React Notes

TSX / JSX

→ allows you to return html tags with javascript in them

→ you can create variables as html:
const name = <h1> Name </h1>

Components

→ a javascript function that return some tsx/jsx

→ components can be called in other tsx files:
<Component/>

Props

→ every react component will take props;

→ you can pass any type of data in props;

→ it's basically a parameter for components

→

CSS in React

→ you give your html elements in your TSX's files
className='my-class' and if you import '../…/style.css'
and you access those className via style.css

→ similar to average html + css

→ you can pass it with style.module.css
import styles from "./ style.module.css"
<h1 className={styles.name} > instead of <h1 className = 'name'>

Conditional Rendering

return (
<div className='..'>
{var ≥ 18 ? <h1>…</h2> : <h1>…</h1> }
</div>
)

Lists

const names = ['Tudor', 'Rares', …]

names.forEach - parse through all of the names

names.filter

names.map((name, key(basically the index)) ⇒{
return <h2 key={key}> {name} </h2>
});

names.reduce
you can do this even if your list has Objects and access the objects fields
{obj.field}

useState Hook

- it is used for telling react to re-render the page when smth happens to that var:

const [varName, setVarName] = useState(initialValueOfTheVar);

so whenever setVarName is called ⇔ varName is changed, react re-renders

- HOW TO CHANGE CSS w useState:
    - <div style={{color: textColor}}>
      const [textColor, setTextColor] = useState(”black”);
      onClick = { () ⇒ {
      setTextColor = “red”}} or have a handleOnClick for it

Component lifecycle

- mounting - start appearing
- updating - changing
- unmounting - stopped appearing

useEffect Hook

- triggers for each lifecycle step
- useEffet(() ⇒ {
        //useEffect is called everytime the component state changes
        console.log(”Component mounted~!”);

        return () ⇒ {
            console.log(”This is called only when unmounted”);
  }
  }, [*here you can add the variable that changes or som shit*])

how to fetch data from an api

- you make a request, get the data and then display it to your website or whatever
- fetch(”api.url”) - uses to fetch data from API:
    - you grab the url from the api
    - fetch() → json
    - fetch().then((response) ⇒ response.json())
      .then((data) ⇒ {

do smth with the data
})

Axios library

- library to fetch data
- import Axios from “axios”
- Axios.get(”api.url”).then((response) ⇒ {

  response.data → manipulate it
  })
- BETTER WAY