX:*
*same:* **sudo lsof -i :4000**
*// this will print out stuff including PID*
**kill -9 <PID>**

## Links from class

*https://dummyjson.com/docs/posts#posts-add*
*"Adding a new post will not add it into the server.*
*It will simulate a POST request and will return the new created post with a new id"*

*Show/hide password toggle in React*
https://www.geeksforgeeks.org/how-to-show-and-hide-password-in-reactjs/

*Object-fit CSS property*
https://developer.mozilla.org/en-US/docs/Web/CSS/object-fit

- [ ] Thinking in React
- [ ] https://react.dev/learn/thinking-in-react

*This is a great old article by Dan Abramov himself and shows how to approach a react project from the UI-first.*

*Dave Ceddia Pure React (paywall: worth it)*
*https://www.purereact.com/*

## Misc notes

- [ ] Higher order funcs and first class functions: diff https://www.geeksforgeeks.org/difference-between-first-class-and-higher-order-functions-in-javascript/
- [ ] useEffect without a dependency array
- [ ] https://react.dev/reference/react/useEffect#examples-dependencies
    - [ ] Other articles/posts
    - [ ] https://dev.to/csituma/why-we-use-empty-array-with-useeffect-iok - anti pattern
    - [ ] https://stackoverflow.com/questions/57760842/why-would-we-use-useeffect-without-a-dependency-array - anti pattern

- [ ] React children prop for passing nested JSX
- [ ] https://react.dev/learn/passing-props-to-a-component#passing-jsx-as-children

- [ ] Event loop video

*Jake Archibald talk on the event loop:*
*https://youtu.be/cCOL7MC4Pl0*
*02:58 - 12:50 (until requestAnimationFrame())*
*Jake on socials: @jaffathecake*

- [ ] *Also, Jake on image formats*
- [ ] https://jakearchibald.com/2020/avif-has-landed/demos/compare/?show=f1&img=/c/f1-good-37eadc74.avif

- [ ] Keys in list items
- [ ] https://react.dev/learn/rendering-lists#keeping-list-items-in-order-with-key

- [ ] Fake JSON API for get and post
    - [ ] ***https://dummyjson.com/custom-response***
- [ ] Each user must generate their own fake endpoints
- [ ] Plumb in Tailwind as dev dependency
- [x] TailwindCSS https://tailwindcss.com/docs/installation/using-vite
    - [x] npm install tailwindcss @tailwindcss/vite
    - [x] **/vite.config**

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import tailwindcss from "@tailwindcss/vite";

// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
});
```

        - [x] **/index.css**
        - [x] @import "tailwindcss";

- ○ RTK and the use case for redux
  - ○ https://redux.js.org/tutorials/fundamentals/part-2-concepts-data-flow
- ○ State Diagrams based on car app, in /state-diagrams/
  - ○ These diagrams show, in 3 stages, how we might continue to develop our car app in "little state trees". The simplest one shows what we achieved, with PartsForm talking to App via the "lifting state up" pattern in native React. App then passes the modified state down to PartsTable as props.
  - ○ The middle complexity diagram shows the architecture of adding a Redux store, acting as a centralised repository for all app state. Instead of modifying state held locally in a component such as App, actions are dispatched to the store which call reducers, which create a new immutable copy of store state. The store state at any one time is available to the whole app from a single centralised source. With just one little state tree, there is very little point in adding Redux.
  - ○ The most complex diagram shows another, entirely separate, part of the app that may be concerned with, say for arguments' sake, deliveries conducted after the parts leave the warehouse system. That exists in its own little state tree with App. PartsForm and PartsTable don't need to know about deliveries, but deliveries may require some information from the store about number of parts in stock etc. If two or more independent parts of an app need to share state, then that is a use case for Redux.