... render (<App />, ...

(10) » In big websites then first direct all components to App.jsx file. & only import App.jsx in index.js.

---

(11) » Components Practice → A simple task given. [HTML inside return, JS, logic outside] (9) ~~**~~ ↳ if only JS no return necessary

(13) Javascript ES6 (Import, Export & Modules).

» ***

⇒ [Think of import as the code actually being present in the file you are working in] (11)

» we have been using default export till now, but if you want to export more than one thing in one file then export { names of fn to export seperated by comma };

» Simi in import ⇒ import PI, { doublePI, triplePI } from "./math.js";

(OR) . (12)

a { import * from "./math.js"; ↳ imports everything the file has.

(as PI) ↳ add this to use it in file (necessary)

then you have to write PI.doublePI, can't directly use doublePI.

↳ discouraged above method recommended.

(9) Practice task given.

---

(9) Setting up of Environment done.

i.e for Atom. Now we work locally.

(12) Calling.

```
function App() {
  return ( <div>
    <Heading />
  </div>
  );
}
```

If no function

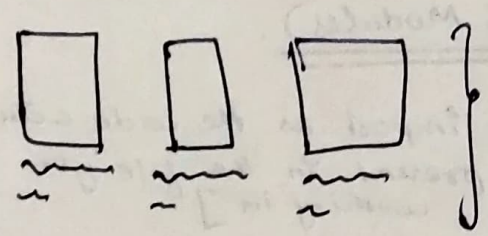
```
ReactDOM.render(<App />, document.getElementById("root"));
```

(16) Keeper App Project (Part 1).

Succesfully completed Part 1 of keeper app project.

---

(17) React Properties (Props).



can write these codes
vs
can use one code 3 times.
by just importing & using.

* But we want diff info in each card though the view remains the same.

We do this: `<Card1/>` → here we can pass things.

Eg:- code

```
function Card(props) {
  return (
    <div>                          className can only go here
      <h2>{props.name}</h2>
    </div>                         can name <Avatar/> here and can have another function for it.
  );                               (So function inside function)
}

ReactDOM.render( <div>             Not here
  <Card name="Abhinav"/>           Used when Avatar used for other things too other than the cards
  </div>,
  document.getElementById('root'));
```

---

(18) React props practice task given.

(18) React DevTools.

- React DevTools is present beside Problems & console in codesandbox.
- It gives you the component tree, so if you have component inside components & so on. It will show you how that looks like.
- Chrome web store → React Developer Tools by FB.
  [View → Developer → Developer tools]
  ↳ components.
- Really helps in debugging.

---

(19) Mapping Data to Components.

```
<Card
  name = "..."
  img =
  tel =
  email =
/>
```

Very tedious process to be followed for every card.

As we have to type in everything manually, we have just automated 90% style till now.

- For this, we can use loop to make life easier.
- map fns is used to handle arrays (containing data).

- Let contacts be the array

```
<Card
  name = {contacts[0].name}
  img = {contacts[0].imgURL}
  tel = ""
  email = ""
/>
```

→ 3 times [1] & [2]

VS

- Let contacts be the array.

```
{contacts.map(createCard)}

function createCard(contact) {
  return <Card key
    name = {contact.name}
    ,etc. --
  />
}
```

(inside return; should be above.)

- key should be there to diff comp as it is loop, to identify diff.
- Unusable prop.

(20) Mapping data Challenge given.

---

(21) JavaScript ES6 Map/Filter/Reduce.

→ We will learn more about map, filter, reduce, find & findIndex.

→ **Map**

→ arrayname.map(fnname) → can be equated to const newNum = to use further.

array/data we use.

fnname → describes what to do with array.
→ can also use anonymous instead of fnname.

→ loops through each array element & does what is in the func.

→ **Filter**

→ Filters out elements we don't want from the array. (false)

→ arrayname.filter(fnname)

```
function New(x){
return x>10;}
```

→ **Reduce**.

→ Accumulating a value by doing something to each item in the array.

→ ∴ loops a large array, does a operation on them, & reduces them to a small value.

→ **find & findIndex** have same syntax & stops when something returns true & returns that value.

& Subscript also used → like slicing.

∴ Used for finding something.

(22) JavaScript ES6 Arrow function.

* Also called fat Arrow fns, they are a shorter way of writing Javascript functions.

* Eg:-
```
const newNum = numbers.map( (x) => {
    return x*x;
});
```

(23) Keeper App Part 2

(29) React Conditional Rendering

* Showing diff things based on whether the user is logged in (or) not.

* Note → whether you write code in App fn or not, everything must go through it bcz that's what we are exporting. If you build everything in different func, then you have to call that fn in App. [Else we have to export that too other than default].

* Single responsibility principle discussed.

* Have any doubt then you can go to the end product of the lecture (-417). Building User login.

* Ternary operator → condition ? Do something if true : Else do this.

  ★ → We cannot use logic in return statement as we discussed before.

  → But we can use ternary operator.

* If you don't want it to nothing on else then write : null.

  ★ [Even && used in some situations for this]
  AND operator