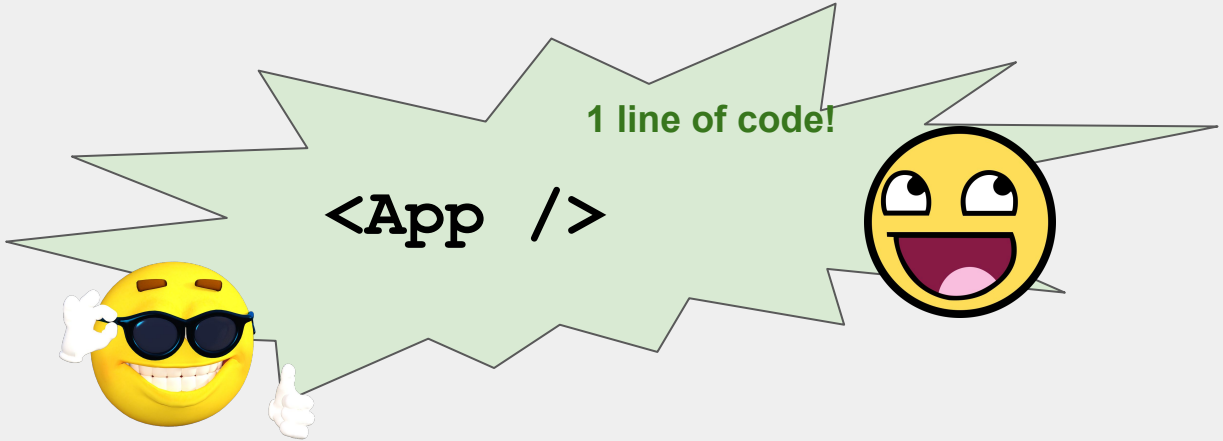
Workshop 2: Catbook in React

Akshaj Kadaveru

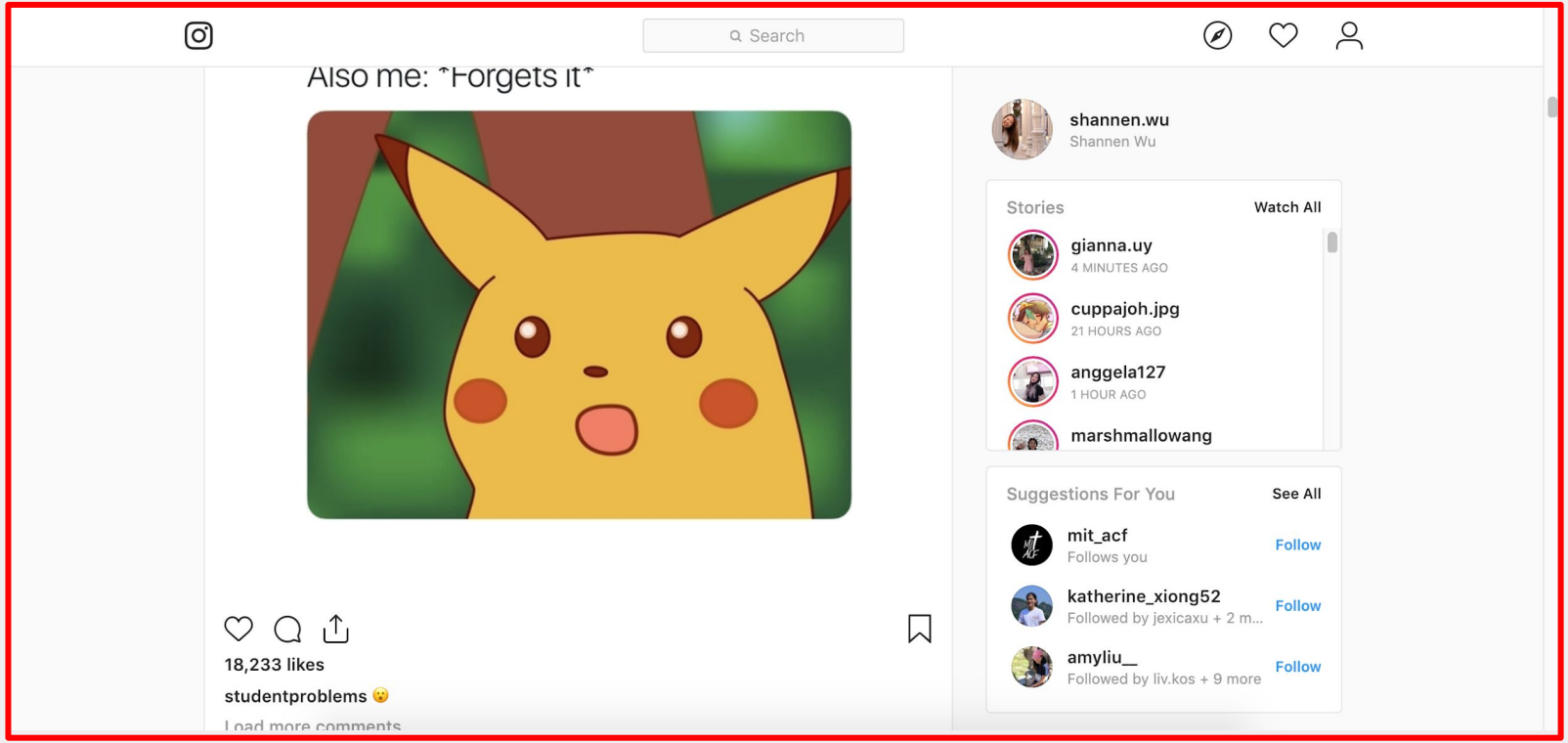
weblab.to/questions
weblab.to/q

How to write any website

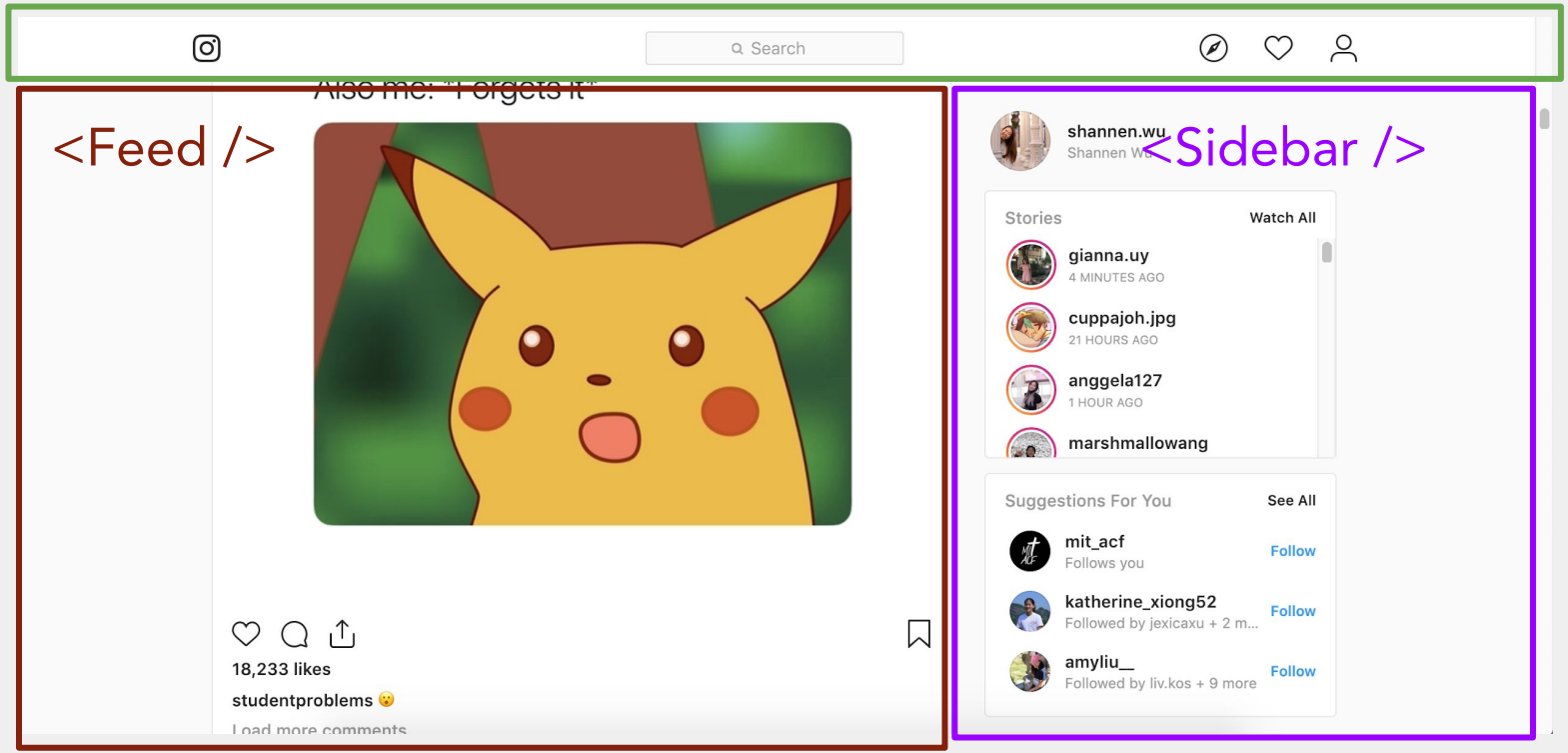


React Apps are "components of components"

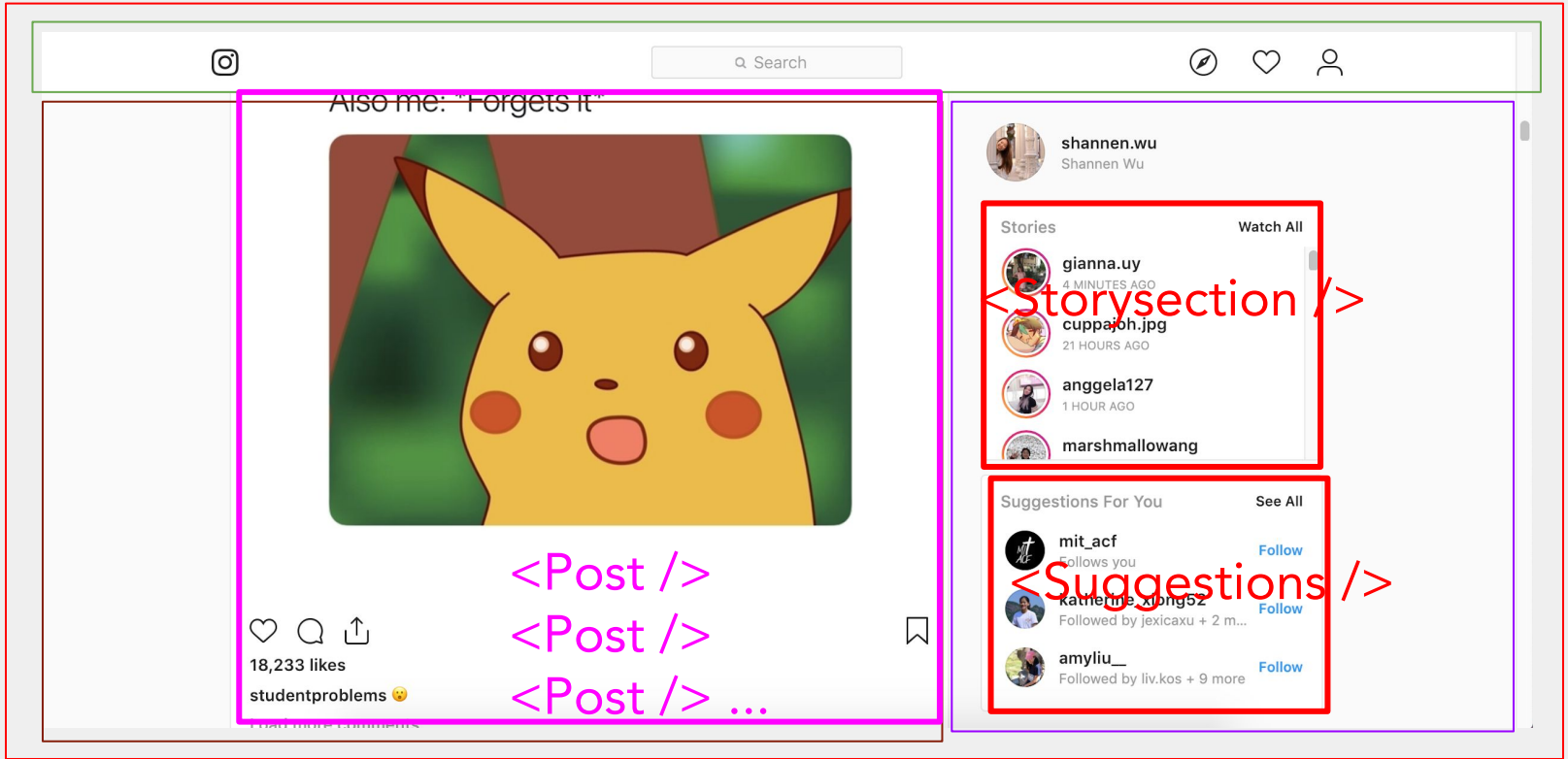
The App component (<App />)



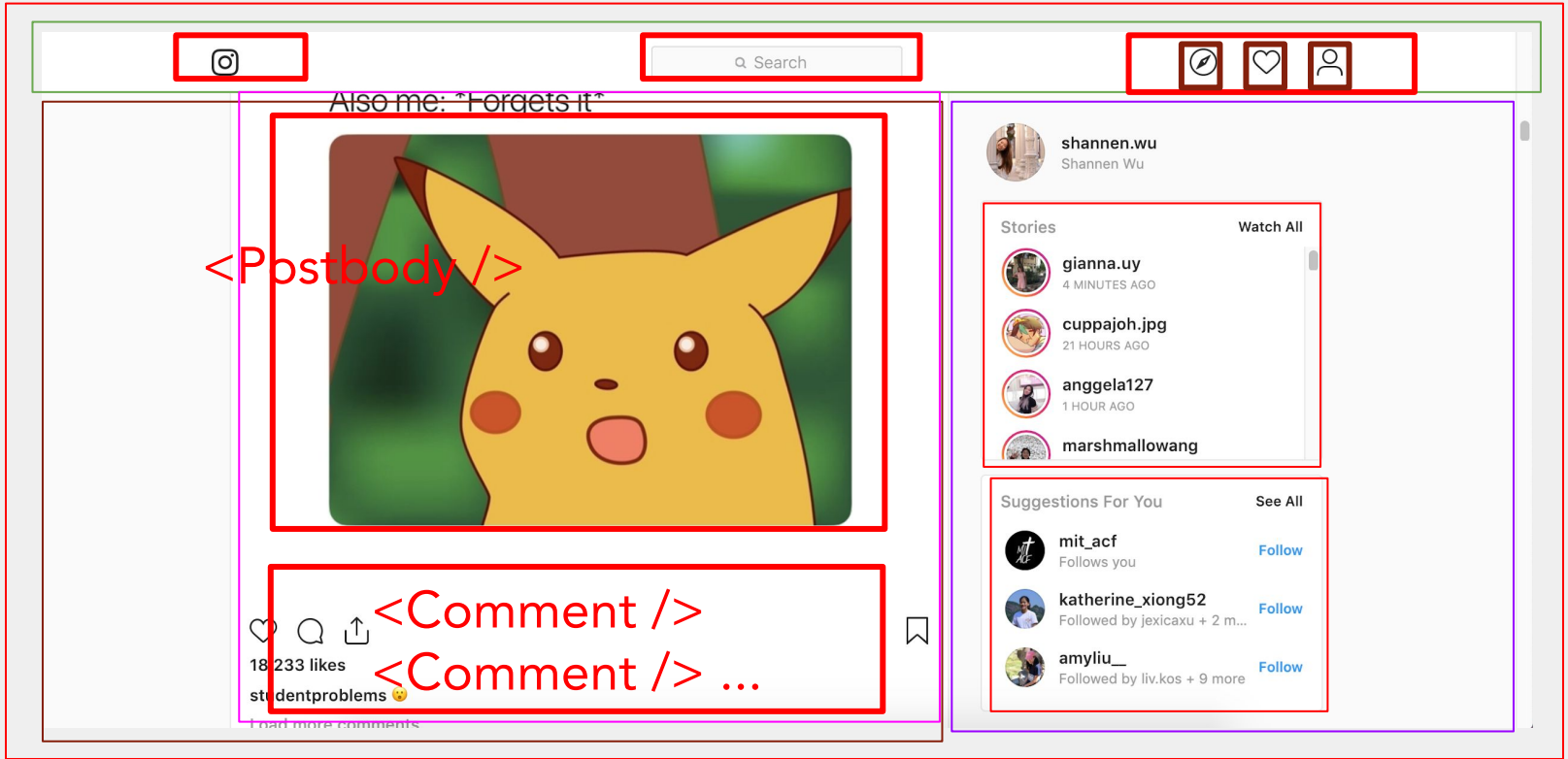
React Apps are "components of components"



React Apps are "components of components"



React Apps are "components of components"



Props ('Inputs')

Inputs passed from a parent to a child component

These are all props! (the inputs)

```
<Post name="Akshaj" text="Welcome to web lab!" />
```

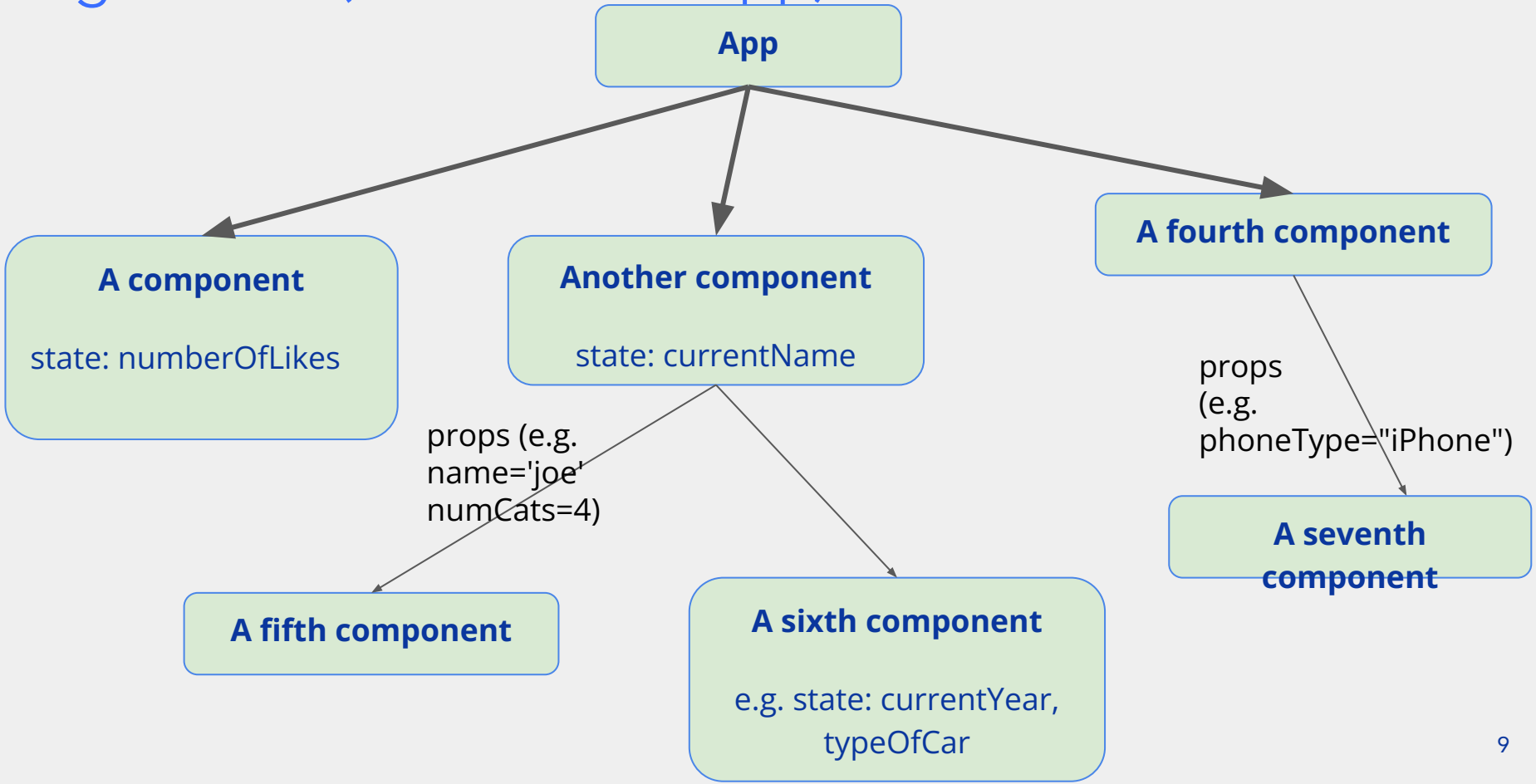
here, props = {name: "Akshaj", text: "Welcome to web lab!"}

State

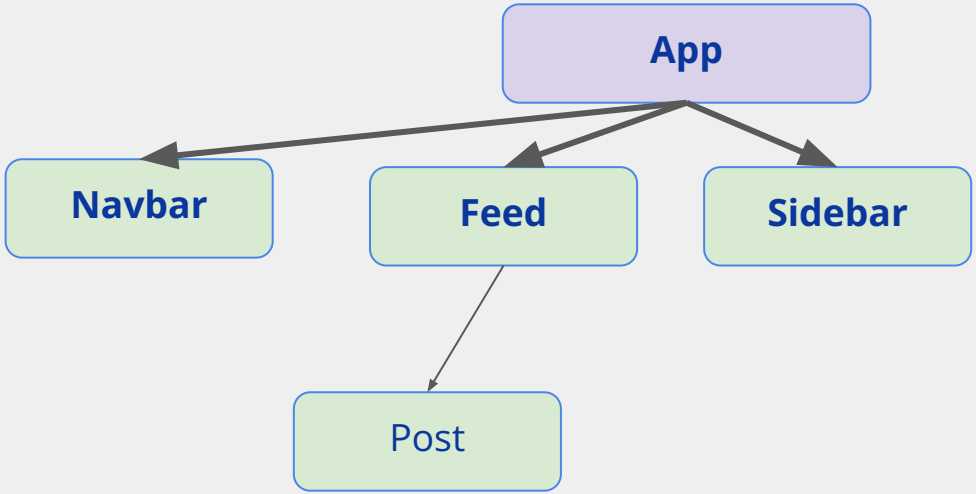
Updatable pieces of information maintained by a component.

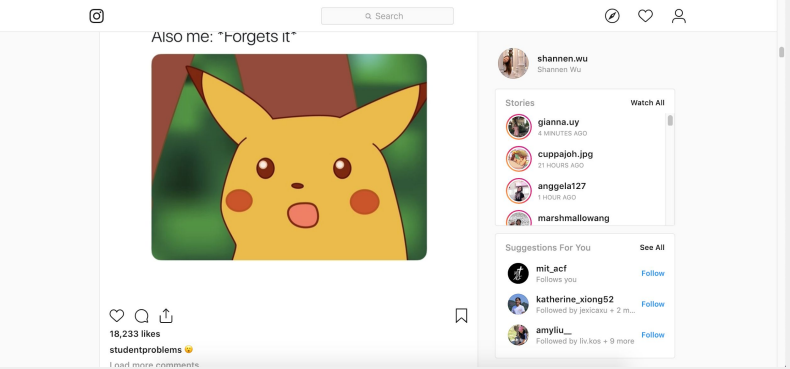
```
const [status, setStatus] = useState("busy");
const [isOnline, setIsOnline] = useState(false);
```

Big Picture (of a random app)



Big Picture

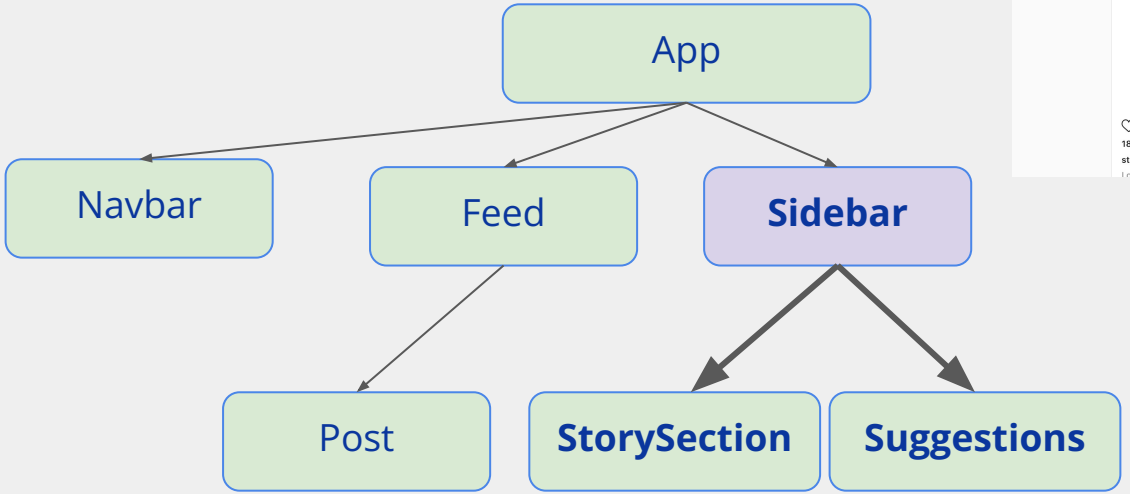


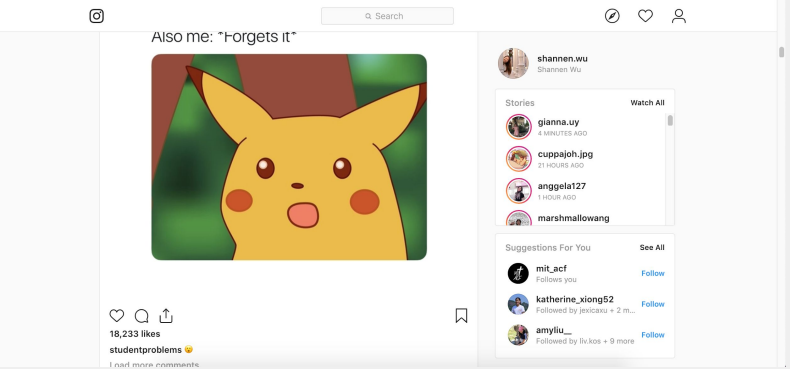


App

```
// App.js

const App = () => {
  return (
    <div>
        <Navbar />
        <div>
            <Feed />
            <Sidebar />
        </div>
    </div>

  )
}
```

Big Picture

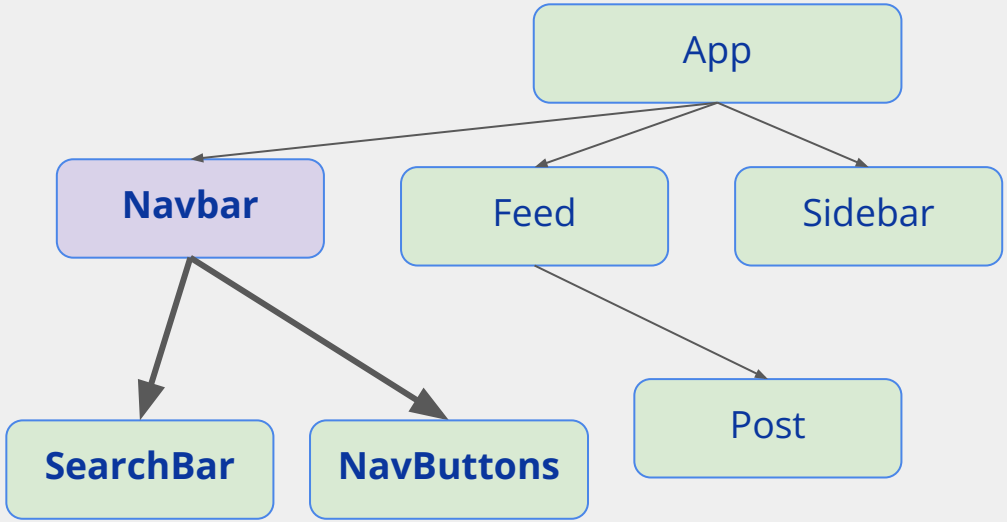


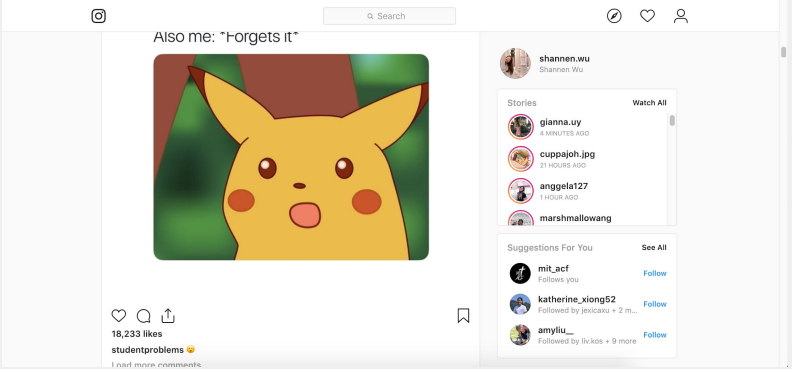


Sidebar

```
// Sidebar.js

const Sidebar = () => {
  return (
    <div>
        <StorySection />
        <Suggestions />
    </div>

  )
}
```

Big Picture

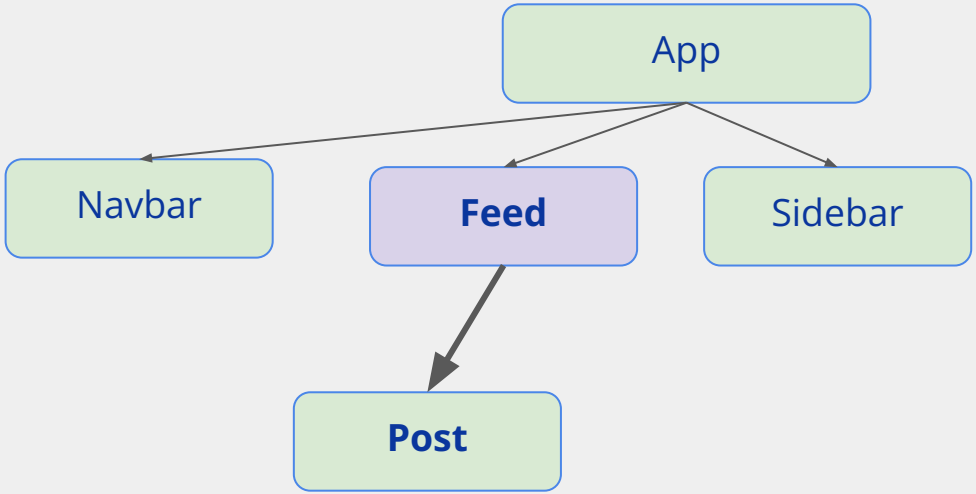


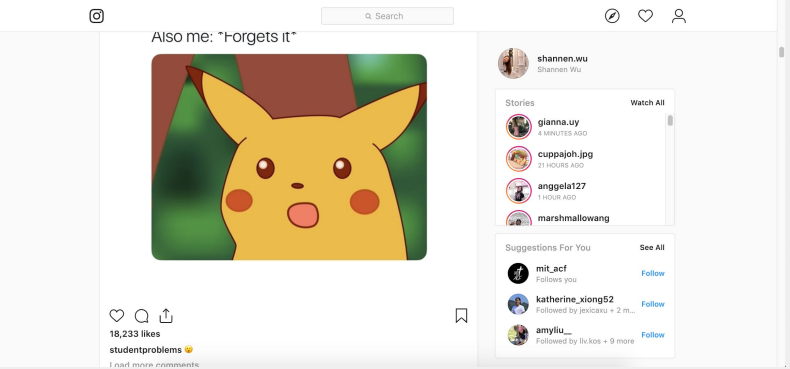


Navbar

```
// Navbar.js

const Navbar = () => {
  return (
    <div>
        <img src="instaIcon.png" />
        <SearchBar />
        <NavButtons />
    </div>

  )
}
```

Big Picture

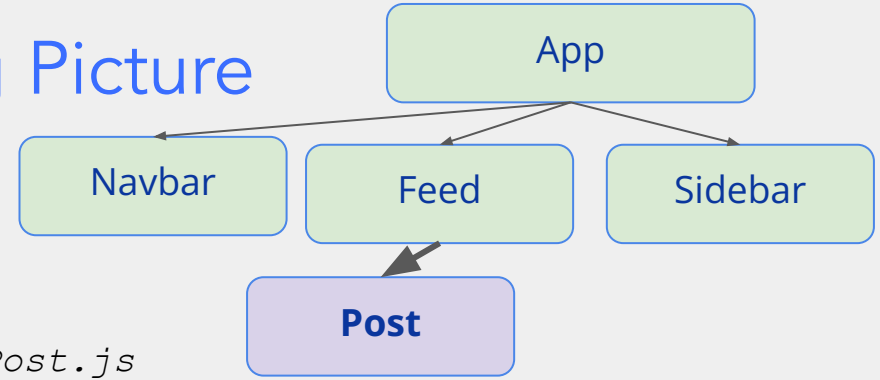


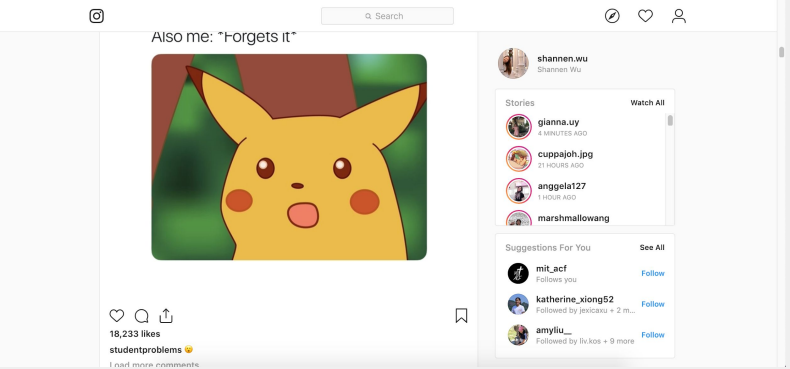


Feed

```
// Feed.js

const Feed = () => {
  return (
    <div>
        <Post />
        <Post />
        <Post />
        <Post />
    </div>

  )
}
```

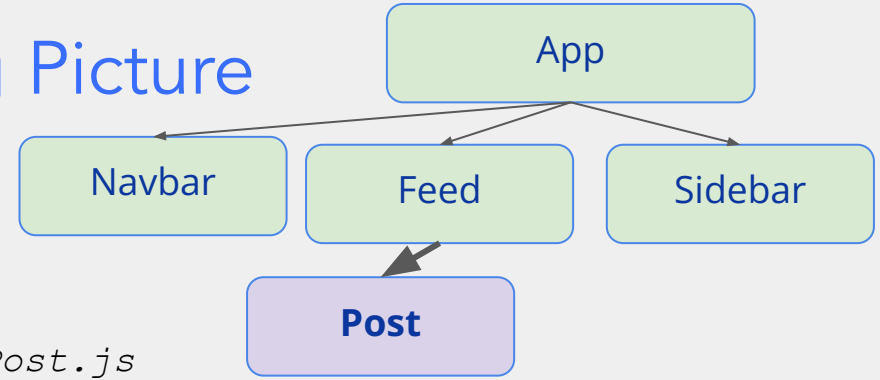
Big Picture

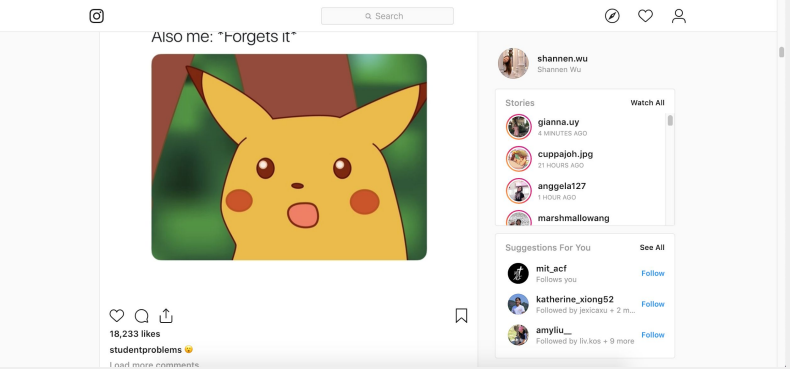




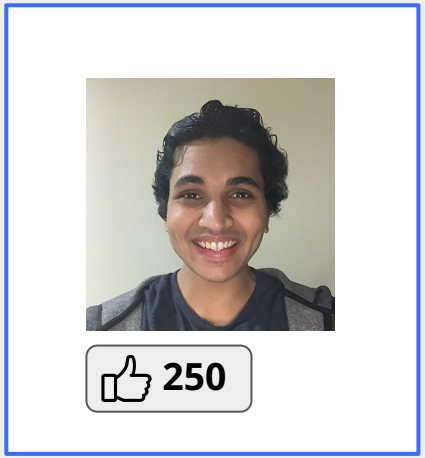

```
// Post.js

const Post = (props) => {
  // declare some state variables here
  // do whatever javascript stuff/calculations you want to do here
  return (
    *a bunch of HTML code*
  )
}
```

Big Picture

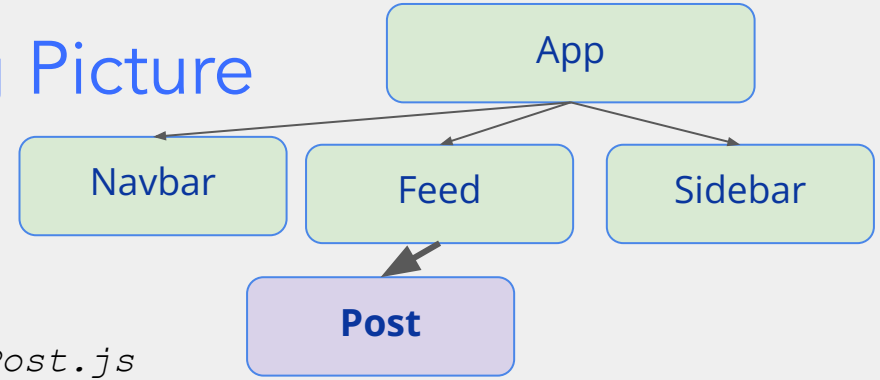


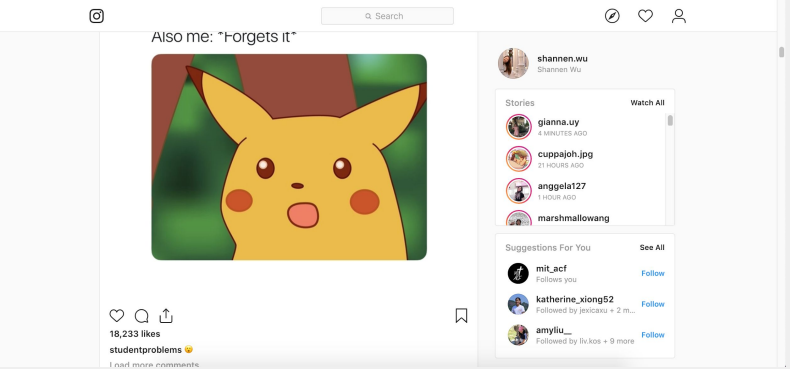



```
// Post.js

const Post = (props) => {
  return (
    <div>
        <img src={props.content} />
        <button>
            <img src="like.png" />
            {250}
        </button>
    </div>
  )
}
```

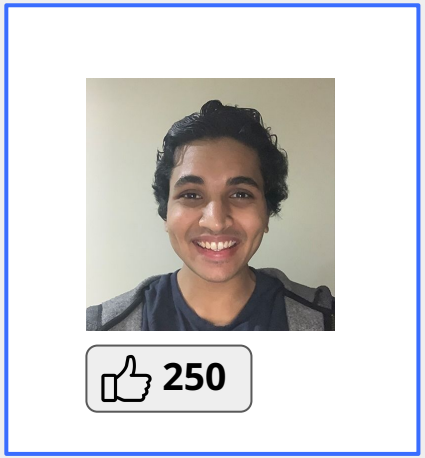


Big Picture

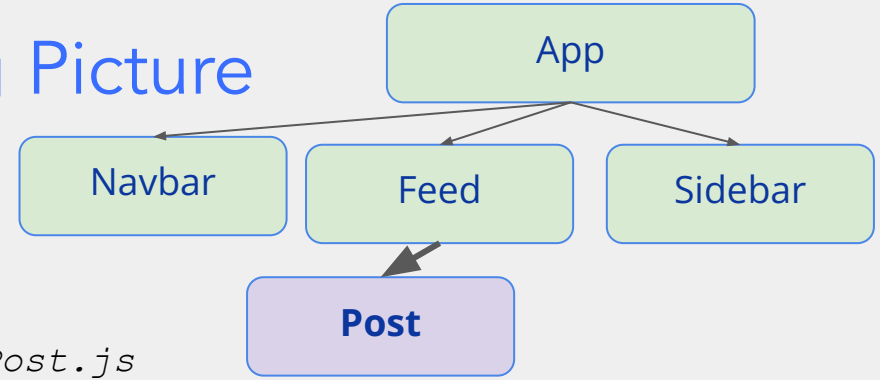


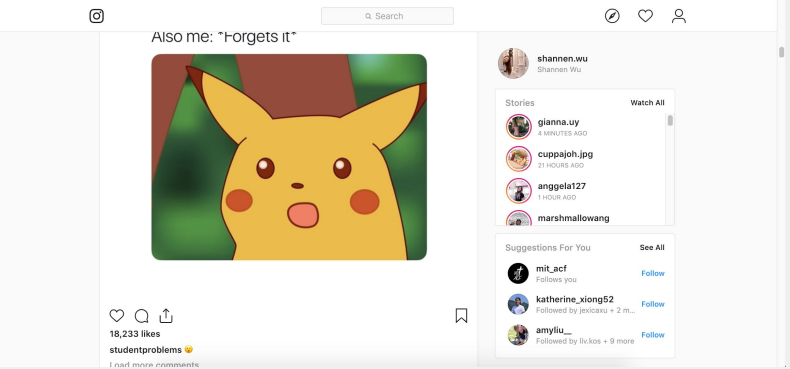



```
// Post.js

const Post = (props) => {
  const [numberOfLikes, setNumberOfLikes] = useState(0);
  return (
    <div>
        <img src={props.content} />
        <button>
            <img src="like.png" />
            {numberOfLikes}
        </button>
    </div>
  )
}
```

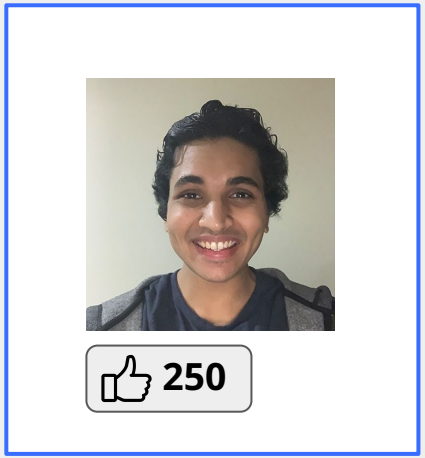


Big Picture





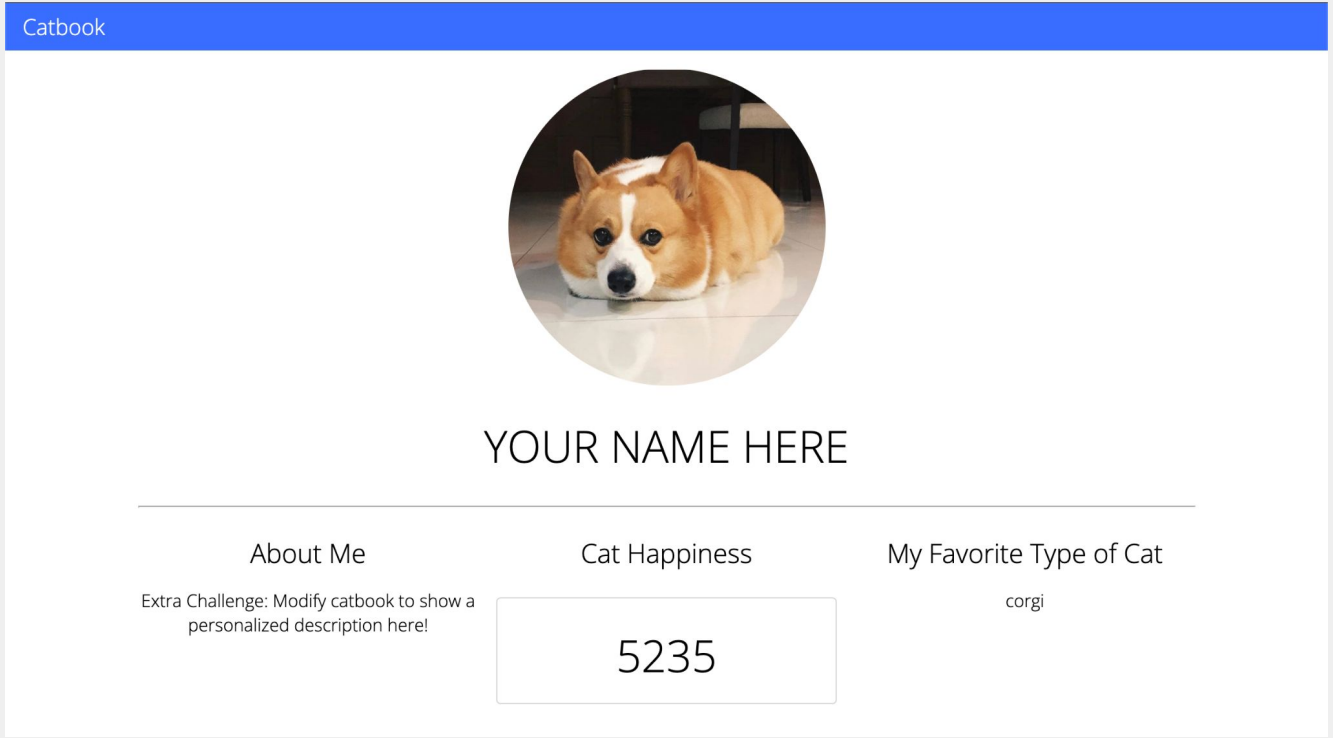

```
// Post.js

const Post = (props) => {
  const [numberOfLikes, setNumberOfLikes] = useState(0);
  const addLike = () => { setNumberOfLikes(numberOfLikes + 1); }
  return (
    <div>
        <img src={props.content} />
        <button onClick={addLike}>
            <img src="like.png" />
            {numberOfLikes}
        </button>
    </div>
  )
}
```



Recap

- A **React component** lets you break down a chunk of your UI into a reusable and independent piece of code.
- A component can be represented as a piece of HTML code, other React components, or both.
- It can receive and maintain its own information
- React uses a **component tree structure** to pass information
- Each component can take in **props** (inputs), and manages its owned contained **state** (private information)

Check out our recap guide at
weblab.to/react-guide-1



weblab.to/profile

Get Started — catbook-react

EXPLORER

Get Started

CATBOOK-REACT

client

.babelrc

.gitignore

.prettierrc

package-lock.json

package.json

webpack.config.js

Start

Open...

Clone Git Repository...

Open a Walkthrough...

Recent

PROBLEMS TERMINAL

bash

Akshajs-MBP:catbook-react akshajkadaveru$

Let's make sure everyone has the right version of node

Type

node -v

you should get '16.3.1' or '16.x.y'

OUTLINE

TIMELINE

w2-complete* 0 0 Prettier

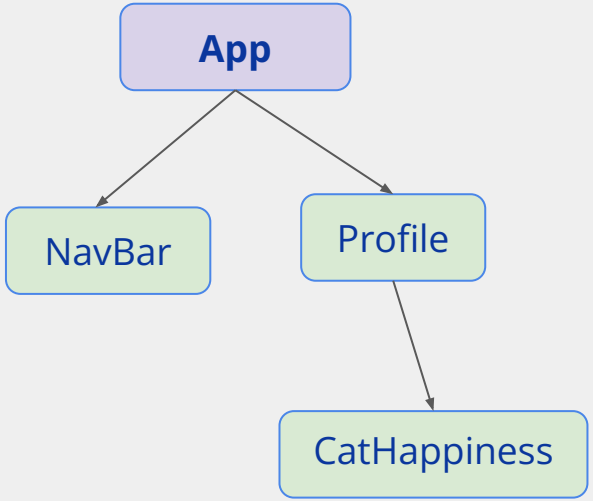
1. Open the catbook-react folder in VS Code

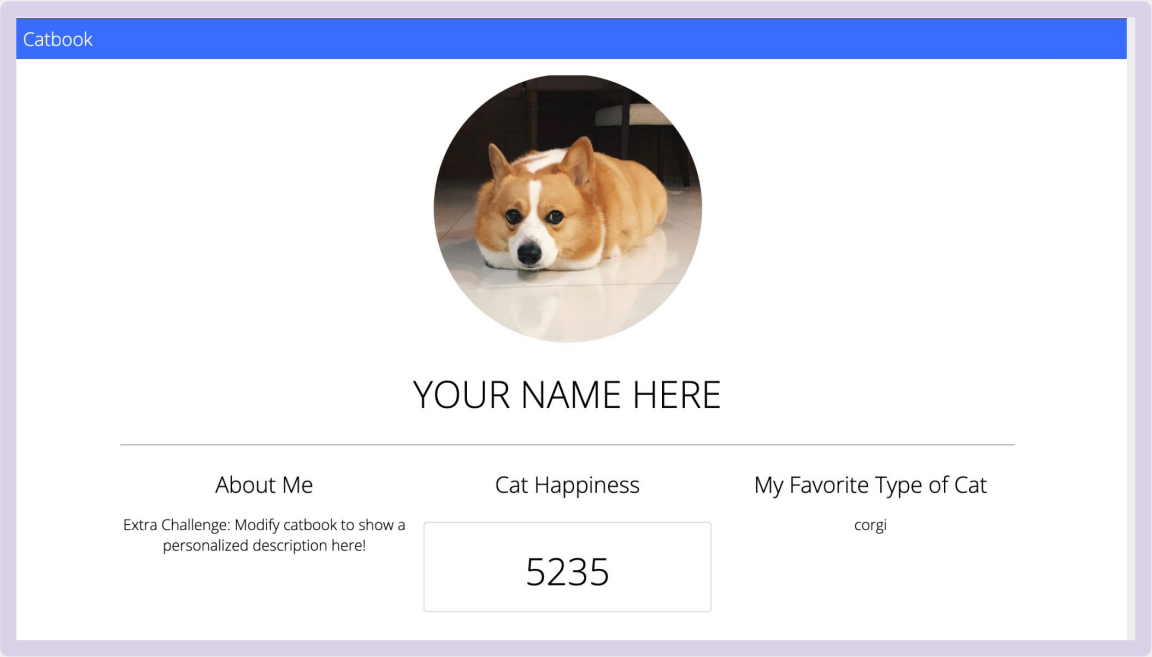
2. Open the terminal window in VS Code, make sure you are in the 'catbook-react' folder

3. Run the following commands:

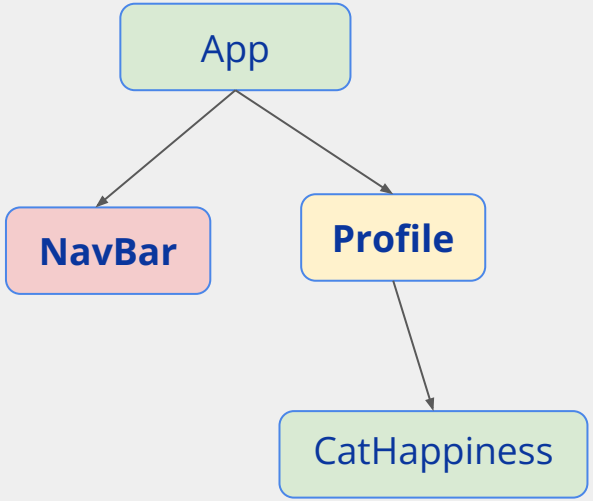
```
git fetch
git reset --hard
git checkout w2-starter
npm install
```

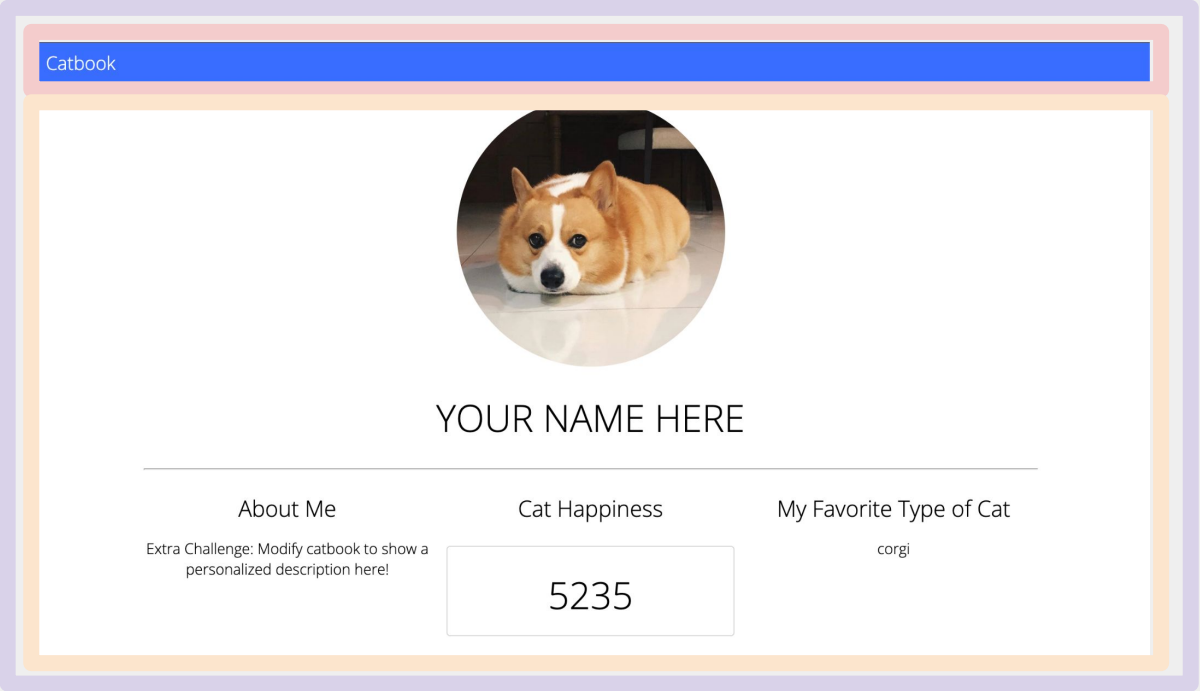
First: the component tree for Catbook!



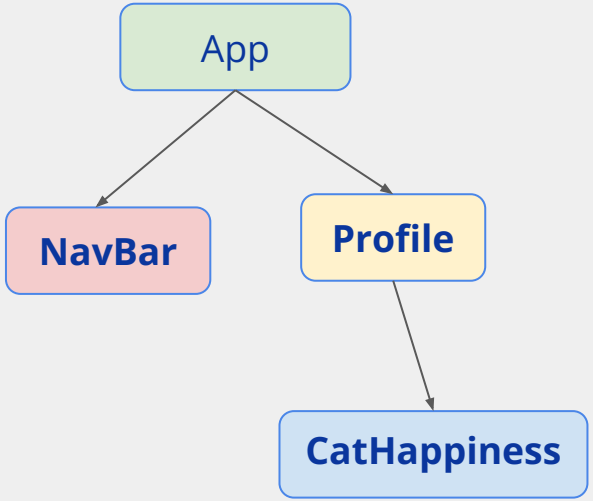


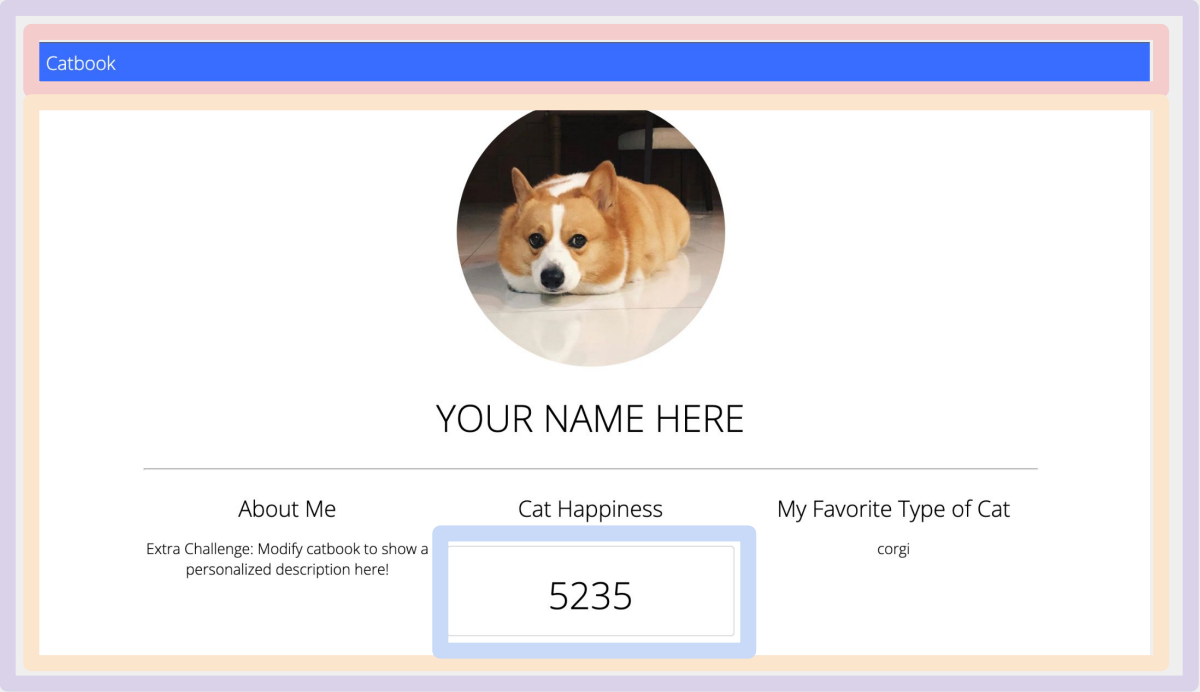
First: the component tree for Catbook!





First: the component tree for Catbook!





```
npm run hotloader
```

Navigate to **localhost:5000** and see the page update with your live changes!

Let's look at the starter code

- We use 'className' instead of class
- We put the React component name in front of our class names
  - why? so that we don't have the same class name in different components
  - CSS always applies to the entire webpage, so must include className to make it specific
    - for example if we set p {color: red} in one component it actually applies to all paragraphs on the whole webpage

Writing Components

Exercise 1: Implementing React Navbar

- You've implemented Navbar using Vanilla HTML- let's do it with React!
- Implement `return()` in **Navbar.js** with HTML code
- Implement **Navbar.css** .. go wild! Try to make the NavBar look like the catbook navbar, but feel free to add your own twist!

Catbook

<div **className**="style-name">

Exercise 1a: React Navbar CSS

```
.NavBar-container {
  padding: 8px 16px;
  background-color: #396dff;
}

.NavBar-title {
    color: white;
    font-size: 20px;
}
```

Exercise 1b: React Navbar JS

// NavBar.js

```
import React from "react";

import "./NavBar.css";

/**
 * The navigation bar at the top of all pages. Takes no props.
 */
const NavBar = () => {
  return (
    <nav className="NavBar-container">
      <div className="NavBar-title">Catbook</div>
    </nav>
  );
};

export default NavBar;
```

Let's get on the same page

Save or close out of all of your 'unsaved' files:


```
git reset --hard
git checkout w2-step1
```

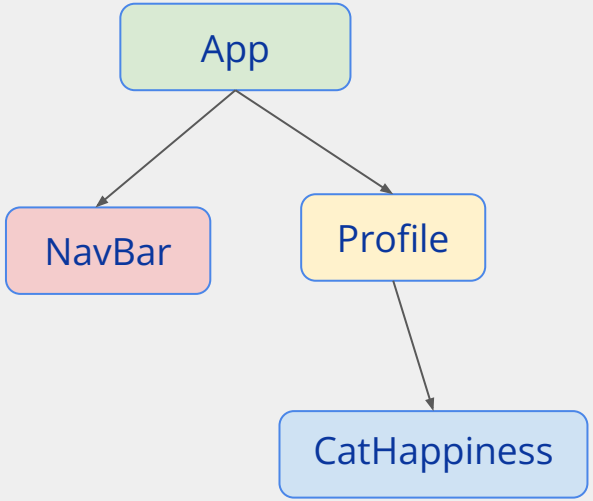
*If it doesn't let you checkout and says 'Please commit your changes or stash them', then 'git stash' should do the trick and you should be able to checkout

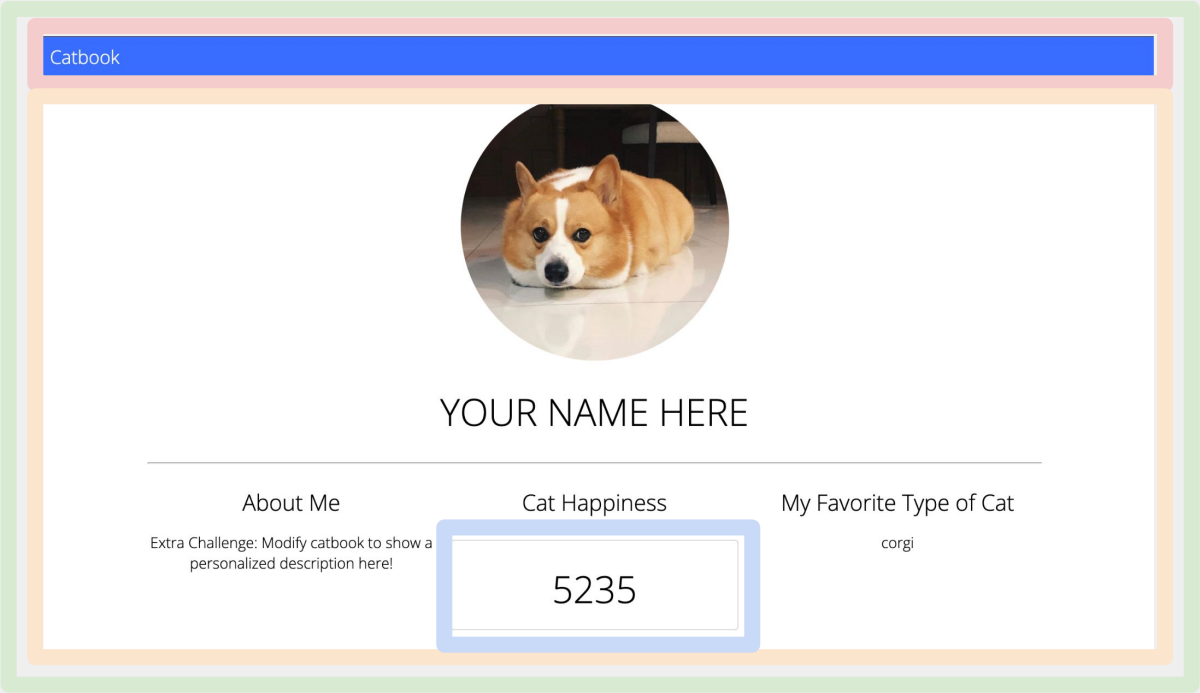
Adding CatHappiness

Which component should we store 'catHappiness' in?

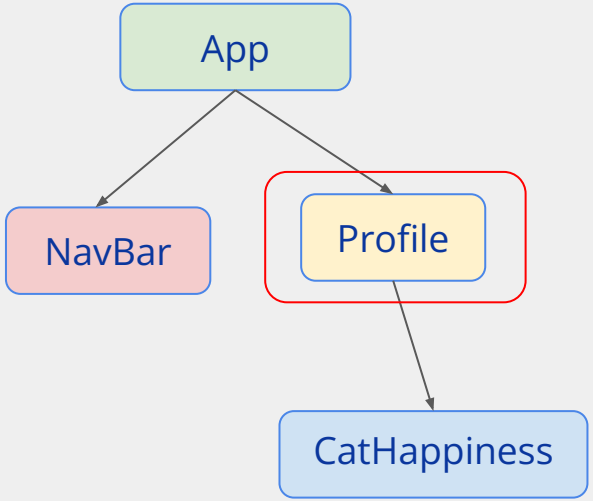
Cat Happiness

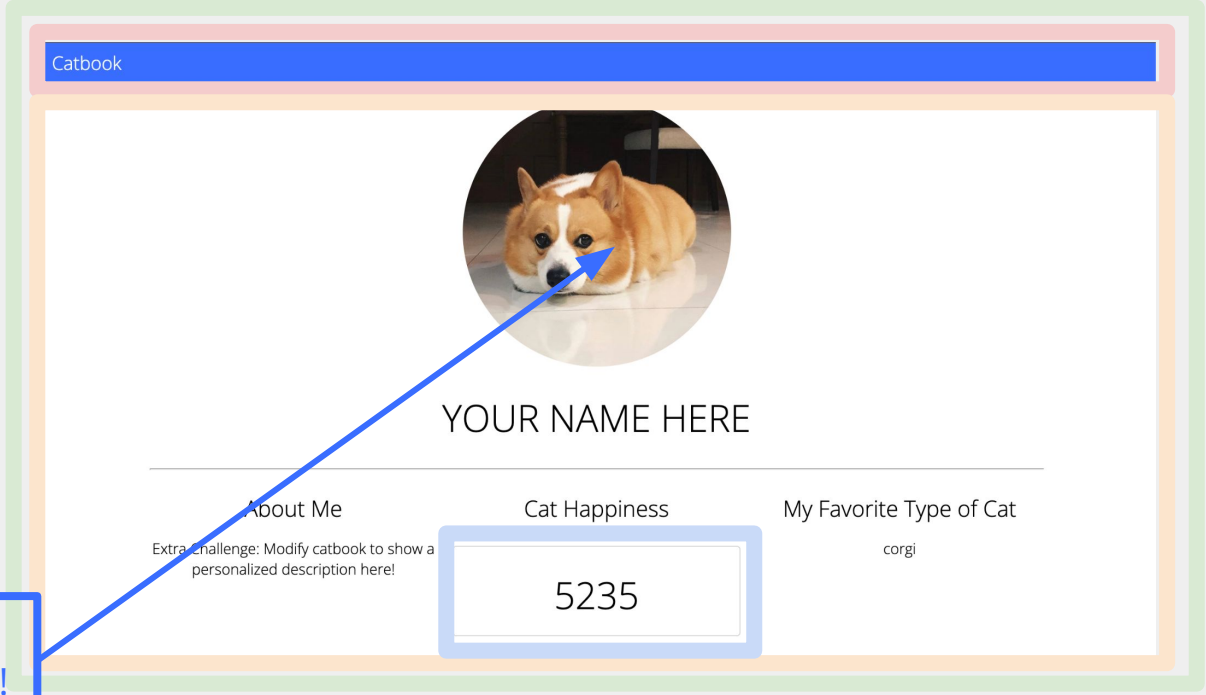
16



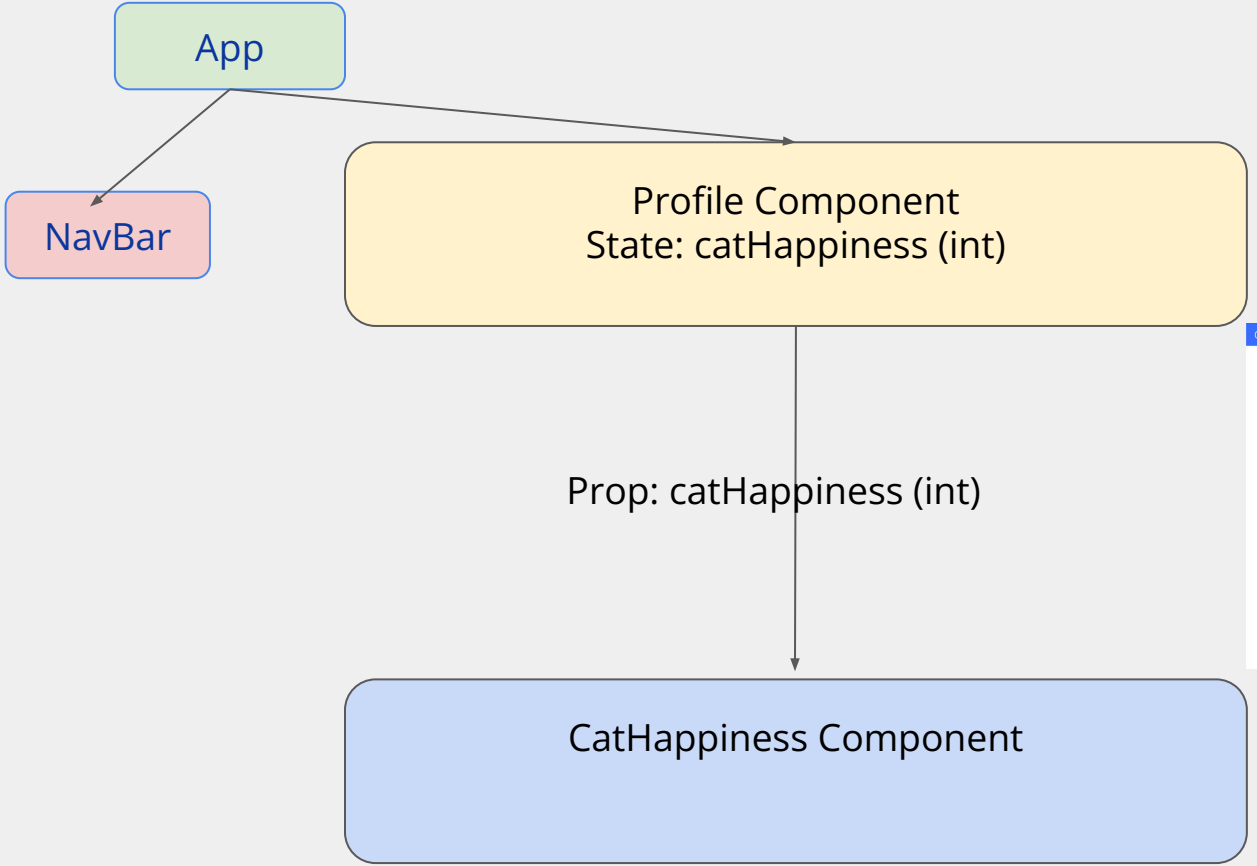


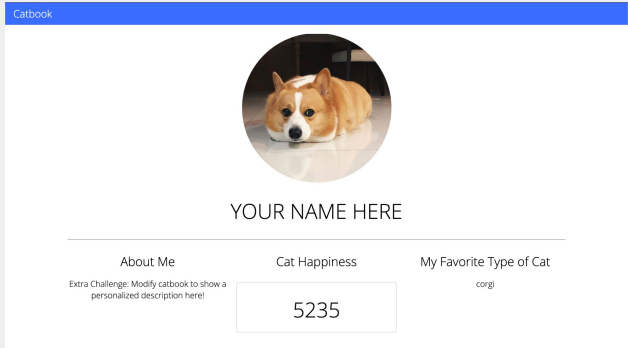
Why Profile? We want to update 'catHappiness' when the cat is clicked





Clicking on the cat needs to update catHappiness, and the cat is in Profile!


App
NavBar
Profile Component
State: catHappiness (int)
Prop: catHappiness (int)
CatHappiness Component


Catbook
YOUR NAME HERE
About Me
Extra Challenge: Modify catbook to show a personalized description here!
Cat Happiness
5235
My Favorite Type of Cat
corgi

How do we add state to a component?

How do we add state to a component?

- `const [variableName, functionThatSetsThatVariable] = useState(defaultValueOfTheVariable)`

... we use **useState**!!

Exercise 2a: Add the 'catHappiness' state variable to Profile

In profile.js.

Also don't forget to import useState (`import React, { useState } from "react"`*)*

Use the React Guide (weblab.to/react-guide-1) if you're stuck!

Exercise 2a: Add the 'catHappiness' state to Profile

// Profile.js (also don't forget to import useState)

```
const Profile = () => {
  const [catHappiness, setCatHappiness] = useState(0);
```

Exercise 2b: Import the CatHappiness Component

// Profile.js

```
import React, { useState } from "react";
import CatHappiness from "../modules/CatHappiness.js";
import "../../utilities.css";
import "./Profile.css";
```

Exercise 3a: Add the CatHappiness component

- Add in the CatHappiness component to Profile.js (in the TODO STEP 1 area), as well as a header in front of it to look like:

- Also, pass in catHappiness as a prop! Don't forget, the syntax will look like:

```
<Component propName={propValue} />
```

```
<div className="Profile-subContainer u-textCenter">
  <h4 className="Profile-subTitle">About Me</h4>
  <div id="profile-description">
    Extra Challenge: Modify catbook to show a personalized description here!
  </div>
</div>
<div className="Profile-subContainer u-textCenter">
  <h4 className="Profile-subTitle">Cat Happiness</h4>
  <CatHappiness catHappiness={catHappiness} />
</div>
<div className="Profile-subContainer u-textCenter">
  <h4 className="Profile-subTitle">My Favorite Type of Cat</h4>
  <div id="favorite-cat">corgi</div>
</div>
```

Are we done?

YOUR NAME HERE

About Me

Extra Challenge: Modify catbook to show a personalized description here!

Cat Happiness

My Favorite Type of Cat

corgi

Exercise 3b: Display the incoming CatHappiness Prop

// CatHappiness.js

```
const CatHappiness = ([redacted]) => {
  return (
    <div className="CatHappiness-container">
      <div className="CatHappiness-story">
        <p className="CatHappiness-storyContent">[redacted]</p>
      </div>
    </div>
  );
};
```

Exercise 3b: Use the incoming the CatHappiness Prop

// CatHappiness.js

```
const CatHappiness = (props) => {
  return (
    <div className="CatHappiness-container">
      <div className="CatHappiness-story">
        <p className="CatHappiness-storyContent">{props.catHappiness}</p>
      </div>
    </div>
  );
};
```

Let's get on the same page

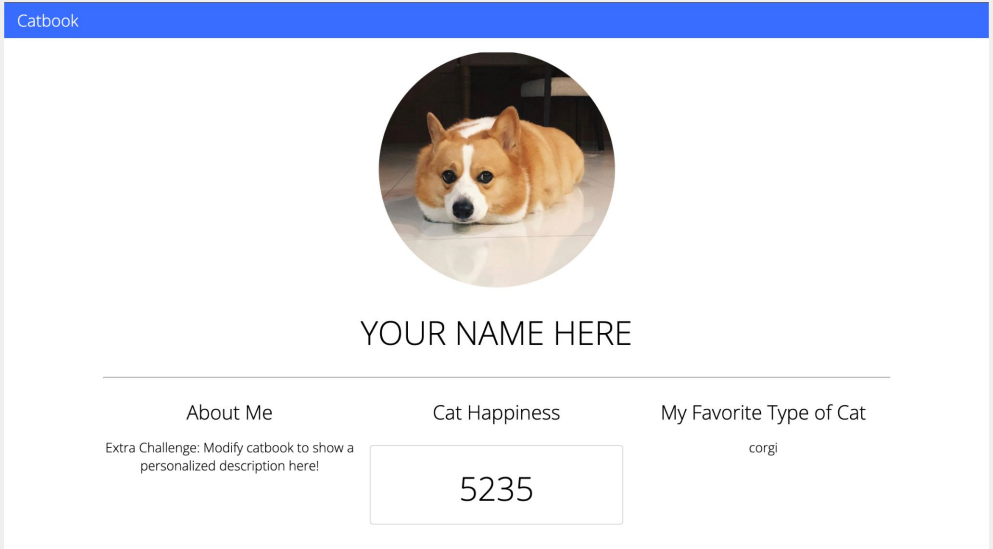
Save or close out of all of your 'unsaved' files:


```
git reset --hard
git checkout w2-step2
```

*If it doesn't let you checkout and says 'Please commit your changes or stash them', then 'git stash' should do the trick and you should be able to checkout

Exercise 4: Update CatHappiness State

Now we need to change the CatHappiness when we click!



Exercise 4: Update CatHappiness State

- (Profile.js line 9) Implement the 'incrementCatHappiness' function
  - (Profile.js line 15) Call the 'incrementCatHappiness' function whenever the profile picture is clicked

HINT: All divs have an 'onClick' prop that takes a function. Whenever a div is clicked, it runs its onClick function.

```
<div
  className="Profile-avatarContainer"
  onClick={() => {
    incrementCatHappiness();
  }}
>
```

Works, just not the most readable. Also unnecessary since we aren't doing anything else inside this function.

```
<div
  className="Profile-avatarContainer"
  onClick={incrementCatHappiness}
>
```

Works and super clean code!! Recommended implementation!

```
<div
  className="Profile-avatarContainer"
  onClick={() => {
    setCatHappiness(catHappiness + 1);
  }}
>
```

Also pretty good

```
<div
  className="Profile-avatarContainer"
  onClick={incrementCatHappiness()}
>
```

Doesn't work since it will execute the function when the div element is created, not when it's clicked on.

Let's get on the same page

Save or close out of all of your 'unsaved' files: # NavBar.css ●

```
git reset --hard
git checkout w2-complete
```

Navigate to localhost:5000 and change the cat happiness by clicking the profile picture!

Recap: Writing Components

- We divide our app into c _ _ _ _ _ _ _ _ s, and put one in each file
- Each component is a function with p _ _ ps as the input, and returns HTML-like code
- Each component can store internal updatable private info as 's _ _ _ e' variables

- **( )** allows us to enter an HTML environment
- Inside the HTML environment, **{}** allows us to create a mini j _ _ _ _ _ _ _ _ t environment

Recap: Writing Components

- We pass in props with `<Post text="I love weblab" />`
- We read in those props with `props.text`
- We declare state variables with

  `const [something, setSomething] = useState(initialValue)`
- React uses **className** instead of **class** for css styles

weblab.to/react-guide-1