





Workshop 3 - More React, using APIs, and Routing

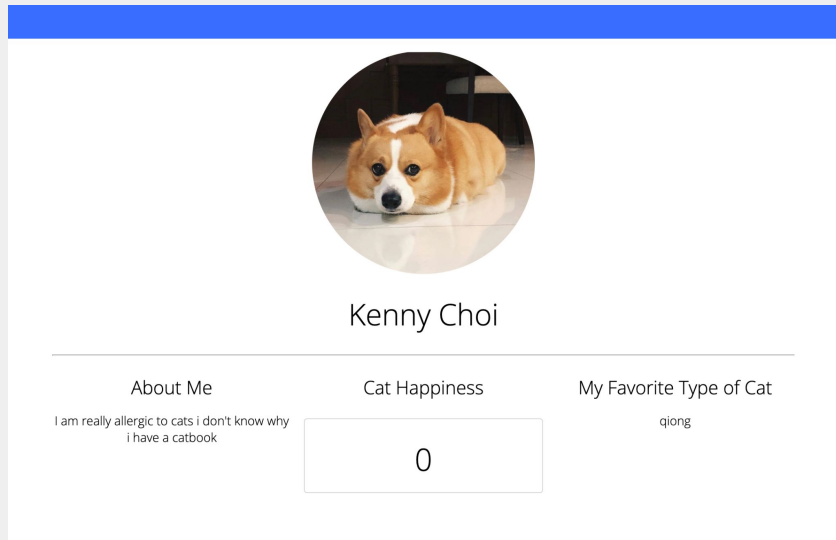
Tony Cui
Andy Jiang

Announcements

- Milestone 0 deadline extended to 5:30pm today!
- Milestone 1 (Project Pitch) signups out! Time slots are for Sat and Sun 1-5pm.
 - weblab.is/milestone1
- Fill out weblab.is/feedback! Note that the “What day is it” question is the day you’re giving feedback for, not the current day.
-     (weblab.is/milkandcookies) uwu

Previously...

- From Workshop 2
 - Profile page in React
 - Components
 - State and Props
- From Lecture
 - What is an API?
 - HTTP requests
 - Asynchronous programming (briefly)
 - useEffect, Lifecycle



This Workshop

- Build frontend for creating + retrieving posts and comments :0
- Interacting with an API from the frontend
 - GET
 - POST

The image shows a mockup of a web application interface. It features a light gray background with white form elements. At the top, there is a 'New Story' input field with a 'Submit' button. Below this, a post by 'Andrew Liu' is shown with the text 'web.labing with Tony <3'. Underneath the post, there is a 'New Comment' input field with a 'Submit' button. Further down, another post by 'Tony Cui' is shown with the text 'Send it or blend it?'. At the bottom, a post by 'Stanley Zhao | Both!' is shown with a 'New Comment' input field and a 'Submit' button.

Review: Frontend vs. Backend

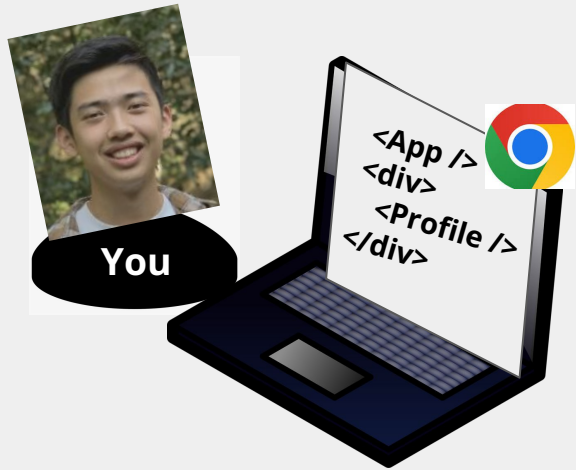


Review: Frontend vs. Backend



FRONTEND: code responsible for **displaying** what is in **front** of you.

Review: Frontend vs. Backend



FRONTEND: code responsible for **displaying** what is in **front** of you.

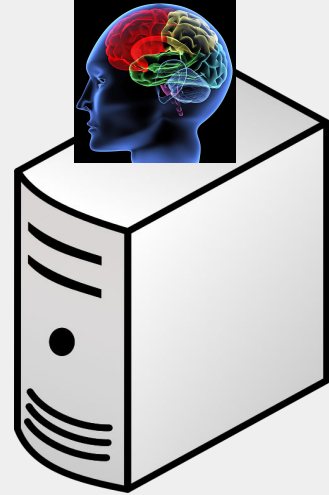
ex. React components, HTML, CSS, etc.

Review: Frontend vs. Backend



FRONTEND: code responsible for **displaying** what is in **front** of you.

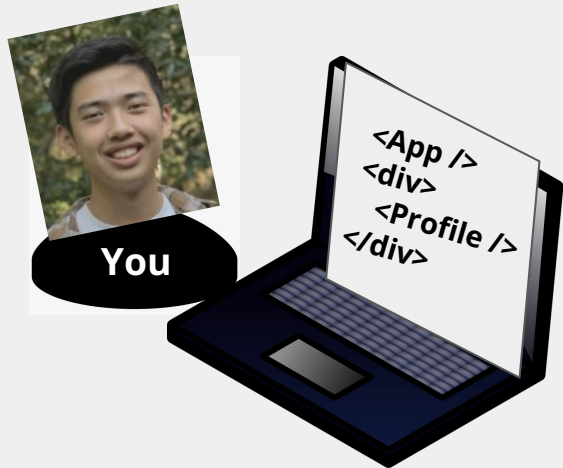
ex. React components, HTML, CSS, etc.



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

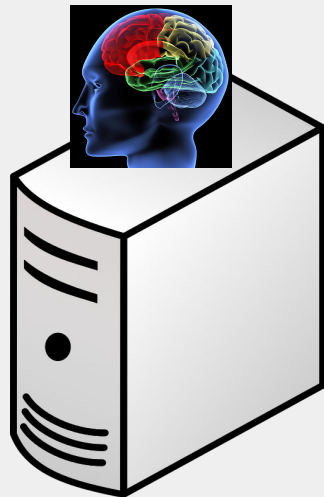
Tomorrow :)

Review: HTTP Requests: 2 main types!



FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

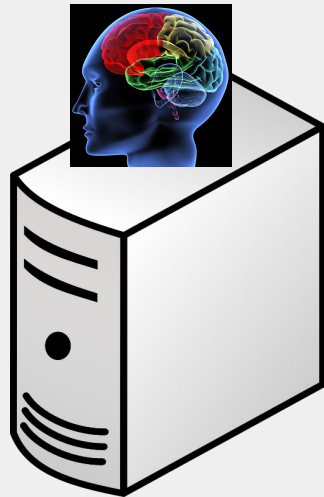
Review: HTTP Requests



FRONTEND: code responsible for **displaying** what is in **front** of you.

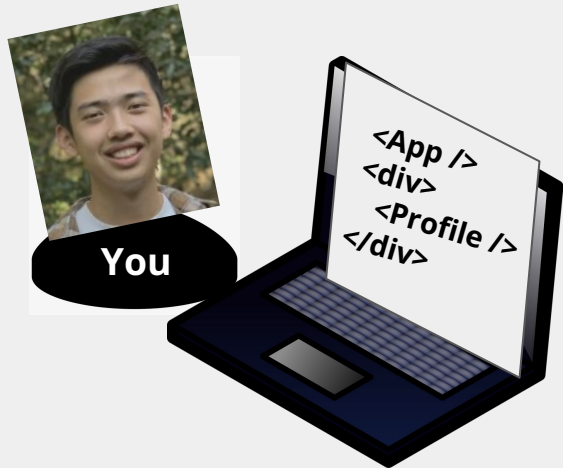
ex. React components, HTML, CSS, etc.

GET: hello give me info on XXX



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

Review: HTTP Requests



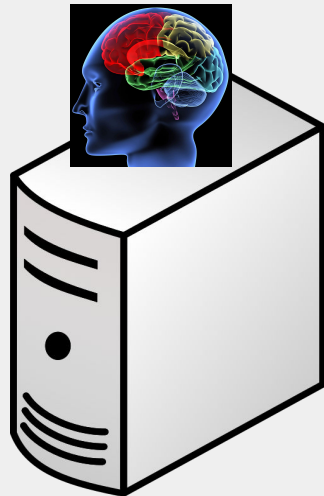
FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.

GET: hello give me info on XXX

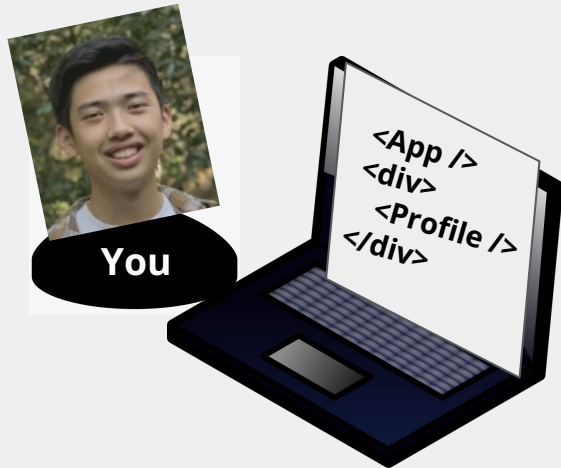


Response: hello here is info on XXX



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

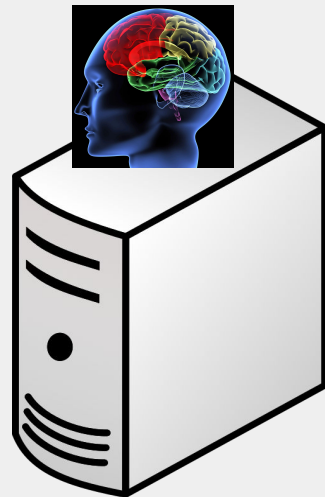
Review: HTTP Requests



FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.

POST: hello here is some new info on XXX



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

How does the backend
exactly know we want?

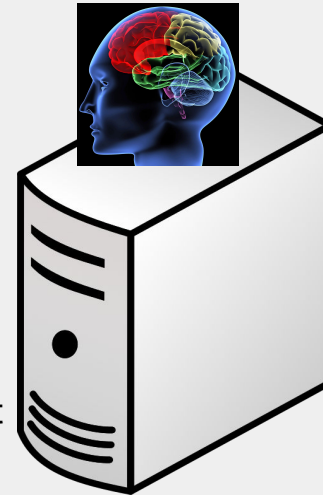
Review: HTTP Requests

API ENDPOINT specification of the input parameters and output response

● /stories

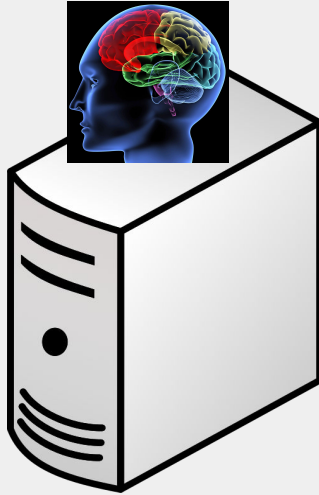
● /story

● /comment



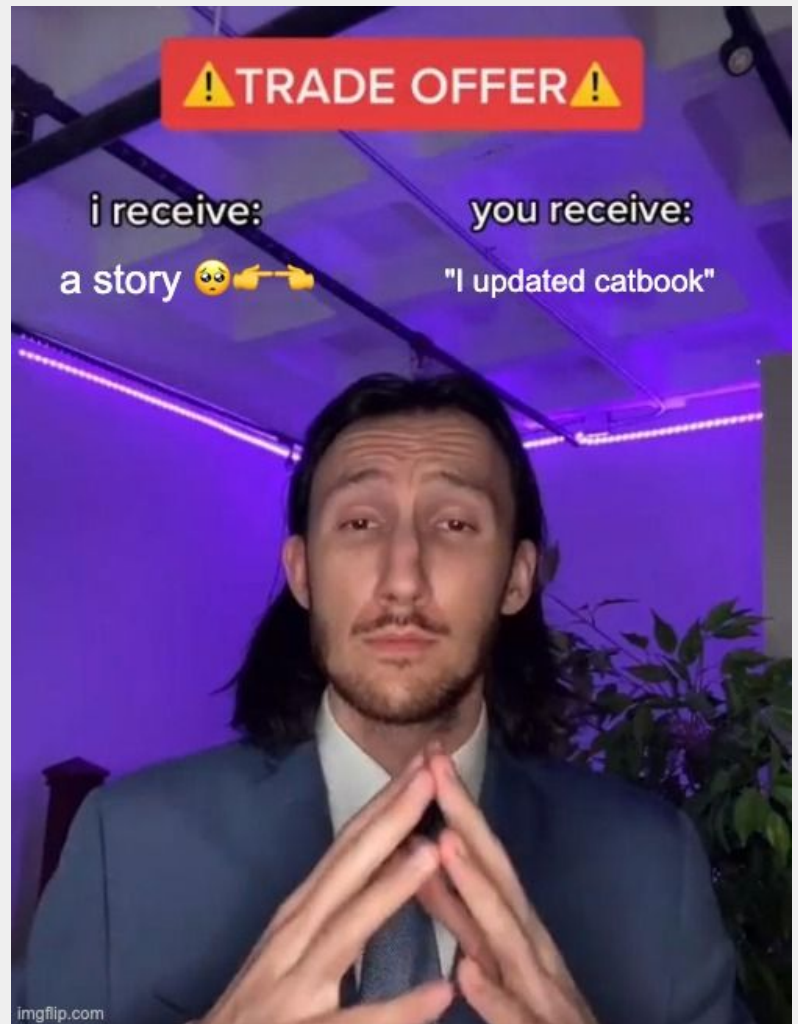
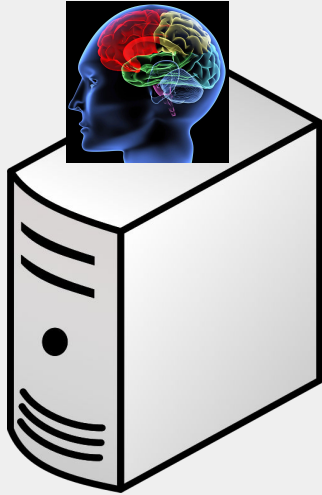
Review: HTTP Requests

● /stories

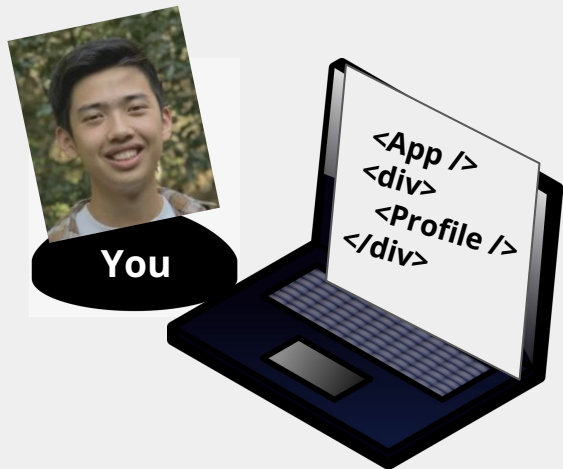


Review: HTTP Requests

● /story



Review: HTTP Requests



FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.

POST: '/story' take my new story!
GET: '/stories'

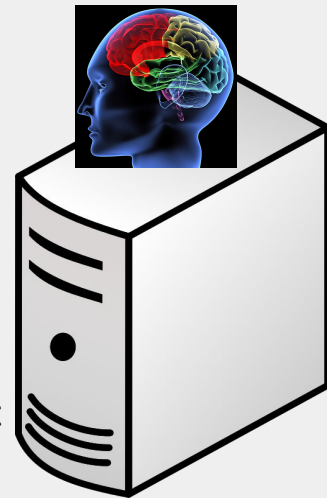
API ENDPOINT specification of the input parameters and output response

GET Response: here u go {stories}

● /stories

● /story

● /comment



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

API Endpoint

FRONTEND POV (Client or API User)

An API endpoint is a black box with specified inputs and outputs as given.

GET Request

- *Input:* Query parameters frontend should send to backend
- *Output:* What kind of information (format) frontend should expect as response from backend

POST Request

- *Input:* What kind of updated information (format) frontend should send to backend

BACKEND POV (Server)

Defines the inputs and outputs of an API endpoint. Also defines how inputs will be handled and how outputs are obtained.

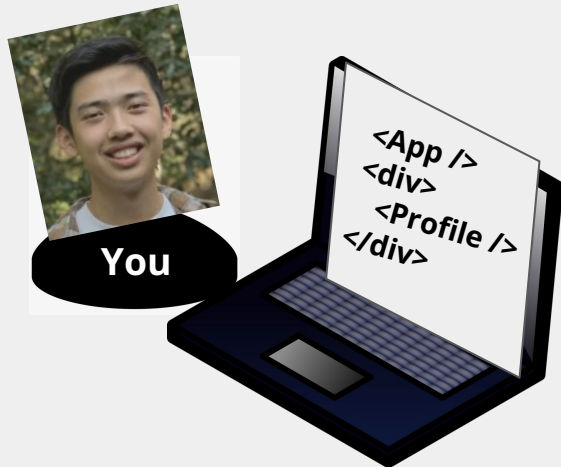
[Inside] GET Endpoint

- Code that defines how backend gets information requested by frontend

[Inside] POST Endpoint

- Code that defines how backend handles the information provided by frontend.

Review: HTTP Requests



FRONTEND: code responsible for **displaying** what is in **front** of you.

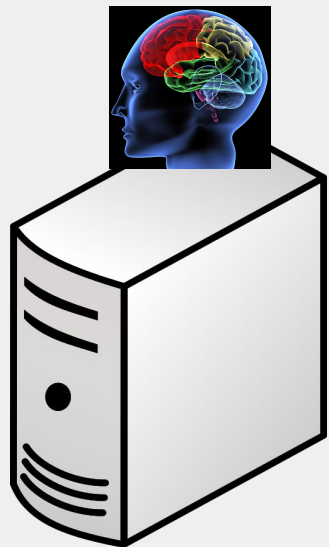
ex. React components, HTML, CSS, etc.

POST /stories: hello here is a new story
GET /stories: hello give me all the stories

API ENDPOINT specification of the input parameters and output response

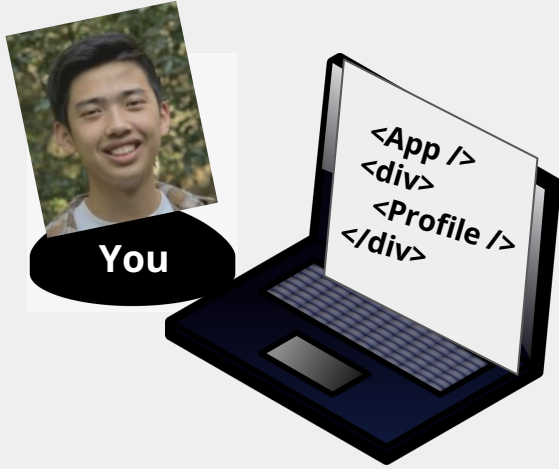
GET /stories Response: hello
here's all the stories

● /stories



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any **new info** from frontend

Review: Frontend vs. Backend



FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.

POST /stories: hello here is a new story

GET /stories: hello give me all the stories

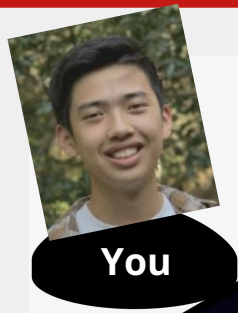
API ENDPOINT specification of the input parameters and output response

GET /stories Response: hello here's all the stories



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** new info from frontend

Review: Frontend vs. Backend



You



FRONTEND: code responsible for **displaying** what is in **front** of you.

ex. React components, HTML, CSS, etc.

POST /stories: hello here is a new story

GET /stories: hello give me all the stories

API ENDPOINT specification of the input parameters and output response

GET /stories Response: hello here's all the stories

/stories

/post

/comment



BACKEND: code working **behind** the scenes responsible for **giving** frontend **info** it needs to display and **updating** any new info from frontend

This Workshop!

Review: Flow of an HTTP request from Frontend



GET
/stories

Tony Cui

I really miss Andrew 🥺

Tony Cui | (Hopefully I can see him tomorrow during Weblab)

Why u such a simp fr

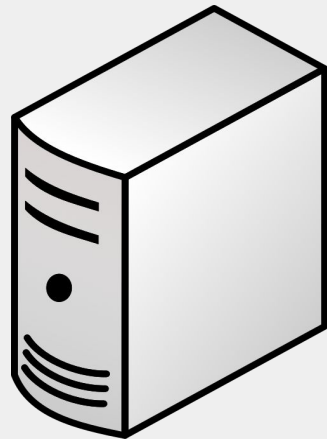
```
{content: "I really miss Andrew 🥺!"  
creator_name: "Tony Cui"  
_id: "5a591353c26863287c2bd311"}
```

API

/stories

/post

/comment



Getting started

- Navigate to catbook-react repository
- `git reset --hard`
- `git checkout w3-starter`
- `npm install`

`weblab.is/questions` <- general questions

`weblab.is/q` <- individual help

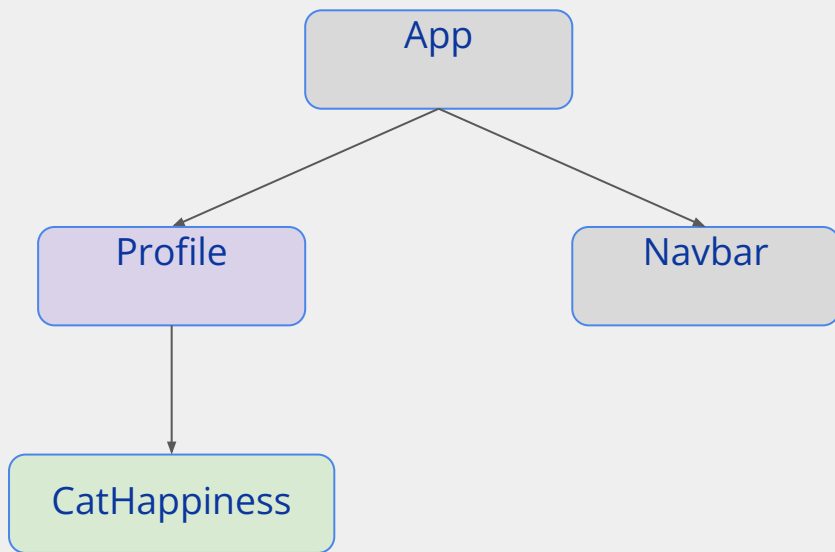
```
git reset --hard  
git checkout w3-starter  
npm install
```

Overview

- Similar folder structure to last time
- New file!
 - `utilities.js`

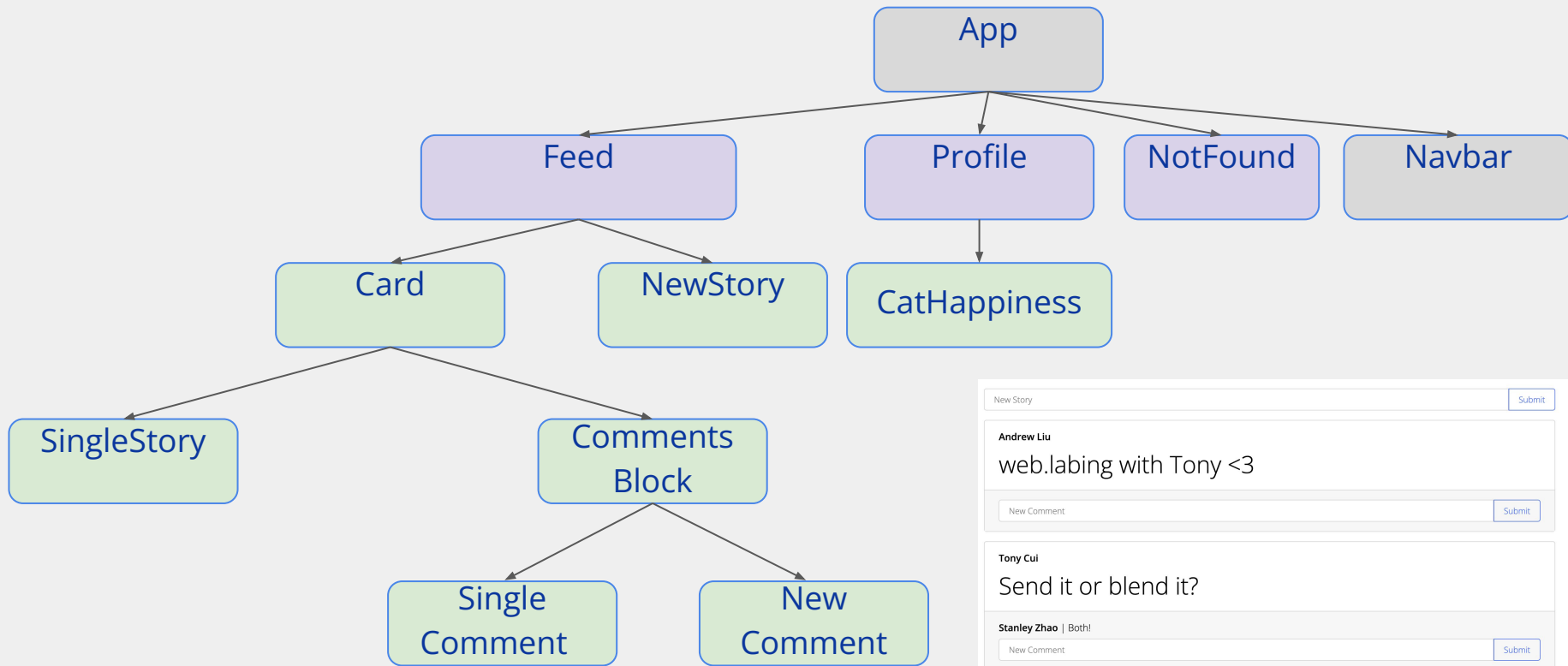
Recall: component hierarchy before

```
git reset --hard  
git checkout w3-starter  
npm install
```



Component hierarchy after today!

```
git reset --hard  
git checkout w3-starter  
npm install
```



Andrew Liu

web.labing with Tony <3

Tony Cui

Send it or blend it?

Stanley Zhao | Both!

Goals for today

```
git reset --hard  
git checkout w3-starter  
npm install
```

- The Feed page
 - Ability to see other posts and comments
 - GET
 - Ability to make your own post and comments
 - POST
- Frontend Routing

The screenshot displays a web interface for a feed. At the top, there is a 'New Story' input field with a 'Submit' button. Below this, the first post is by 'Andrew Liu' with the text 'web.labing with Tony <3'. Underneath the post is a 'New Comment' input field with a 'Submit' button. The second post is by 'Tony Cui' with the text 'Send it or blend it?'. At the bottom, there is a header for 'Stanley Zhao | Both!' followed by another 'New Comment' input field and a 'Submit' button.

Let's go!

```
git reset --hard  
git checkout w3-starter  
npm install
```

- `npm run hotloader`
- In a browser, head to 'localhost:5050'
- Should look a lot like workshop 2

Step 0:

Warmup

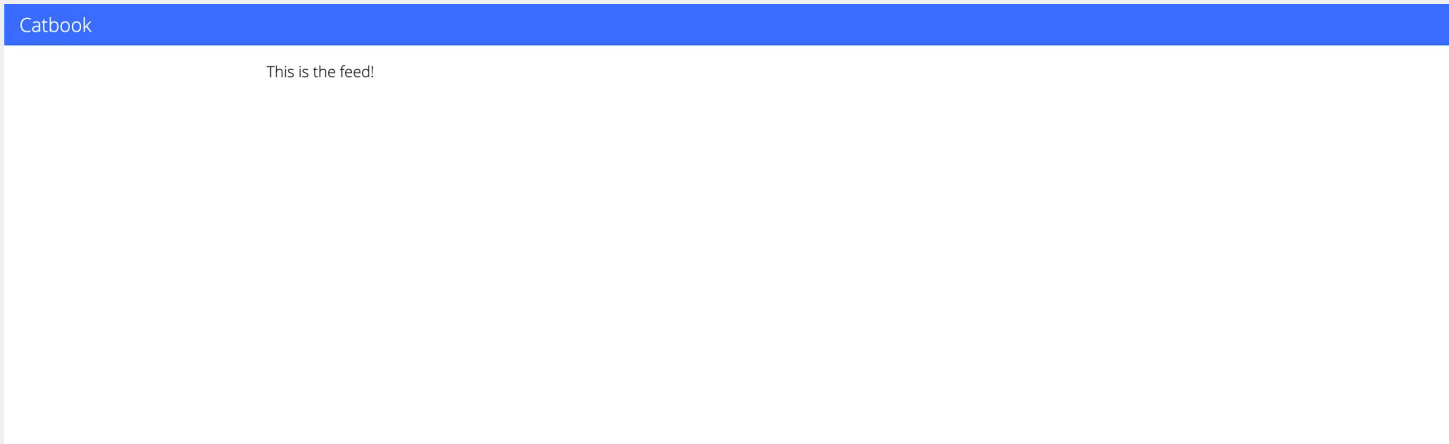
Questions?

```
git reset --hard  
git checkout w3-starter  
npm run hotloader
```

```
git reset --hard  
git checkout w3-starter  
npm run hotloader
```

Warmup

- In App.js
 - 1) Import Feed.js
 - 2) Render Feed instead of Profile



Step 1:

Get Stories

Questions?

```
git reset --hard  
git checkout w3-step1
```

```
git reset --hard  
git checkout w3-step1
```

Follow along: step 1

- In `Feed.js`
 - Let's get (GET) stories!
- Follow along with me
 - Hold stories in state
 - Get stories in `useEffect`
 - Return a render-able list of stories

GET /api/stories

Returns all stories

Parameters: none

JSON.stringify(*list*) makes a list render-able

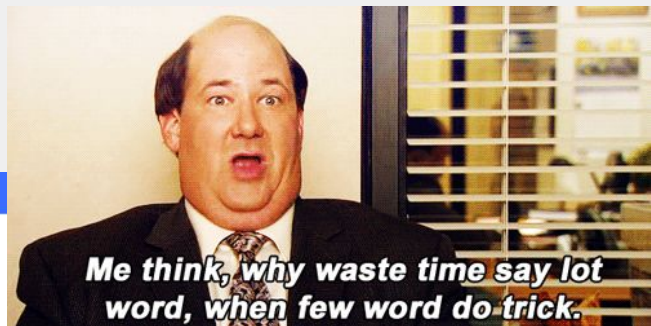

```
git reset --hard  
git checkout w3-step1
```

Follow along: step 1

- Testing our GET

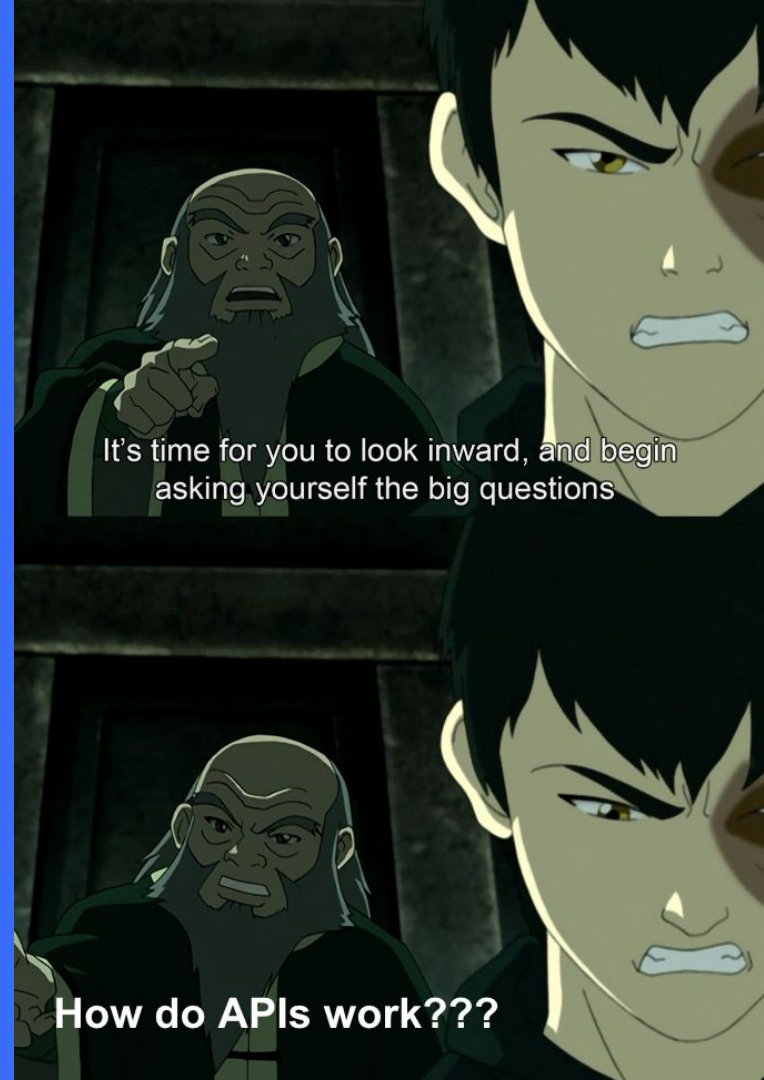
Catbook

```
{ "_id": "5ff6aca7c2524eeb167e18ba", "creator_name": "Anonymous User", "content": "testing anon", "_v": 0 },  
{ "_id": "5ff6b0fc619219ef6a801388", "creator_name": "Anonymous User", "content": "fdsa", "_v": 0 },  
{ "_id": "5ff6b774e59ebcf1df04676d", "creator_name": "Anonymous User", "content": "asd", "_v": 0 },  
{ "_id": "5ff6bbdd98a89700245e9d70", "creator_name": "Anonymous User", "content": "fdsa", "_v": 0 },  
{ "_id": "5ff6bbe398a89700245e9d71", "creator_name": "Anonymous User", "content": "asd", "_v": 0 },  
{ "_id": "5ff6bd8d0c09340024708427", "creator_name": "Anonymous User", "content": "asd", "_v": 0 },  
{ "_id": "610c20576363e0002439024c", "creator_name": "Anonymous User", "content": "hhhhh", "_v": 0 },  
{ "_id": "610c206c6363e0002439024e", "creator_name": "Anonymous User", "content": "aa", "_v": 0 },  
{ "_id": "610c207f6363e00024390250", "creator_name": "Anonymous User", "content": "hello aziz", "_v": 0 },  
{ "_id": "61d7260d43623211b07344f9", "creator_name": "Anonymous User", "content": "is working?", "_v": 0 },  
{ "_id": "61d73aa0a218200025e28b57", "creator_name": "Anonymous User", "content": "hallo", "_v": 0 },  
{ "_id": "61d765425df08e00255d8a64", "creator_name": "Anonymous User", "content": "what about this", "_v": 0 },  
{ "_id": "61d767f35df08e00255d8b0e", "creator_name": "Anonymous User", "content": "live?", "_v": 0 },  
{ "_id": "61d768115df08e00255d8b1e", "creator_name": "Anonymous User", "content": "again", "_v": 0 },  
{ "_id": "61d768da5df08e00255d8b2f", "creator_name": "Anonymous User", "content": "yeet", "_v": 0 },  
{ "_id": "61d769225df08e00255d8b41", "creator_name": "Anonymous User", "content": "test", "_v": 0 },  
{ "_id": "61d76a2c5df08e00255d8b54", "creator_name": "Anonymous User", "content": "try live", "_v": 0 },  
{ "_id": "61d76b755df08e00255d8bf8", "creator_name": "Anonymous User", "content": "try again", "_v": 0 },  
{ "_id": "61d76bf45df08e00255d8c33", "creator_name": "Anonymous User", "content": "try again", "_v": 0 },  
{ "_id": "61d76c855df08e00255d8c49", "creator_name": "Anonymous User", "content": "now is the time", "_v": 0 },  
{ "_id": "61d76cb65df08e00255d8c62", "creator_name": "Anonymous User", "content": "maybe now", "_v": 0 },  
{ "_id": "61d76d025df08e00255d8c92", "creator_name": "Anonymous User", "content": "live reloading!", "_v": 0 },  
{ "_id": "629efcc9c3f0aa0025e9fc60", "creator_name": "Anonymous User", "content": "Are you serious", "_v": 0 },  
{ "_id": "63494ba8c50e9a00256edfd8", "creator_name": "Anonymous User", "content": "hello, everyone", "_v": 0 },  
{ "_id": "63a75247cd3fb60023f565d8", "creator_name": "Anonymous User", "content": "wassup", "_v": 0 },  
{ "_id": "63be26f13b82000230773b4", "creator_name": "Anonymous User", "content": "pa5", "_v": 0 }
```



Step 2: Single Story Questions?

```
git reset --hard  
git checkout w3-step2
```



It's time for you to look inward, and begin asking yourself the big questions

How do APIs work???

```
git reset --hard  
git checkout w3-step2
```

Your turn: step 2

- In `Feed.js`
 - Import `SingleStory.js`
 - Render a `SingleStory` component, passing in dummy name and content

```
git reset --hard  
git checkout w3-step2
```

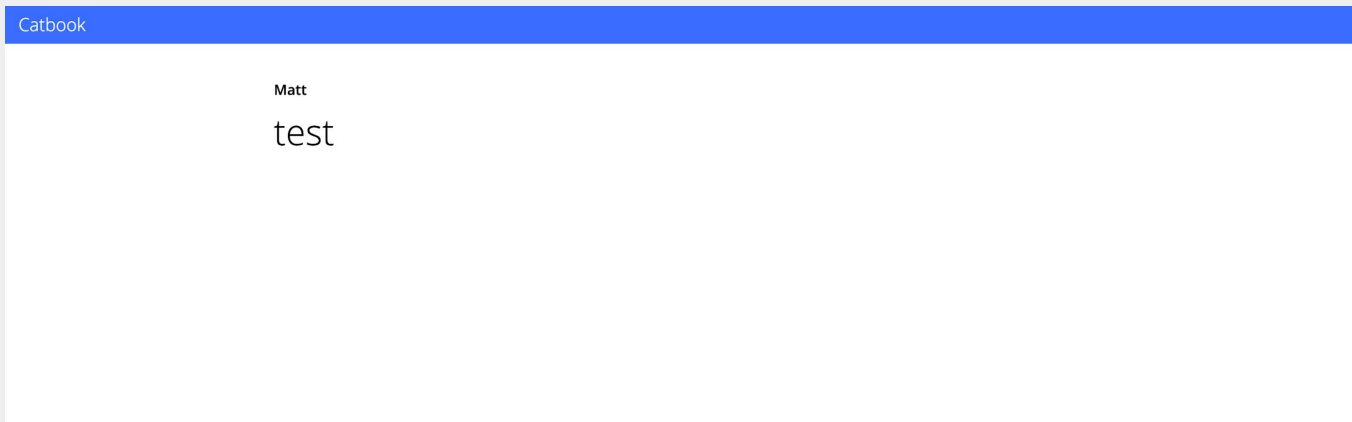
Your turn: step 2

- In `SingleStory.js`

- Let's make our stories prettier!
- Use JSX to render passed in props

- CSS classes to use:

- `Card-story`
- `u-bold`
- `Card-storyContent`



Step 3:

Multiple Stories

Questions?

```
git reset --hard  
git checkout w3-step3
```

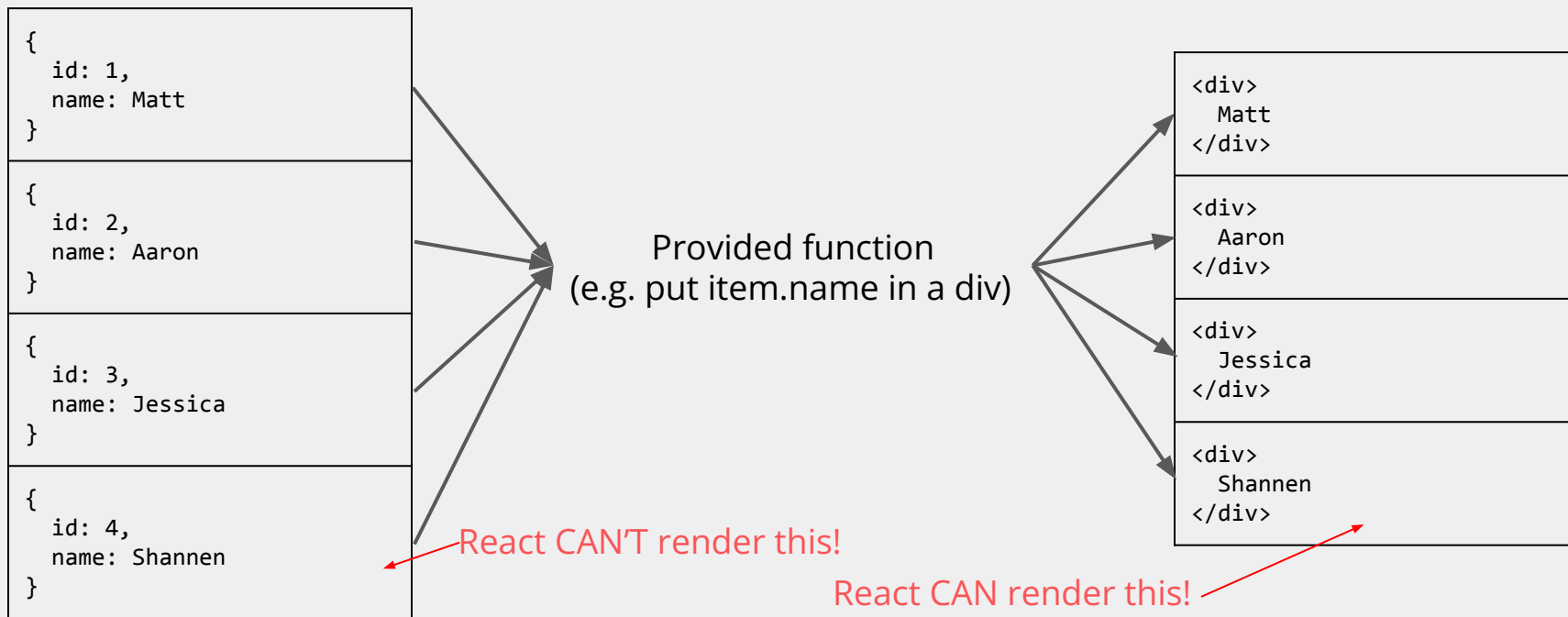
```
git reset --hard  
git checkout w3-step3
```

Follow along: step 3

- In `Feed.js`
 - Putting our state and `SingleStory` together
 - Object array -> Component array?

Review: the map function

- Recall: the `map()` method creates a new array by applying a function to every element of the starting array.



```
git reset --hard  
git checkout w3-step3
```

Follow along: step 3

- In `Feed.js`
 - Putting our state and `SingleStory` together
 - Object array -> Component array?

Catbook

Anonymous User

testing anon

Anonymous User

fdsa

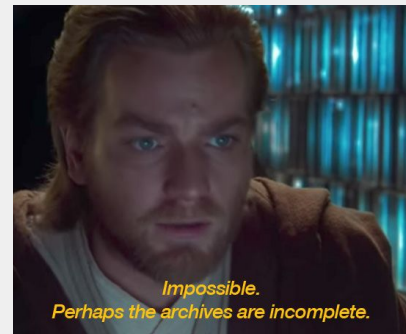
Anonymous User

asdf


```
git reset --hard  
git checkout w3-step3
```

Follow along: step 3

- In `Feed.js`
 - What to do if no stories?



Catbook

No stories!

Step 4:

New Story

Questions?

```
git reset --hard  
git checkout w3-step4
```

```
git reset --hard  
git checkout w3-step4
```

That was GET, how about POST?

POST /api/story

Posts a new story

Parameters: content (string)

```
git reset --hard  
git checkout w3-step4
```

Follow along: step 4

- In `NewPostInput.js`
 - Let's make a component that can post (POST) stories
 - POST `/api/story`**
Posts a new story
Parameters: **content** (string)
- Follow along with me
 - The `NewStory` component
 - Using it in `Feed`

```
git reset --hard  
git checkout w3-step4
```

Follow along: step 4

Catbook

Aaron Sipser

Hello world!

When you just want 1 new story

My backend code:



```
git reset --hard  
git checkout w3-step4
```





Follow along: step 4

- Questions?
 - Please ask! We'll have you do something similar to steps 1-4 later!
- Catch up
 - `git reset --hard`
 - `git checkout w3-step5`

Announcements

- 🙌🙌 Please give us feedback! weblab.is/feedback
- 🕒 **Milestone 0 (team creation, 10 ideas) due at midnight tonight!**
- 🥛🍪 Talk to academic chairs! weblab.is/milkandcookies
 - Any questions, concerns, feels :))
- 📝 Homework 1 (Optional, for react practice) on weblab.is/home
- 🎥 Recordings at weblab.is/recordings
 - Let us know on [milkandcookies](https://weblab.is/milkandcookies) if any recordings are hard to use (Git, JS callbacks don't have the board showing :pp)
 - If you had trouble following the Figma workshop yesterday, Joyce made a video of it here! weblab.is/figma-workshop (it's also in the resources section of weblab.is/home)
- 🧠 Office hours tonight! 7-9pm in 32-082
- 🧠 Casual office hours for the next half hour right here!

Announcements

- Milestone 0 deadline extended to 5:30pm today!
- Milestone 1 (Project Pitch) signups out! Time slots are for Sat and Sun 1-5pm.
 - weblab.is/milestone1
- Fill out weblab.is/feedback! Note that the “What day is it” question is the day you’re giving feedback for, not the current day.
-     (weblab.is/milkandcookies) uwu

Workshop 3c

Workshop 3 Review

- Step 0: Replacing Profile with Feed Component
- Step 1: Declare State, GET to load all of our comments from the database
- Step 2: Rendering SingleStory Component with Dummy Data
- Step 3: Using .map() to map Story Data (from GET) to Single Stories
- Step 4: Wrap NewPostInput in a new Component + Functionality to Post

let's continue today :)

(Follow Along)

Step 5:

Adding Navigation

Questions?

```
git reset --hard  
git checkout w3-step5
```

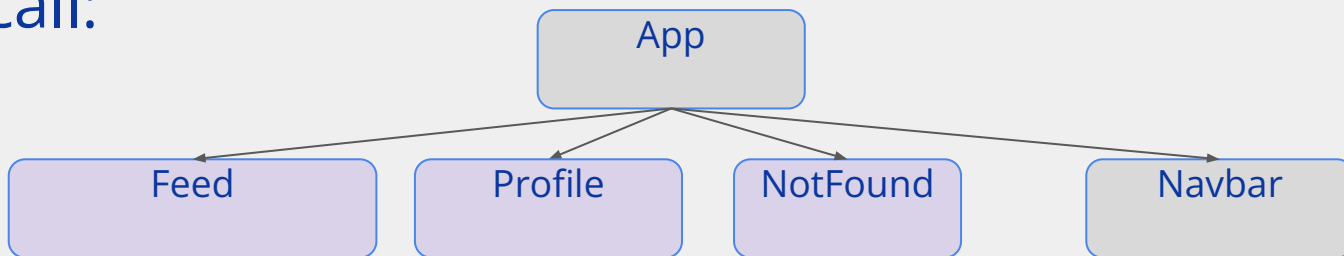
Catbook



Catbook Home Profile

App navigation?

- Recall:



- Determine content via **routing!**

- base URL + **route path**

- catbook.com
- catbook.com/
- catbook.com/**profile**
- catbook.com/**asdf**

Routing

Reach Router

We will use the **Reach Router library**
[\(read more here\)](#)

Reach Router

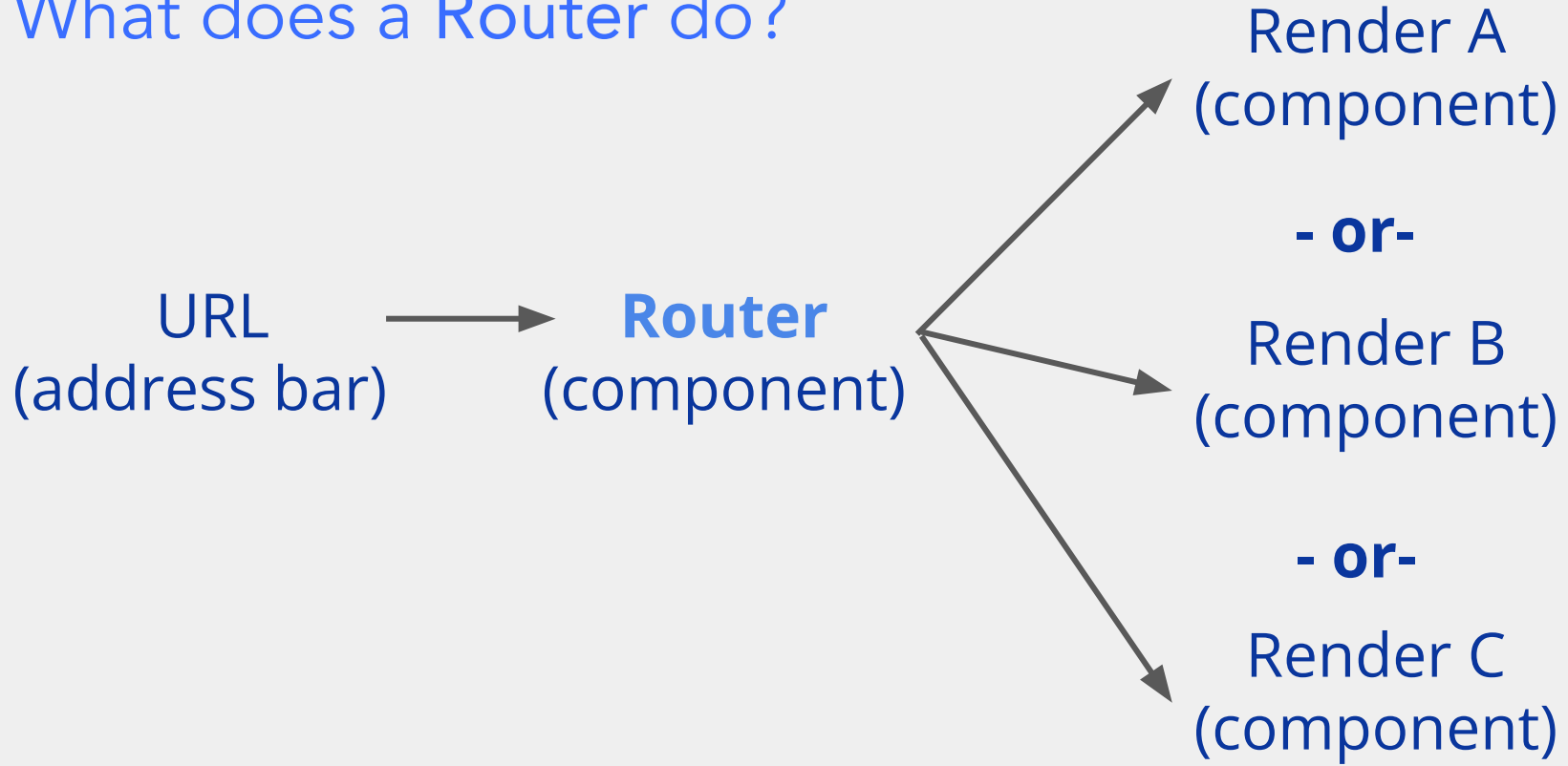
We will use the **Reach Router library**
[\(read more here\)](#)

`<Router />`

and

`<Link />`

What does a Router do?



Routes: Example

```
<App>  
  <Router>  
    <Home path="/" />  
    <Dashboard path="/dashboard" />  
    <Team path="/team" />  
  </Router>  
</App>
```

Routes: Example

All of these are **paths**

```
<App>
```

```
  <Router>
```

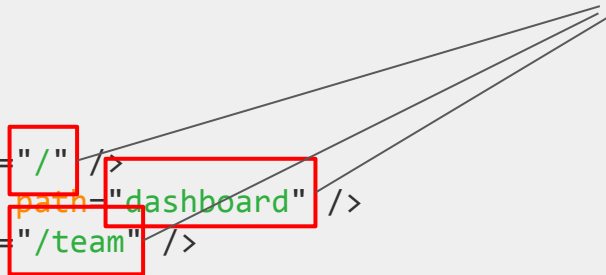
```
    <Home path="/" />
```

```
    <Dashboard path="dashboard" />
```

```
    <Team path="/team" />
```

```
  </Router>
```

```
</App>
```



Routes: Example

```
<App>
```

```
  <Router>
```

```
    <Home path="/" />
```

```
    <Dashboard path="dashboard" />
```

```
    <Team path="/team" />
```

```
  </Router>
```

```
</App>
```

Relative path: NO leading slash "relative"

→ append "relative" to the end of your current URL

google.com/maps → google.com/maps/relative

Routes: Example

```
<App>
```

```
  <Router>
```

```
    <Home path="/" />
```

```
    <Dashboard path="dashboard" />
```

```
    <Team path="/team" />
```

```
  </Router>
```

```
</App>
```

Relative path: NO leading slash "relative"

→ append "relative" to the end of your current path

google.com/maps → google.com/maps/relative

Absolute path: Leading slash "/absolute"

→ *always* go to root path + "/absolute"

(suppose google.com is root path)

google.com/absolute

Routes: Example

<App>

<Router>

<Home path="/" />

<Dashboard path="dashboard" />

<Team path="/team" />

</Router>

</App>

Root path: "/"

Entry point of your website

Refers to the **root** of the application

Routes: Example

Suppose our base URL is **"sports.com"**

```
<App>  
  <Router>  
    <Home path="/" />  
    <Dashboard path="dashboard" />  
    <Team path="/team" />  
  </Router>  
</App>
```



```
// "sports.com"  
  
<App>  
  <Home />  
</App>  
  
// "sports.com/dashboard"  
<App>  
  <Dashboard />  
</App>  
  
// "sports.com/team"  
<App>  
  <Team />  
</App>
```

WARNING: Routing can be hard!

Nested Routes

```
<App>
  <Router>
    <Root path="/">
      <Home path="/" />
      <Dashboard path="dashboard">
        <DashboardHome path="/" />
        <Trends path="trends" />
        <Graphs path="graphs" />
      </Dashboard>
      <Team path="/team">
        <TeamHome path="/" />
        <Profile path=":userId" />
      </Team>
    </Root>
  </Router>
</App>
```



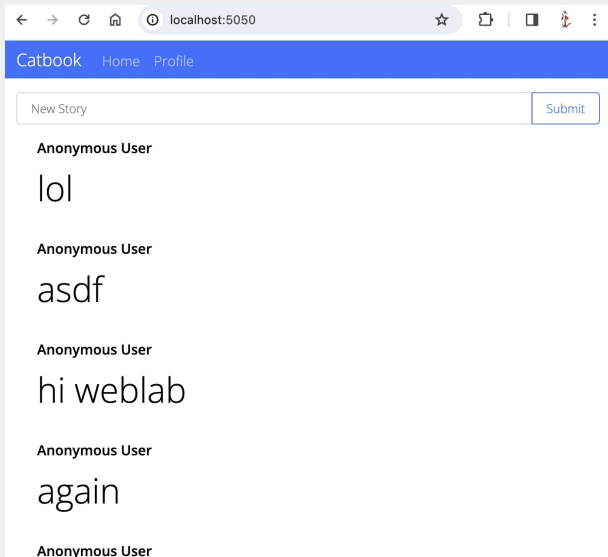
```
// "/"
<Root>
  <Home />
</Root>

// "/dashboard"
<Root>
  <Dashboard>
    <DashboardHome />
  </Dashboard>
</Root>

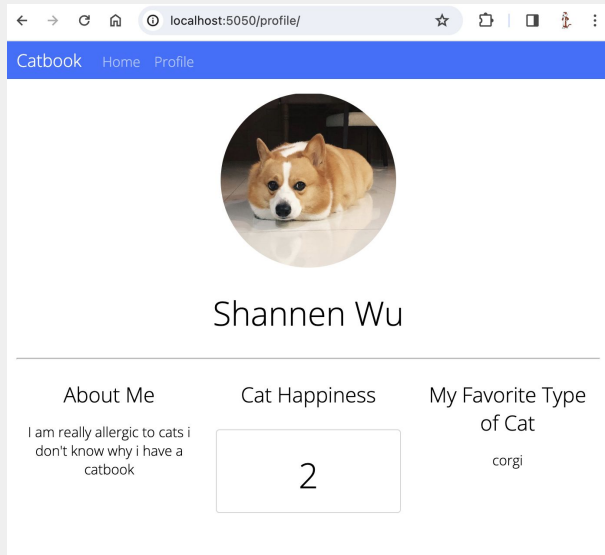
// "/dashboard/trends"
<Root>
  <Dashboard>
    <Trends />
  </Dashboard>
</Root>

// "/team/123"
<Root>
  <Team>
    <Profile userId="123" />
  </Team>
</Root>
```

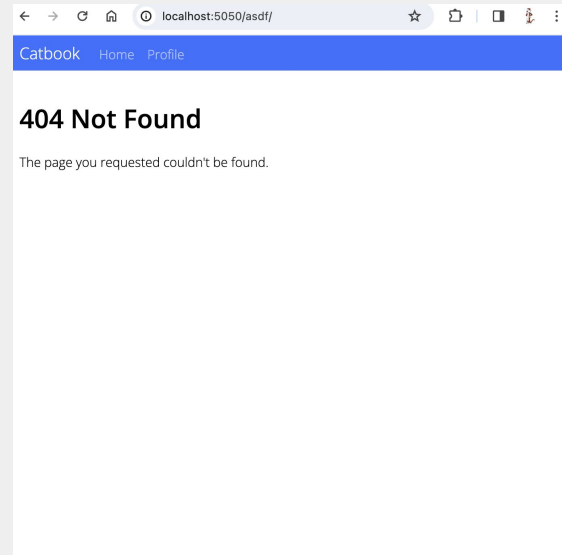

Catbook AFTER Step 5



localhost:5050



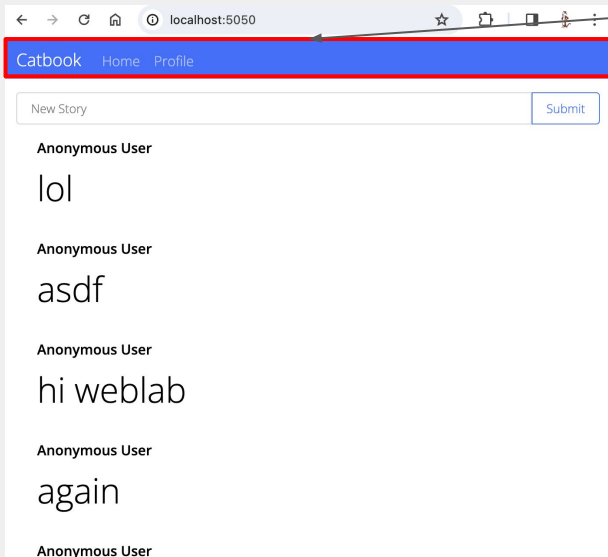
localhost:5050/profile



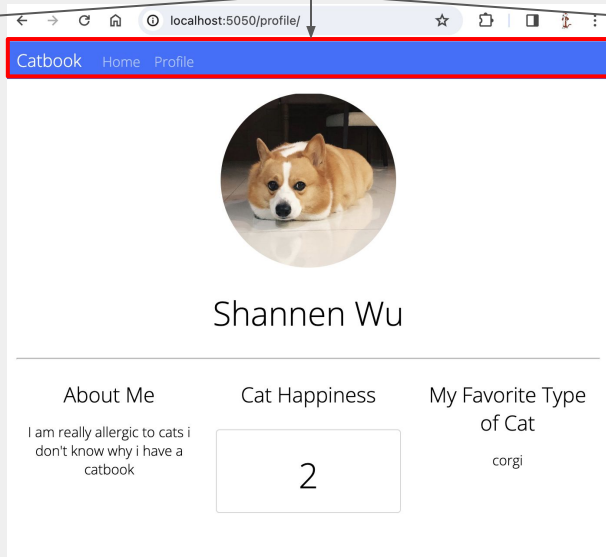
localhost:5050/anythingelse

Catbook AFTER Step 5

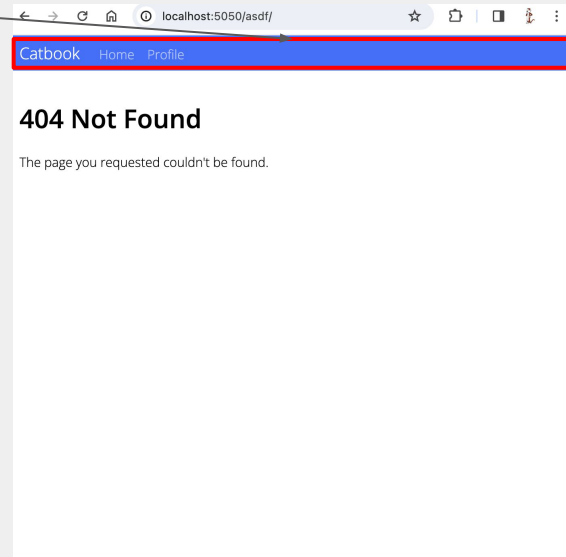
They all have the `<NavBar />` component!



localhost:5050

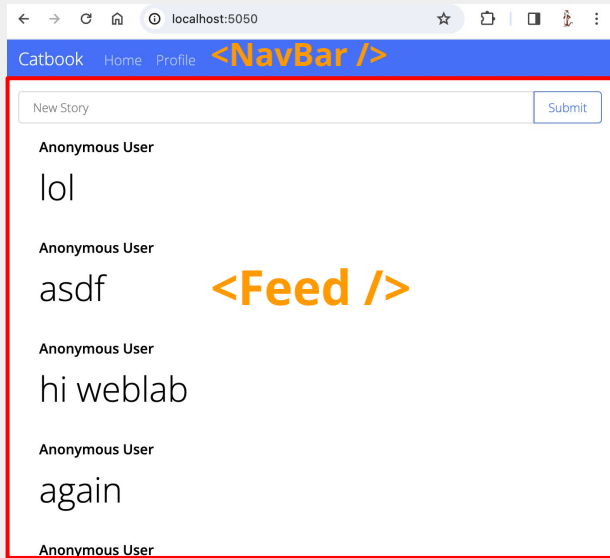


localhost:5050/profile

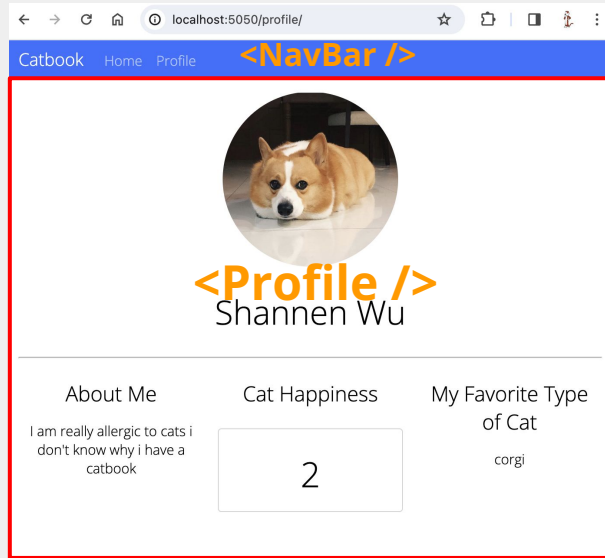


localhost:5050/anythingelse

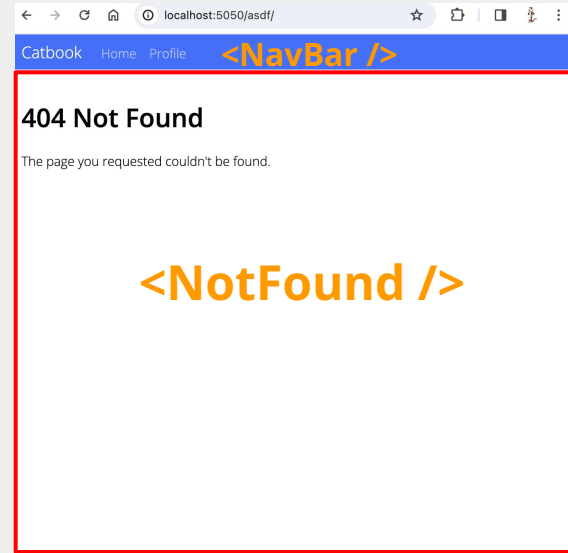
Catbook AFTER Step 5



localhost:5050



localhost:5050/profile



localhost:5050/anythingelse

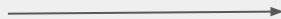


Routes: Catbook

```
<App>  
  <NavBar />  
  <Router>  
    <Feed path="/" />  
    <Profile path="profile" />  
    <NotFound default />  
  </Router>  
</App>
```

Routes: Catbook

```
<App>
  <NavBar />
  <Router>
    <Feed path="/" />
    <Profile path="profile" />
    <NotFound default />
  </Router>
</App>
```



```
// "/"
<App>
  <NavBar />
  <Feed />
<App />

// "/profile"
<App>
  <NavBar />
  <Profile />
<App />

// "/dkjfalldksfj"
<App>
  <NavBar />
  <NotFound />
<App />
```

Link

Relative: `<Link to="relative">Click me</Link>`

Absolute: `<Link to="/absolute">Click me</Link>`

Link

Current URL: localhost:5050/profile

Relative:

```
<Link to="/newpage">Click me</Link>
```

[Click me](/newpage)

Absolute:

```
<Link to="http://localhost:5050/newpage">Click me</Link>
```

[Click me](http://localhost:5050/newpage)

Link

Current URL: localhost:5050/profile

Relative:

`<Link to="/newpage">Click me</Link>`

URL goes to



localhost:5050/profile/newpage

Absolute:

`<Link to="/newpage">Click me</Link>`

Link

Current URL: localhost:5050/profile

Relative:

`<Link to="/newpage">Click me</Link>`

URL goes to



localhost:5050/profile/newpage

Absolute:

`<Link to="/newpage">Click me</Link>`

URL goes to



localhost:5050/newpage

since the URL of the root path is
localhost:5050/

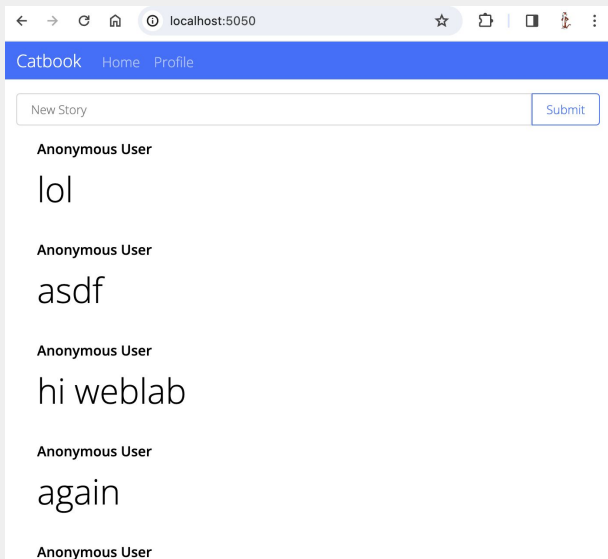
```
git reset --hard  
git checkout w3-step5
```

Follow along: step 5

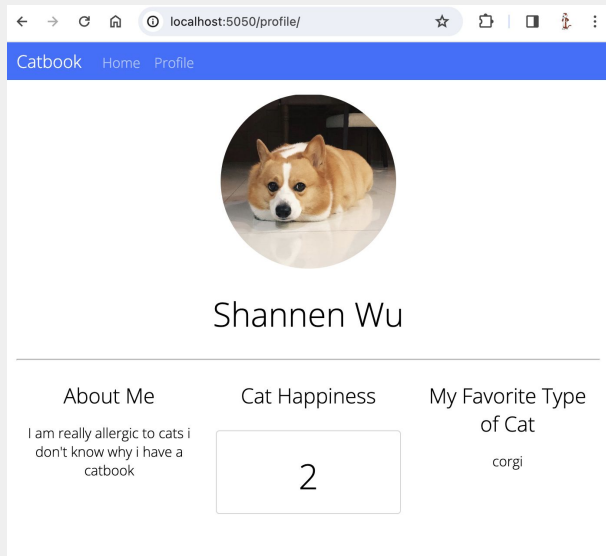
- In `App.js`
 - Use the **Router** component
- In `NavBar.js`
 - Use the **Link** component

Follow along: step 5

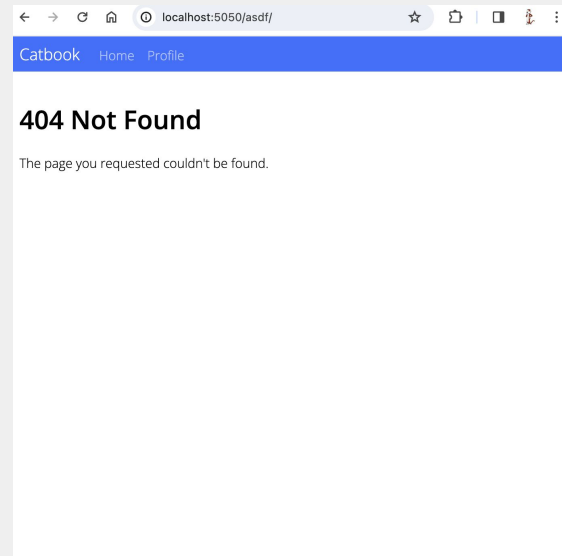
```
git reset --hard  
git checkout w3-step5
```



localhost:5050



localhost:5050/profile



localhost:5050/anythingelse

Overview of Steps 6-8

Overview of Steps 6-8

- It's time to make **comments!**

Steps 1-4: Feed.js



Steps 6-8: Card.js



Overview of Steps 6-8

Previously:

GET /api/stories

Returns all stories

Parameters: none

POST /api/story

Posts a new story

Parameters: content

Now:

GET /api/comment

Returns all comments for a story

Parameters: parent

(string id of parent story)

POST /api/comment

Posts a new comment

Parameters: parent,

(string id of parent story)

content



(Follow Along)

Step 6:

Obtaining Comments

Questions?

```
git reset --hard  
git checkout w3-step6
```

```
git reset --hard  
git checkout w3-step6
```

New Story

Anonymous User

lol

Anonymous User

asdf

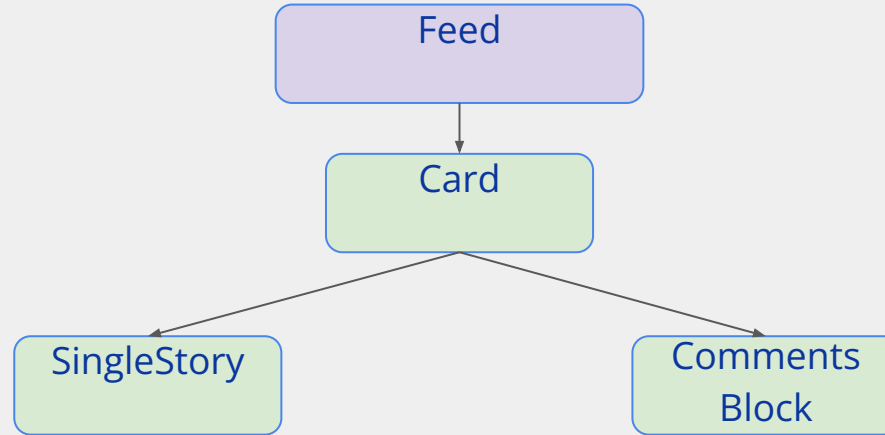
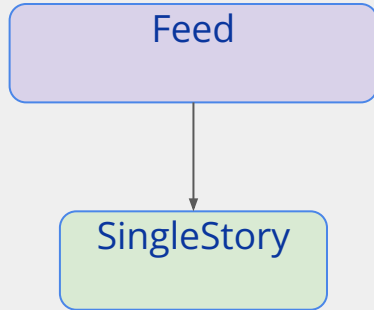


New Story

Anonymous User

lol

```
[{"_id":"6491d9791f16b00021a57674","creator_name":"Anonymous User","parent":"63bf16cd13b8200023078044","content":"dcsd","_v":0}, {"_id":"6491d9a61f16b00021a57872","creator name":"Anonymou
```




```
git reset --hard  
git checkout w3-step6
```

Follow along: Step 6

- Use `<Card (props) />` in `Feed.js`
- GET comments in `Card.js`
- Render them like stories

The screenshot shows a web application interface. At the top is a blue header bar. Below it is a form with a text input labeled 'New Story' and a 'Submit' button. Below the form is a list of stories. The first story is by 'Anonymous User' and contains the text 'lol'. Below the text is a JSON object representing the story data:

```
{ "_id": "6491d9791f16b00021a57674", "creator_name": "Anonymous User", "parent": "63bf16cd13b8200023078044", "content": "dcscd", "_v": 0 }, { "_id": "6491d9a61f16b00021a57872", "creator_name": "Anonymous User", "parent": "63bf16cd13b8200023078044", "content": "ascc", "_v": 0 }, { "_id": "64982c304430f300213fa961", "creator_name": "Anonymous User", "parent": "63bf16cd13b8200023078044", "content": "ascc", "_v": 0 }
```

(Follow Along!)

Step 7: Styling Comments

Questions?

```
git reset --hard  
git checkout w3-step7
```

Anonymous User

lol

```
[{"_id":"6491d9791f16b00021a57674","creator_name":"Anonymous
User","parent":"63bf16cd13b8200023078044","content":"dcscd","__v":0},
{"_id":"6491d9a61f16b00021a57872","creator_name":"Anonymous
User","parent":"63bf16cd13b8200023078044","content":"ascc","__v":0},
{"_id":"64982c304430f300213fa961","creator_name":"Anonymous
User","parent":"63bf16cd13b8200023078044","content":"wwwwww","__v":0},
```

```
git reset --hard
git checkout w3-step7
```



Anonymous User

lol

Anonymous User | dcscd

Anonymous User | ascc

Anonymous User | wwwwww

Anonymous User | sad

```
git reset --hard  
git checkout w3-step7
```

Follow along: step 7

- Implement `SingleComment.js`
- Use `<SingleComment (props) />` in `Card.js` (like how we used `<Card />` in `Feed.js`)

The screenshot shows a web form with a blue header bar. Below the header is a text input field labeled "New Story" with a "Submit" button to its right. Below the input field is a large text area containing the text "Hello world!". At the bottom of the text area, there is a comment by "Matt Farejowicz" saying "Nice one". The name "Aaron Sipser" is visible above the text area, likely indicating the user who created the story.

(Exercise!)

Step 8: New Comments

Questions?

```
git reset --hard  
git checkout w3-step8
```



Anonymous User

again

Anonymous User | cook

Anonymous User | 3

Anonymous User | xxxxx

Anonymous User | s1s1

Anonymous User |

Anonymous User |

Anonymous User |

Anonymous User | sfd

```
git reset --hard  
git checkout w3-step8
```



Anonymous User

again

Anonymous User | cook

Anonymous User | 3

Anonymous User | xxxxx

Anonymous User | s1s1

Anonymous User |

Anonymous User |

Anonymous User |

Anonymous User | sfd

New Comment

Submit

```
git reset --hard  
git checkout w3-step8
```

Your turn! Step 8

- Implement NewComment in NewPostInput.js
- Use `<NewComment (props) />` in Card.js

GET /api/comment

Returns all comments for a story

Parameters: parent
(string id of parent story)

The screenshot shows a web application interface. At the top is a blue header bar. Below it is a 'New Story' form with a text input field and a 'Submit' button. Below the form is a list of comments. The first comment is by 'Aaron Sipser' with the text 'Hello world!'. The second comment is by 'Matt Farejowicz' with the text 'Nice one'. At the bottom is a 'New Comment' form with a text input field and a 'Submit' button.

POST /api/comment

Posts a new comment

Parameters: parent
(string id of parent story)
content

(Follow Along)

Step 9: Cleaning up

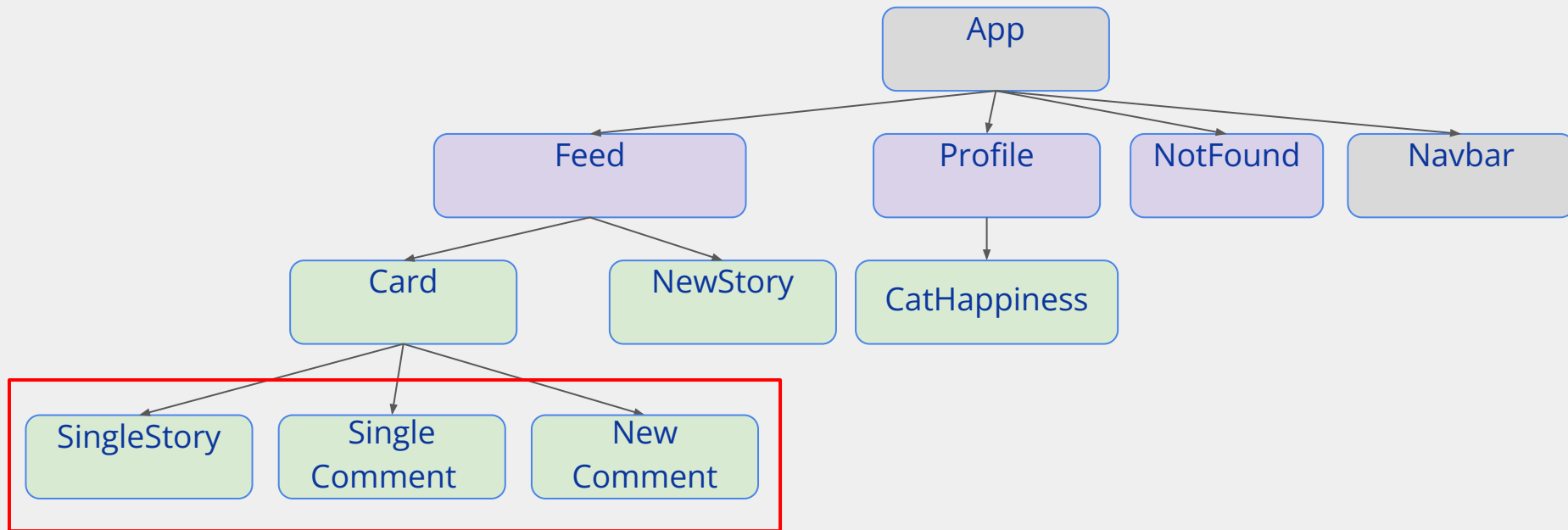
Questions?

```
git reset --hard  
git checkout w3-step9
```



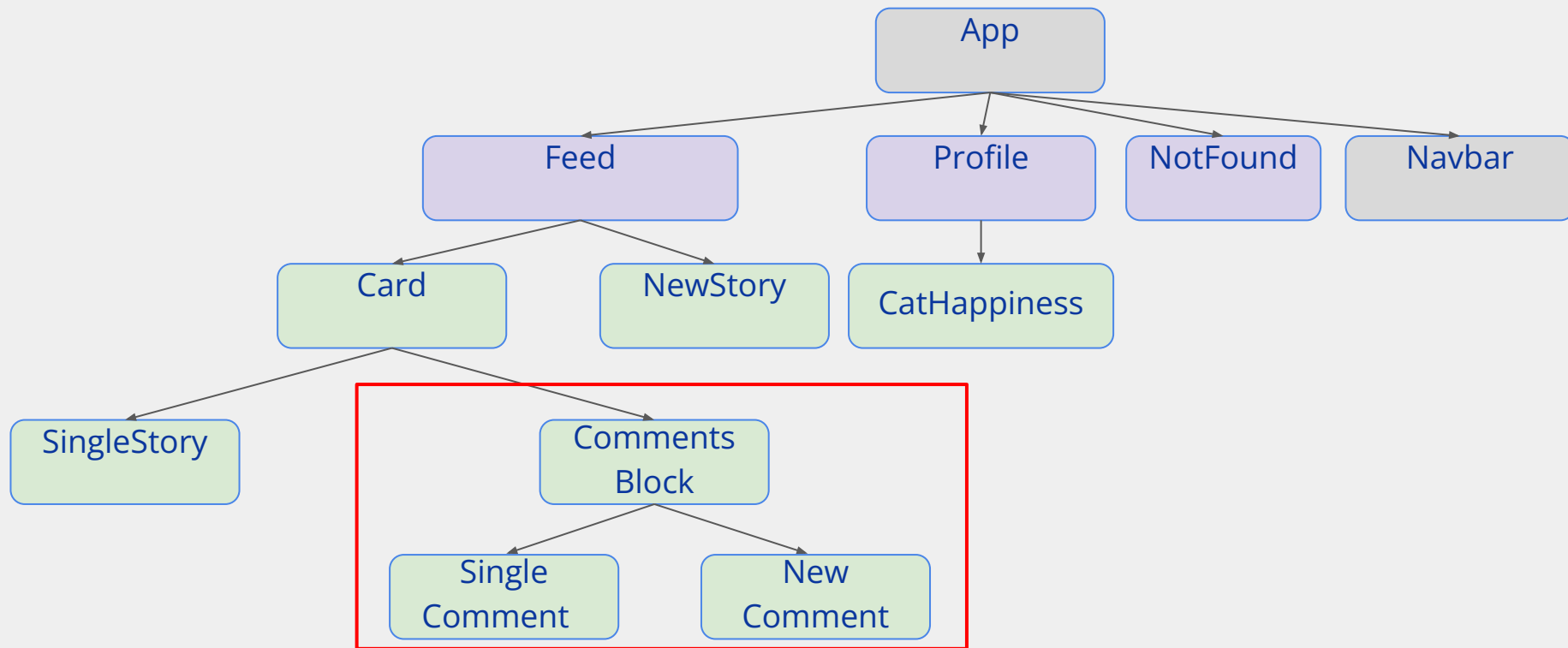
```
git reset --hard  
git checkout w3-step9
```

BEFORE: Component hierarchy



```
git reset --hard  
git checkout w3-step9
```

AFTER: Component hierarchy



```
git reset --hard  
git checkout w3-step9
```

Follow along: step 9

- Extracting out components?
 - Want to keep components small and modular
- Let's try and improve upon Card.js

```
git reset --hard  
git checkout w3-step9
```

End of step 9

Catbook [Home](#) [Profile](#)

Aaron Sipser
Hello world!

Matt Farejowicz | Nice one

(Follow Along)

Step 10:

Render on “Submit”

Questions?

```
git reset --hard  
git checkout w3-step10
```

```
git reset --hard  
git checkout w3-step10
```

Follow along: step 10

- What if we want our stories/comments to appear as soon as we click “Submit”?
 - That way we don’t need to refresh the page each time

```
git reset --hard  
git checkout w3-step10
```

Follow along: step 10

- In `Feed.js`, write the function `addNewStory` that updates the `stories` state
- Pass `addNewStory` down to `NewStory` in `NewPostInput.js`

```
git reset --hard  
git checkout w3-step10
```

Homework: finish step 10

- We've done stories—do the same for comments!
- In `Card.js`
 - 1) Define an `addNewComment` function
 - 2) Pass as prop to `CommentsBlock`
- In `CommentsBlock.js`
 - 1) Pass `addNewComment` as prop to `NewComment`
- In `NewPostInput.js`
 - 1) Use `.then` to call `addNewComment` after posting

The end!

- If you weren't able to finish live updating, complete it at home.
- If you'd just like to see the solution:
 - `git reset --hard`
 - `git checkout w3-complete`

Come back at 1:05pm :)
Remember not to eat in the lecture hall!

WALL

WALL

DOOR

Turkey

HAM

Spicy
Italian

Club

Tuna
Roast
Beef

Veg w
Chees
e

Veg
no
Chees
e

WALL

DOOR