Week 5:
Getting started with your prototype

Week 5, Part 1:

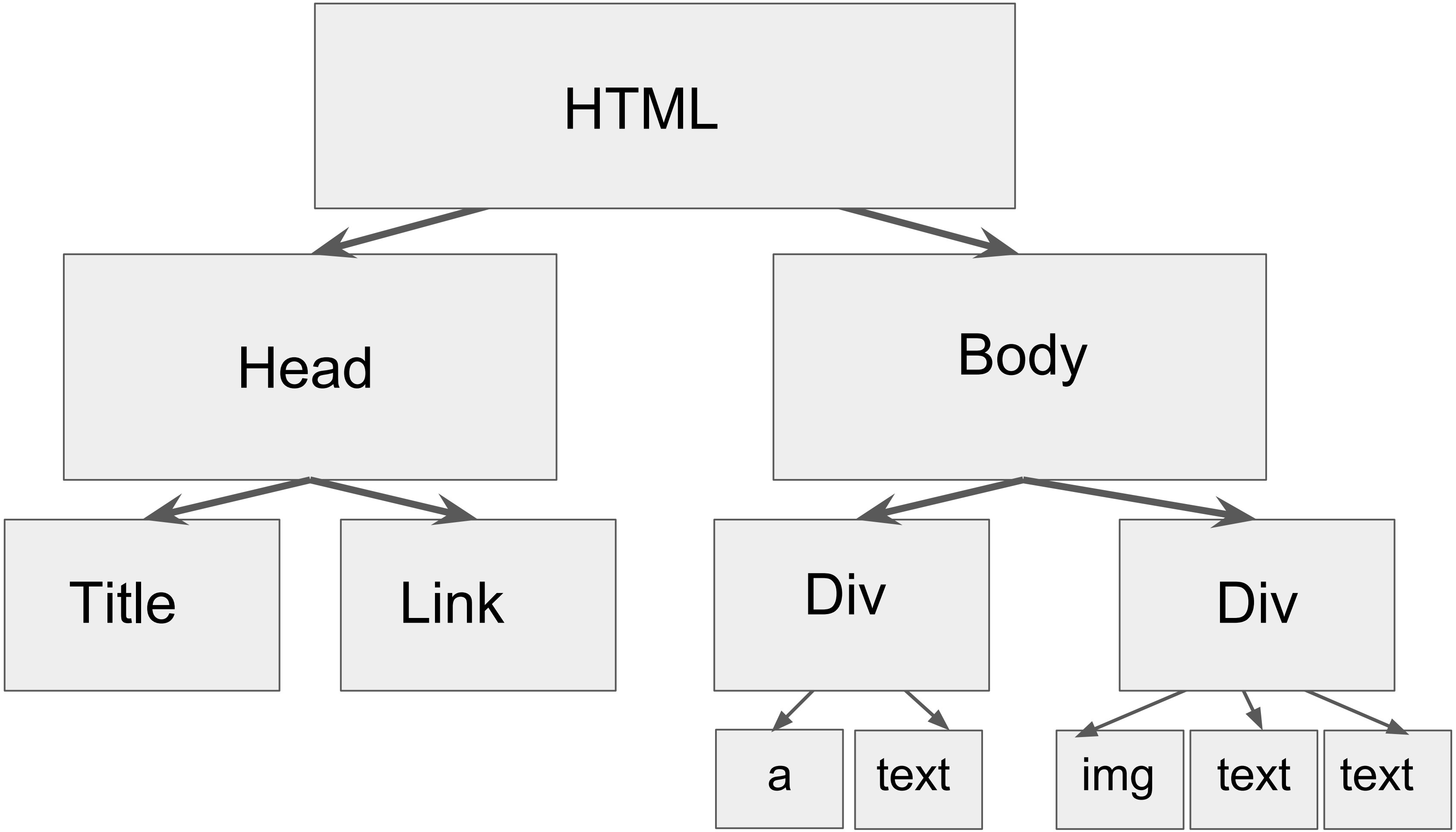
The Document Object Model and Preact

Who’s Dom?

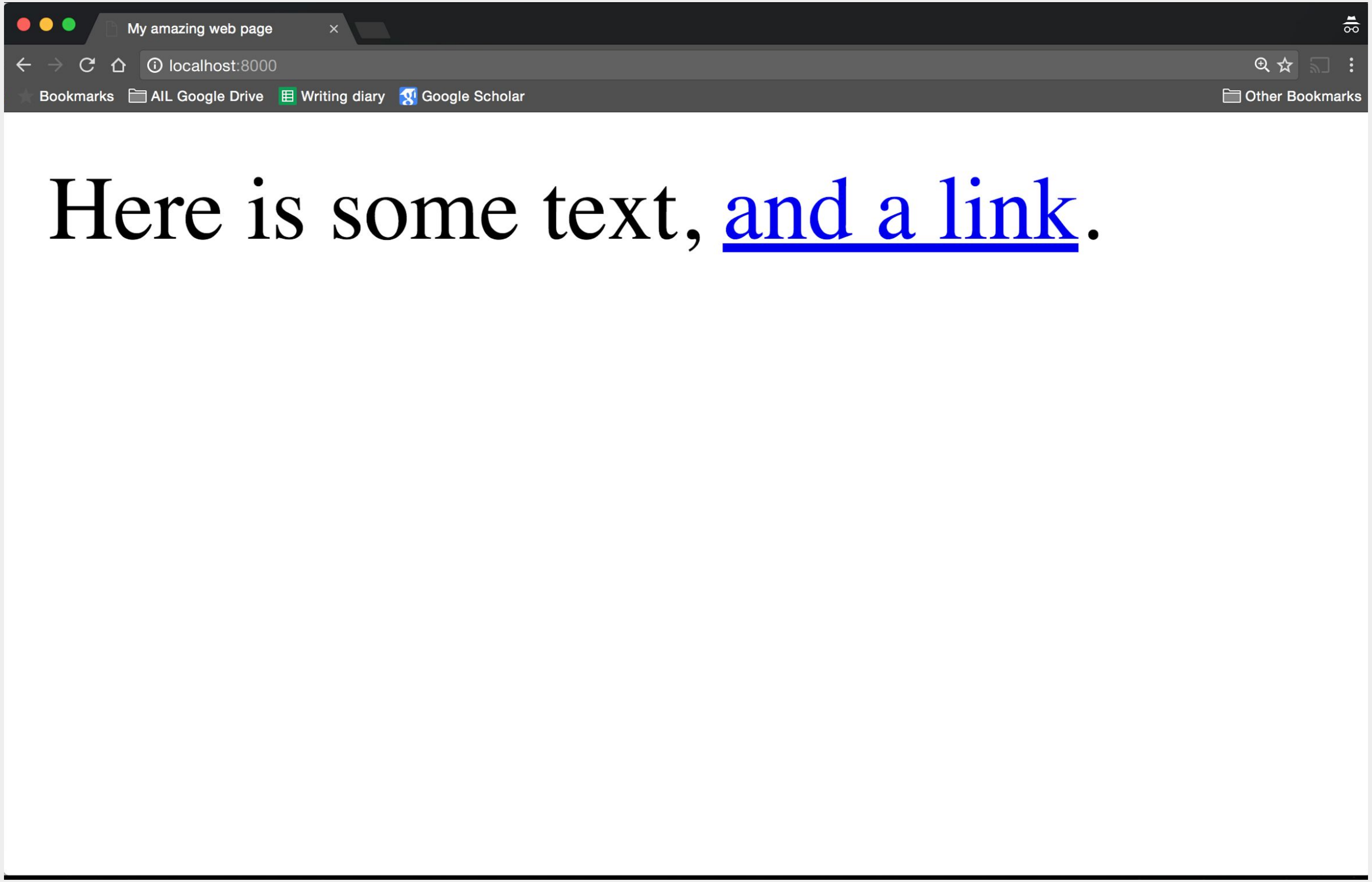
Document Object Model

What’s the document object model?

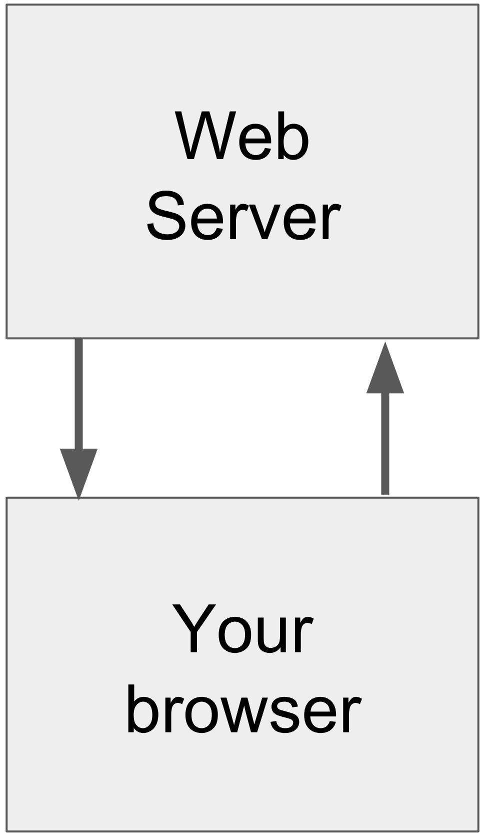
The Document Object Model (DOM) is a hierarchical description of a web page.


HTML
Head
Body
Title
Link
Div
Div
a
text
img
text
text

```
<html>
    <head>
        <title>My amazing web page</title>
        <link rel=”stylesheet” type=”text/css”
            href=”styles/style.css” />
    <body>
        <div>Here is some text, <a
                href=”http://www.google.com”>and a link</a>.
        </div>
    </body>
</html>
```



Here is some text, and a link.


Web
Server
Your
browser

The DOM describes the relationship between DOM elements.

```
<html>
    <head>
        <title>My amazing web page</title>
        <link rel=”stylesheet” type=”text/css”
            href=”styles/style.css” />
    <body>
        <div>Here is some text, <a
                href=”http://www.google.com”>and a link</a>.
        </div>
    </body>
</html>
```

Every tag denotes an element.

```
<html>
	<head>
		<title>My amazing web page</title>
		<link rel=”stylesheet” type=”text/css”
			href=”styles/style.css” />
	<body>
		<div>Here is some text, <a
				href=”http://www.google.com”>and a link</a>.
		</div>
	</body>
</html>
```

These tags are arranged in a hierarchy.

Every element is a parent and/or a child.

They are often both, but **never neither.**

Markup and styles live on the server.

They are sent to your browser when you visit a web page, and your browser displays them.

The process of displaying a web page is called **rendering**.

Rendering:

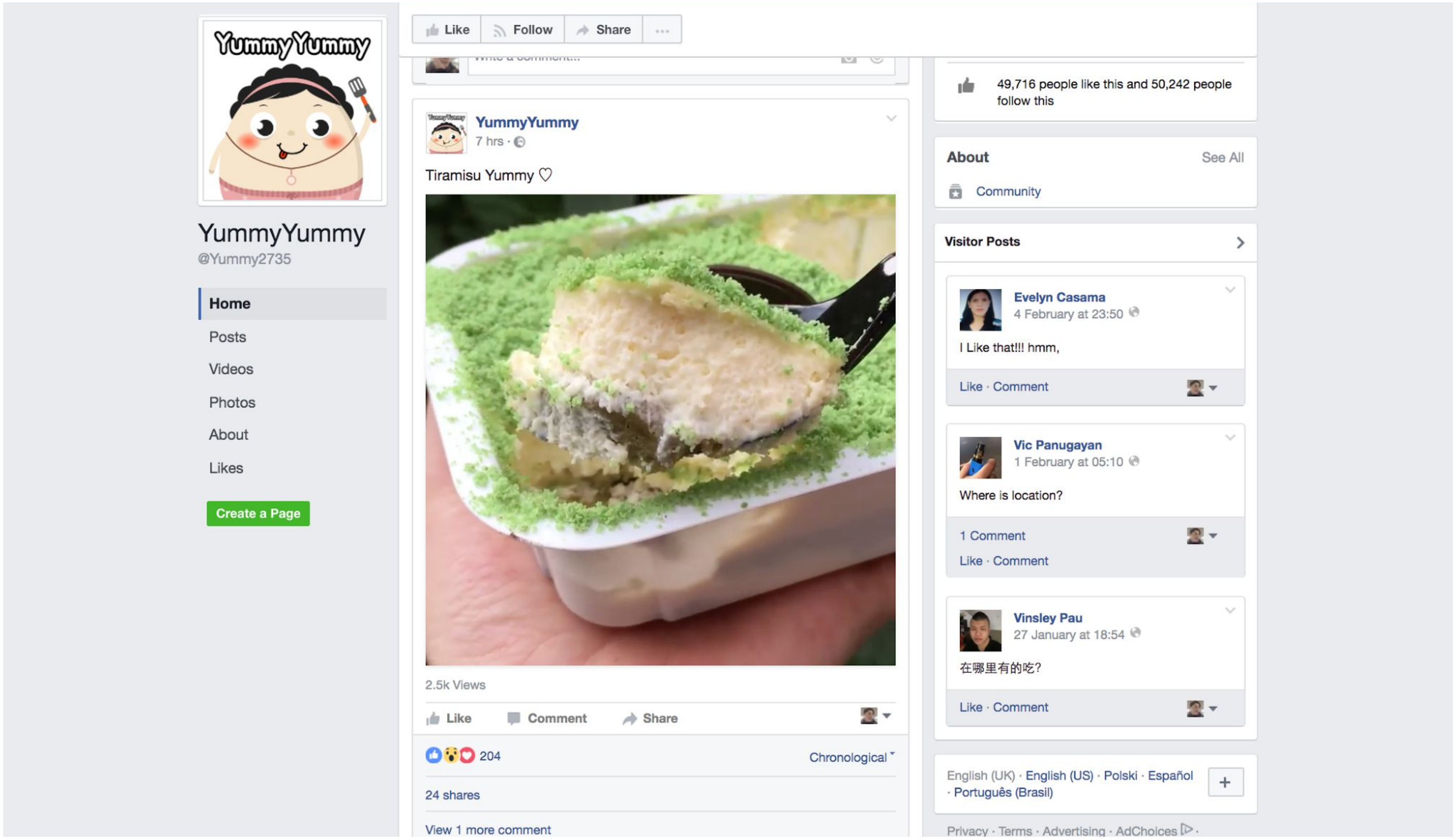
The process of displaying the visual representation of the DOM in a browser.

“So what’s the big deal?”

The web is getting complicated and rendering is inefficient.

Where's the complication?

- Dynamic content
- Constant updates
- Feeds
- Sharing
- Star ratings
- Other stuff ...



YummyYummy
@Yummy2735

Home
Posts
Videos
Photos
About
Likes

Create a Page

Like · Follow · Share · ...

YummyYummy
7 hrs ·

Tiramisu Yummy ♡

2.5k Views

Like · Comment · Share

204 · Chronological

24 shares

View 1 more comment

49,716 people like this and 50,242 people follow this

About · See All
Community

Visitor Posts

Evelyn Casama
4 February at 23:50
I Like that!!! hmm,
Like · Comment

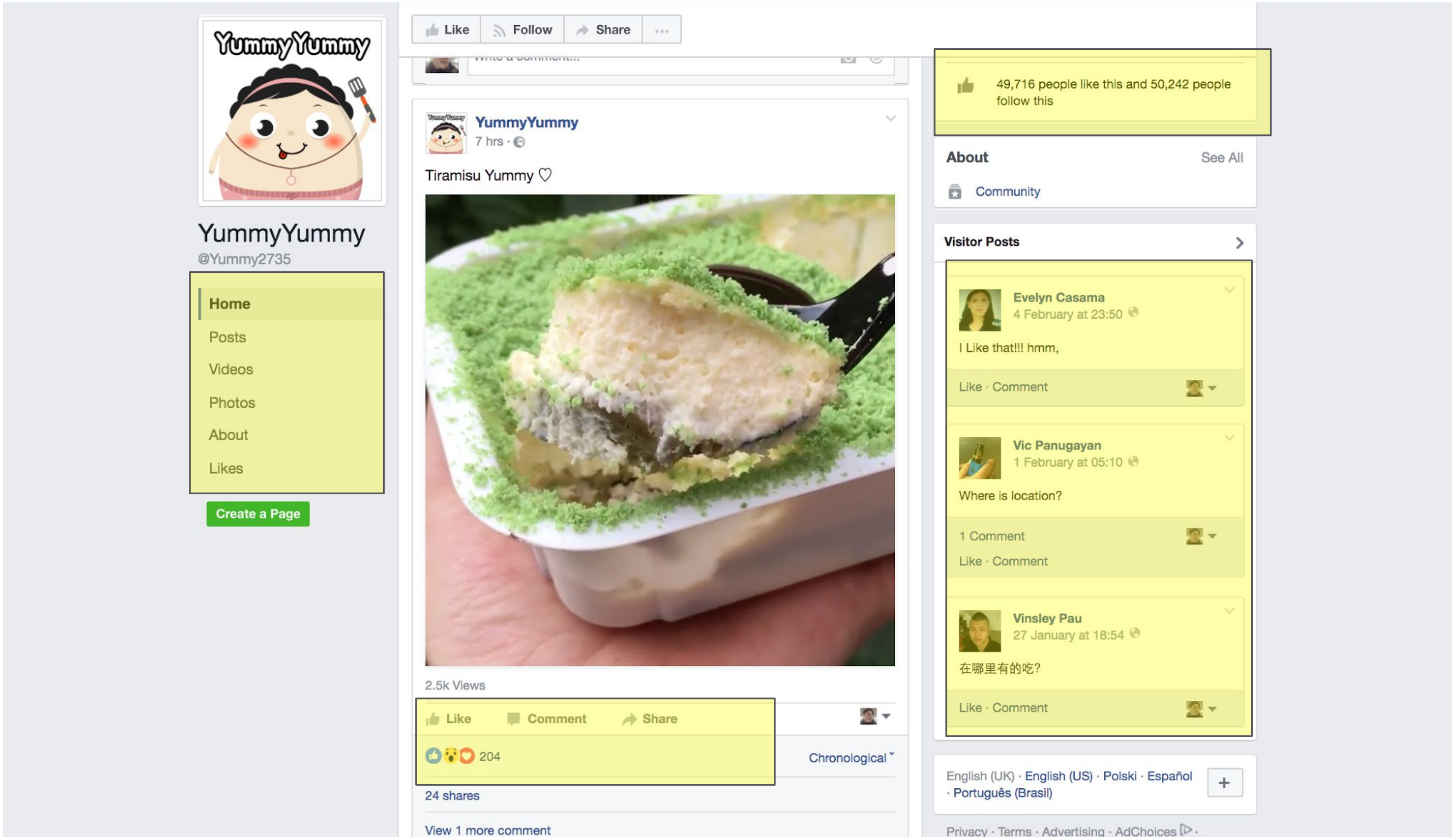
Vic Panugayan
1 February at 05:10
Where is location?
1 Comment
Like · Comment

Vinsley Pau
27 January at 18:54
在哪里有的吃?
Like · Comment

English (UK) · English (US) · Polski · Español · Português (Brasil)

Privacy · Terms · Advertising · AdChoices ·

Lots of data is coming in at once.


Like · Follow · Share · ...

YummyYummy
@Yummy2735

Home
Posts
Videos
Photos
About
Likes

Create a Page

YummyYummy
7 hrs ·

Tiramisu Yummy ♡

2.5k Views

Like Comment Share

204

Chronological

24 shares

View 1 more comment

49,716 people like this and 50,242 people follow this

About See All

Community

Visitor Posts

Evelyn Casama
4 February at 23:50

I Like that!!! hmm,

Like · Comment

Vic Panugayan
1 February at 05:10

Where is location?

1 Comment

Like · Comment

Vinsley Pau
27 January at 18:54

在哪里有的吃?

Like · Comment

English (UK) · English (US) · Polski · Español · Português (Brasil)

Privacy · Terms · Advertising · AdChoices ·

To change the value of one element, a web page used to have to entirely re-render.

(omg so slow)

This is due mainly to browser limitations:

- Browsers don’t really have memory
- Browsers have to re-render to reflect changes to the DOM

React

React is a **JavaScript library** for handling the virtual DOM and rendering components of a web page.

React was developed by Facebook.

React

(Thanks, Facebook.)

Hang on ... JavaScript?

- The **browser language** that has been gaining a lot of traction in the past ~6 years (after being pretty universally hated since it was invented in 1995).
- It has nothing to do with Java (but they're both related to C).
- **It can manipulate DOM elements.**

Why would I want to do things with the DOM?

- Show elements
- Hide elements
- Create elements
- Apply styles to elements
- Put data inside elements
- Listen for events
- Trigger responses to events
- Position elements

React uses JS to manipulate the DOM in dynamic and useful ways.

* But, like all programming languages, it has its own weirdness.

React (as an approach to web building) **has two big advantages:**

1. React gives us a **virtual DOM.**

What’s a virtual DOM?

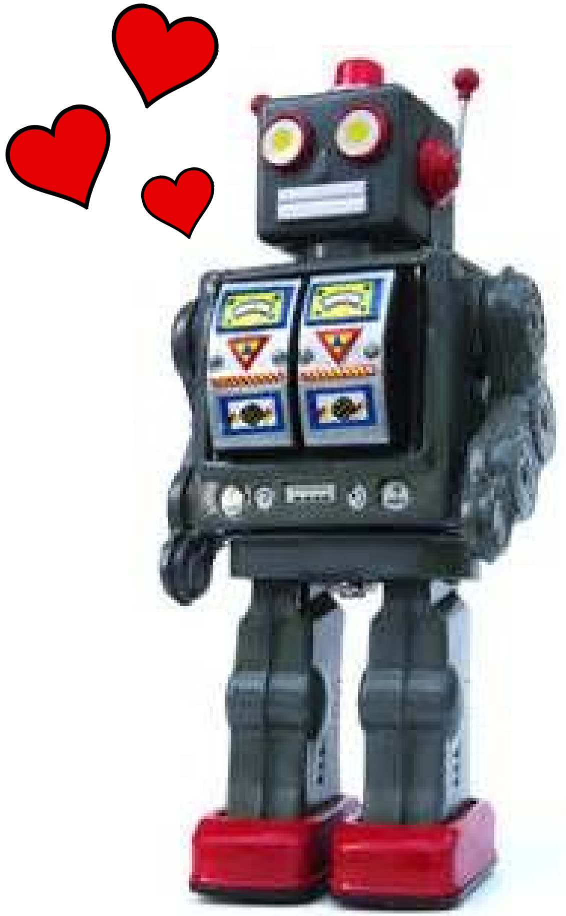
Browsers don’t really have a good way of remembering what their previous state was when they receive new information.

What’s a virtual DOM?

React’s **virtual DOM** acts like a robot, that remembers the previous state of the DOM, and when it receives new information **is able to extract just the differences, and only render those differences.**

Since we then only need to render little bits that have changed, this makes things a lot faster and much easier.

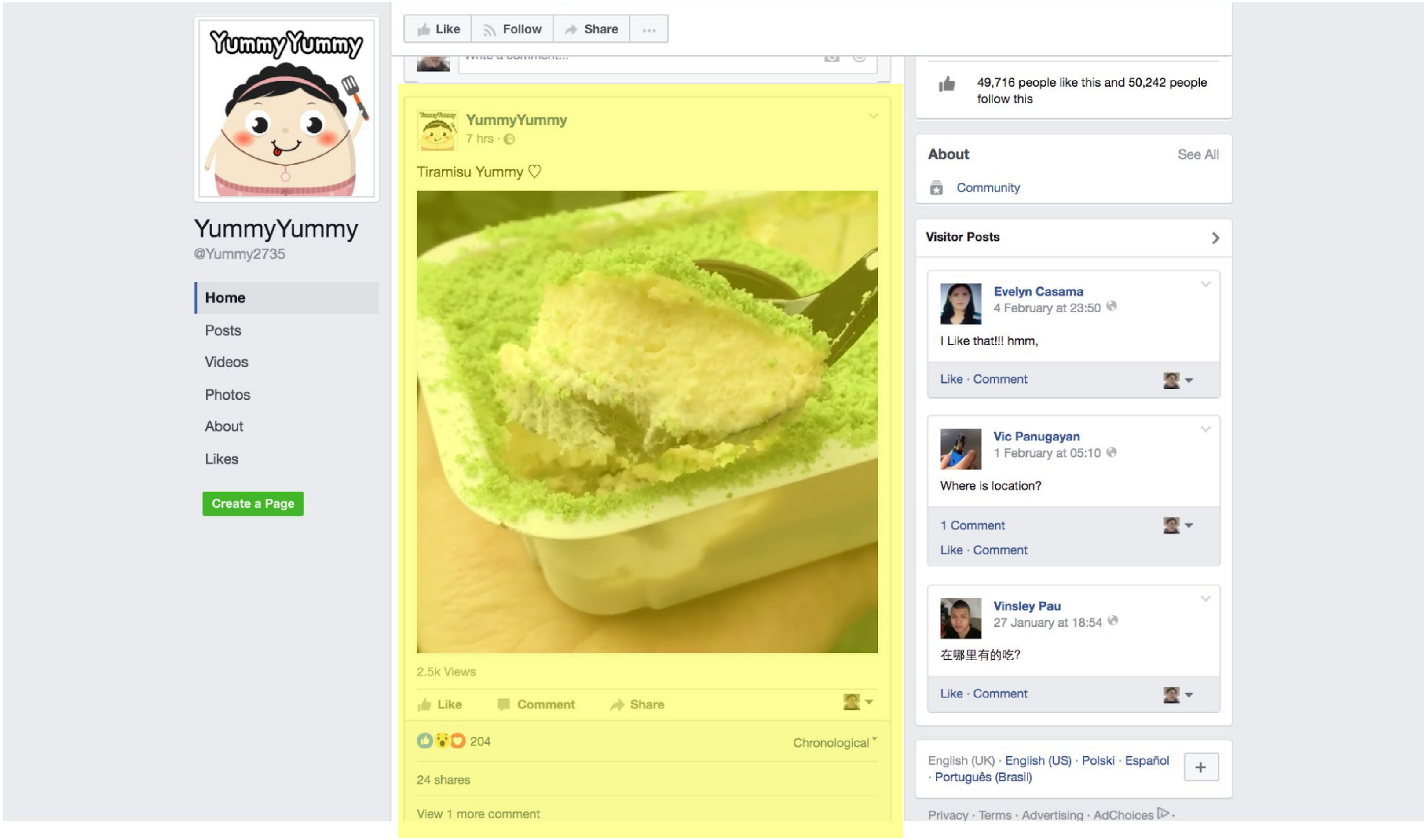
(Good robot.)



2. React uses **components.**

Let’s consider a Facebook feed.

- A feed is made up of lots of posts.
- Each post has associated data such as likes, comments, a number of shares.
- We can interact with this data in a number of ways.



YummyYummy

@Yummy2735

Home

Posts

Videos

Photos

About

Likes

Create a Page

Like Follow Share ...

YummyYummy

7 hrs ·

Tiramisu Yummy ♡

2.5k Views

Like Comment Share

204 Chronological

24 shares

View 1 more comment

49,716 people like this and 50,242 people follow this

About See All

Community

Visitor Posts

Evelyn Casama

4 February at 23:50

I Like that!!! hmm,

Like · Comment

Vic Panugayan

1 February at 05:10

Where is location?

1 Comment

Like · Comment

Vinsley Pau

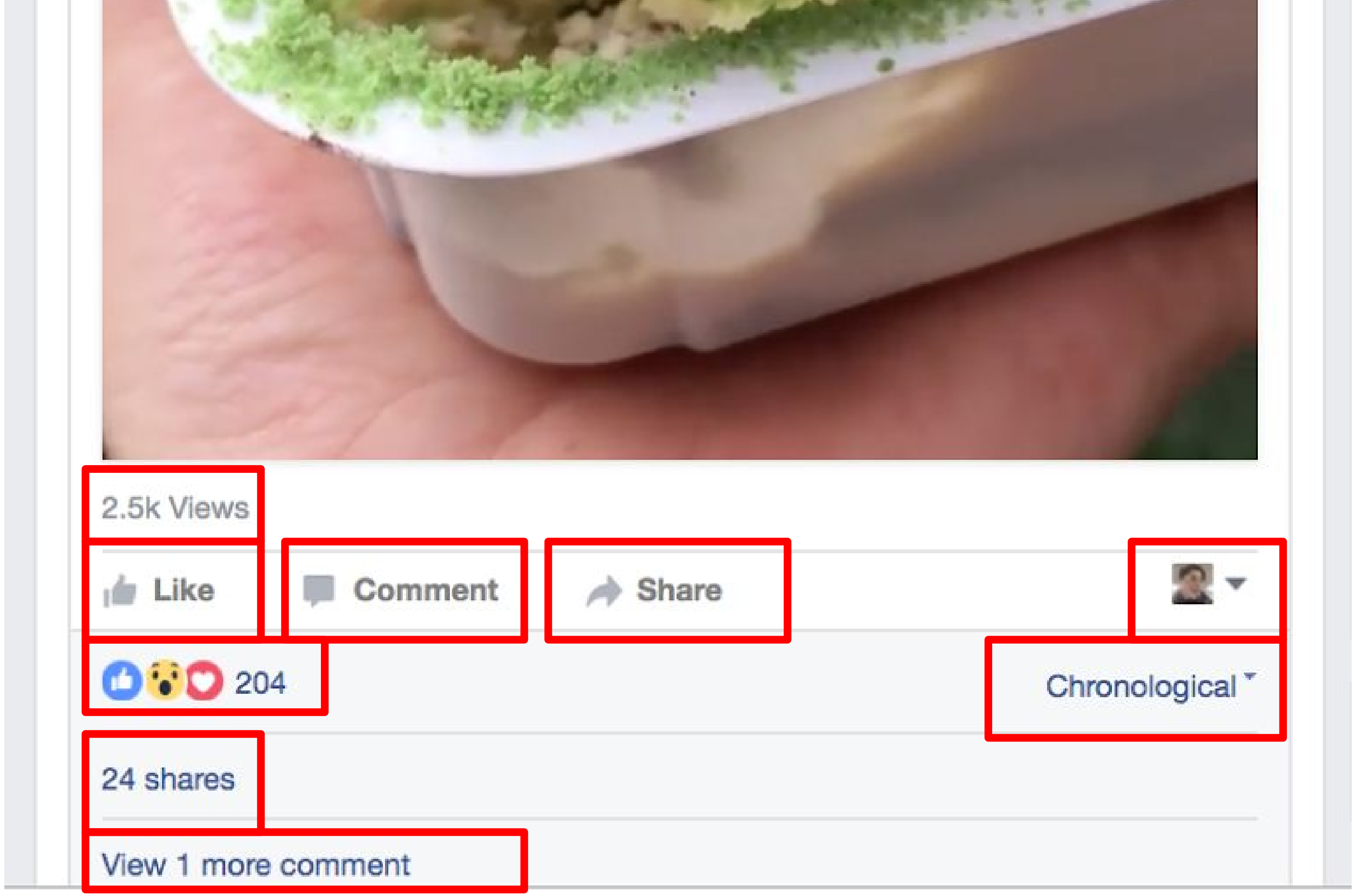
27 January at 18:54

在哪里有的吃?

Like · Comment

English (UK) · English (US) · Polski · Español · Português (Brasil)

Privacy · Terms · Advertising · AdChoices ·


2.5k Views

Like

Comment

Share

204

Chronological

24 shares

View 1 more comment

Components can:

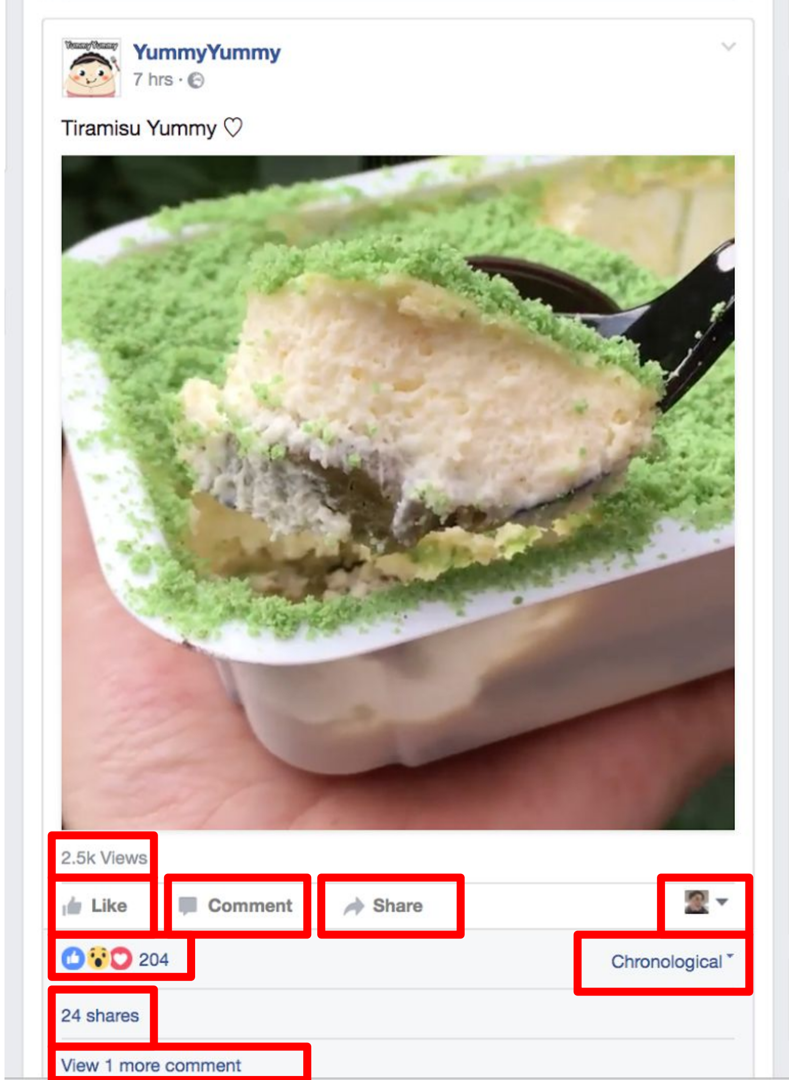
Display data

Collect/send data

Trigger events

Be nested inside other components

(... and lots of other things)



BBC

UBER

okcupid

KHAN
ACADEMY

airbnb

Hang on … aren’t we using Preact?

Yes.

What is Preact?

It’s a really small implementation of React (3k vs 150k), because React is enormous.

It has slightly less functionality, but plenty for our purposes.

Why Preact?

It’s much simpler to use, but you will still get to learn how React does things, and become familiar with what it means to build React applications.

This sounds hard. I don’t even know JavaScript.

Let’s get started!

Go to:

bit.do/guiBoilerplate

Week 5, Part 2:

A look inside the boilerplate

boilerplate (n):
a standard template
that can be reused
without greatly
changing the original.

After Node/NPM are installed, go into the directory on the command line and run

```
npm install
```

to set up the environment.

After all the modules are installed, start a live reload development server (this means as you make changes they’ll be reflected in the app in your browser):

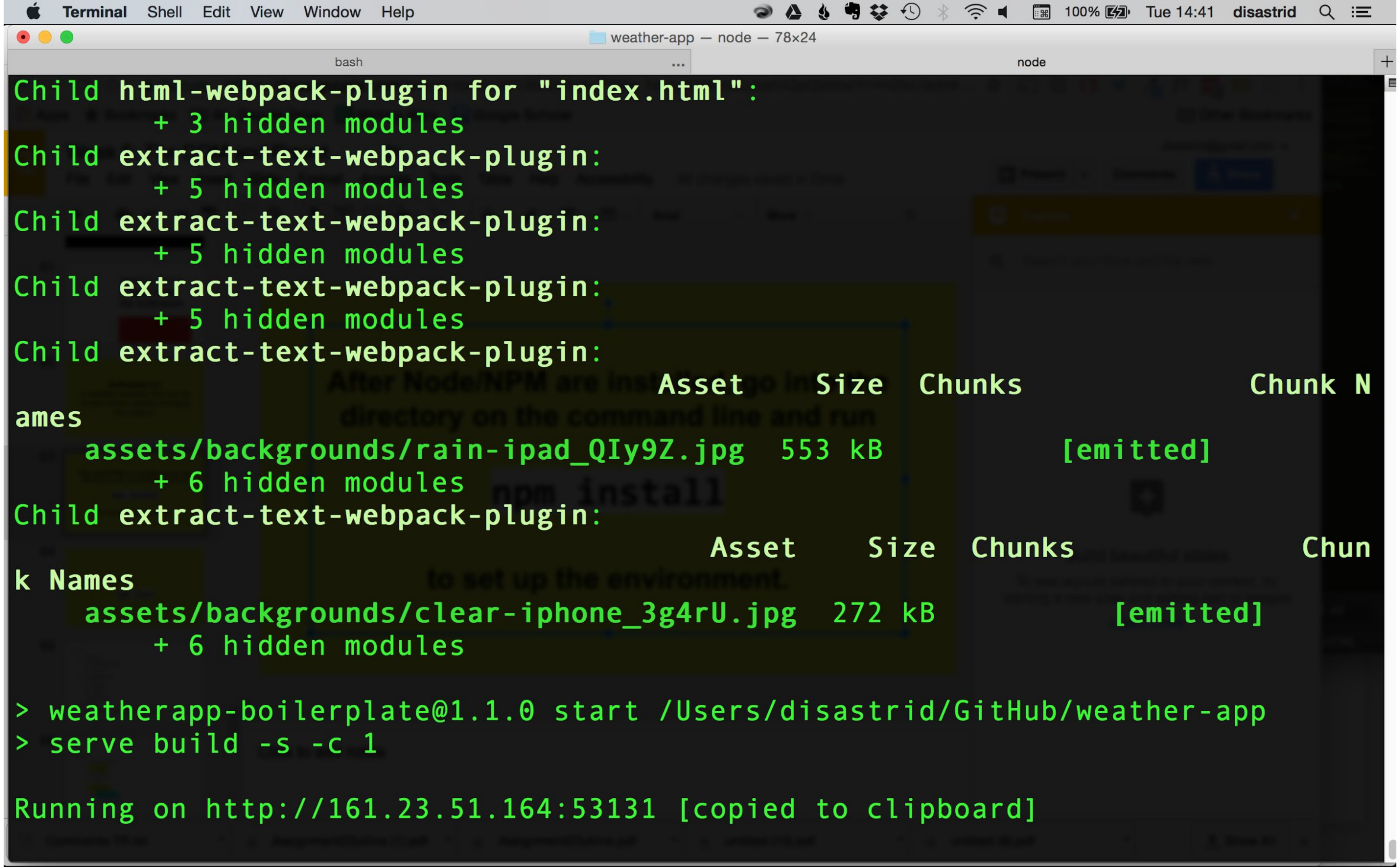
```
npm run dev
```

Generate a production build in ./build:

```
npm run build
```

To start the app, type:

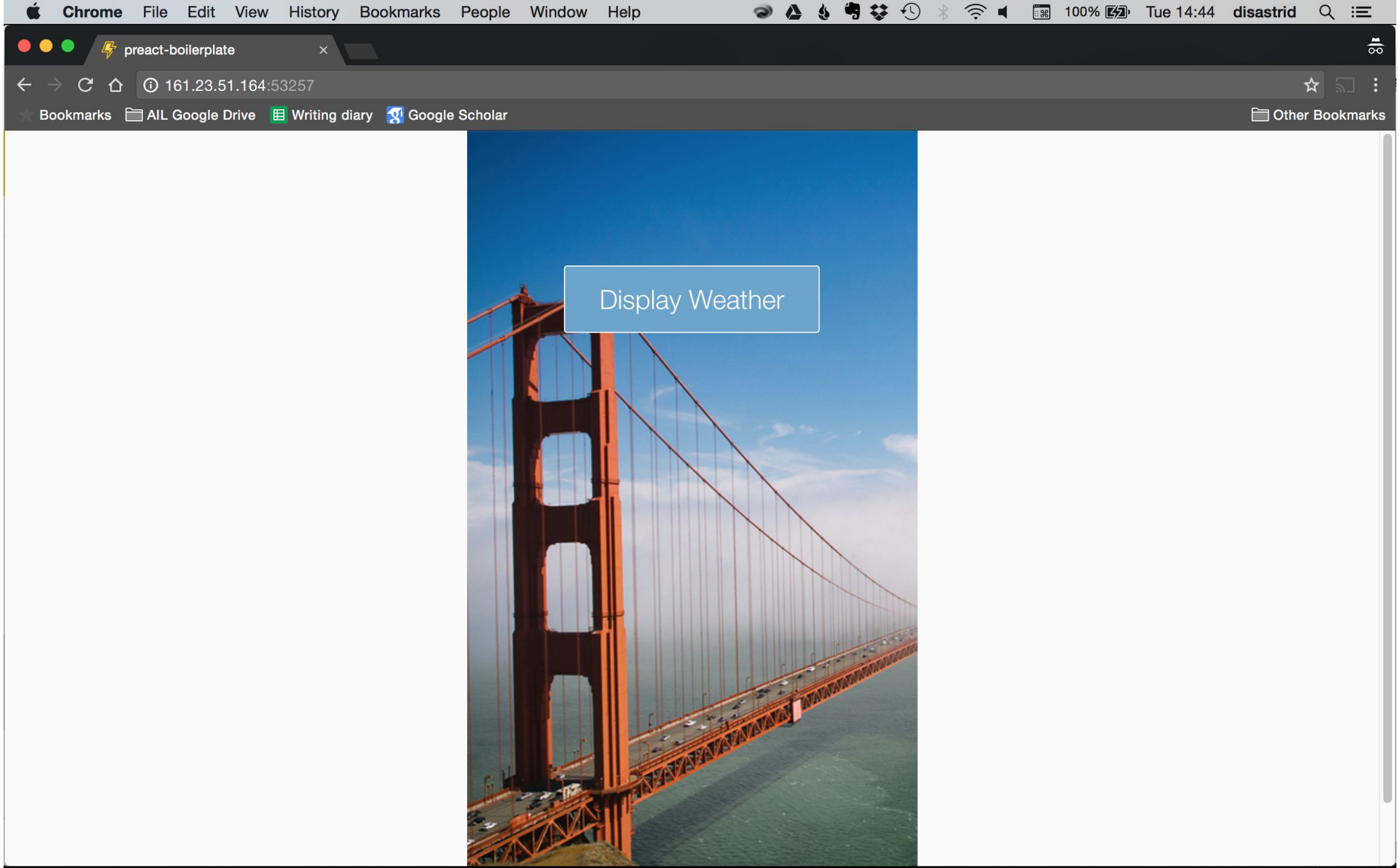
```
npm start
```

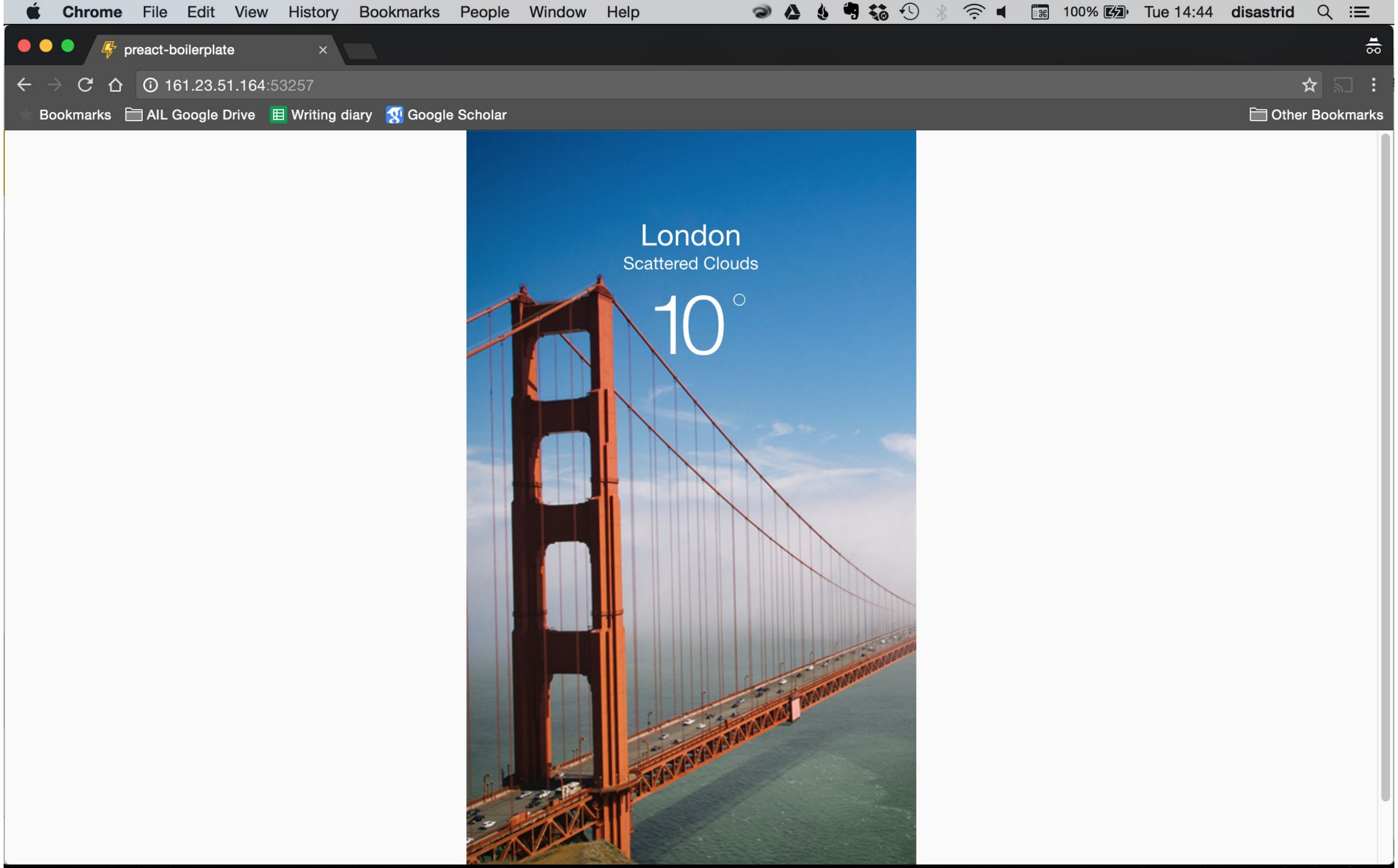


```
Child html-webpack-plugin for "index.html":
        + 3 hidden modules
Child extract-text-webpack-plugin:
        + 5 hidden modules
Child extract-text-webpack-plugin:
        + 5 hidden modules
Child extract-text-webpack-plugin:
        + 5 hidden modules
Child extract-text-webpack-plugin:
                                     Asset    Size  Chunks             Chunk N
ames
    assets/backgrounds/rain-ipad_QIy9Z.jpg  553 kB          [emitted]
        + 6 hidden modules
Child extract-text-webpack-plugin:
                                        Asset    Size  Chunks             Chun
k Names
    assets/backgrounds/clear-iphone_3g4rU.jpg  272 kB          [emitted]
        + 6 hidden modules

> weatherapp-boilerplate@1.1.0 start /Users/disastrid/GitHub/weather-app
> serve build -s -c 1

Running on http://161.23.51.164:53131 [copied to clipboard]
```

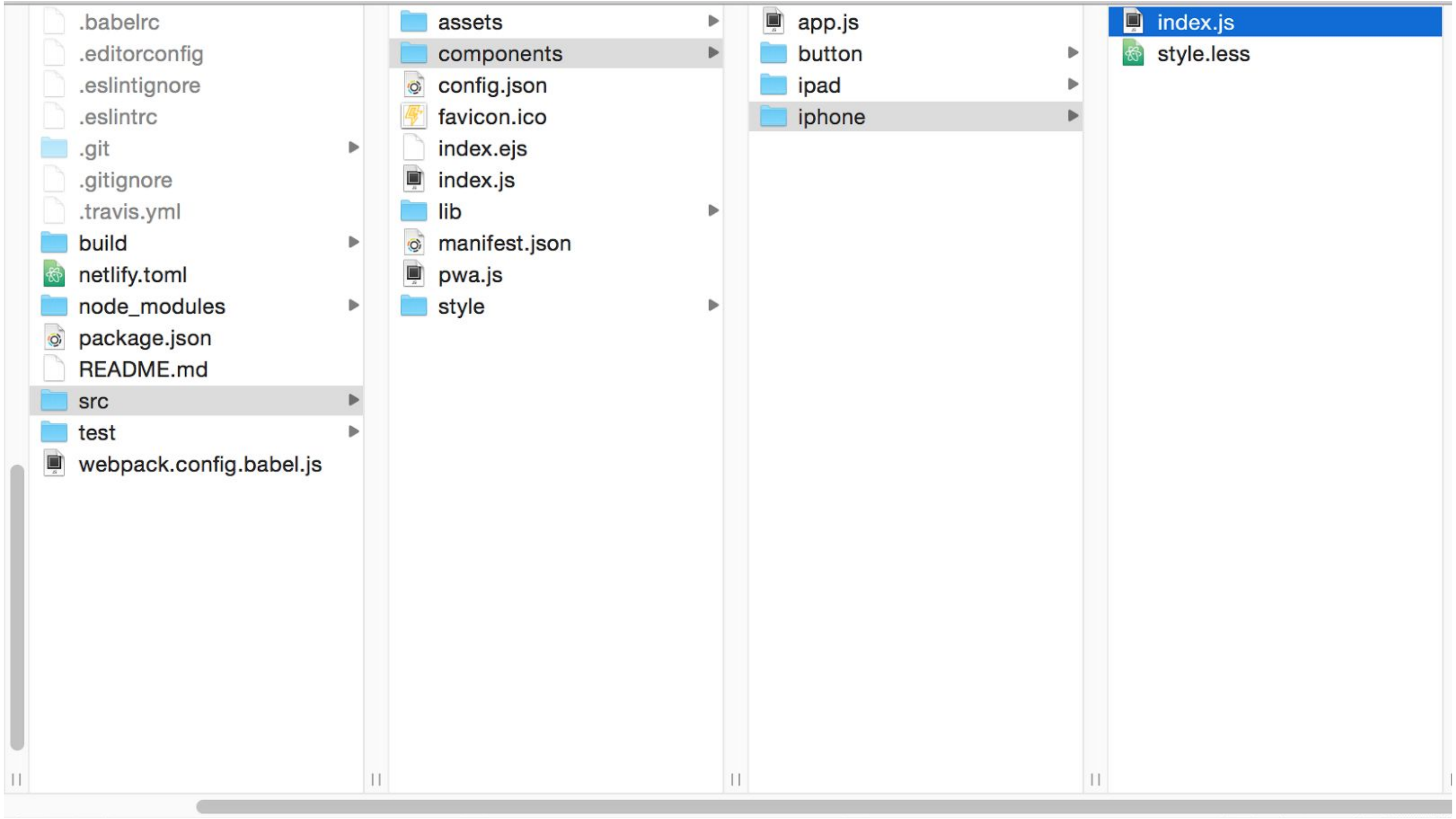

Chrome
File
Edit
View
History
Bookmarks
People
Window
Help
100%
Tue 14:44
disastrid
preact-boilerplate
161.23.51.164:53257
Bookmarks
AIL Google Drive
Writing diary
Google Scholar
Other Bookmarks
Display Weather



London

Scattered Clouds

10°



.babelrc
.editorconfig
.eslintignore
.eslintrc
.git
.gitignore
.travis.yml
build
netlify.toml
node_modules
package.json
README.md
src
test
webpack.config.babel.js

assets
components
config.json
favicon.ico
index.ejs
index.js
lib
manifest.json
pwa.js
style

app.js
button
ipad
iphone

index.js
style.less

Index.js contains the meat of our boilerplate.

It imports things we need: Preact render function, button component, styles, jQuery.

It has some functionality:

- Runs the constructor (which sets up our start conditions)
- Runs the **render function**
- Check if the API should be asked for data (it does so when the button is clicked)

The constructor

Runs once on startup to set up the default state.

```
    this.state.temp = "";
    this.setState({ display: true });
```

So we have an empty temperature string, and we’ve set the display state to true.

The render() function

This tells Preact what to display.

There’s lots of variables in here - that way it’s easy to pass data or events into this function, and Preact’s virtual DOM will notice changes and only re-render those bits.

Little pieces of HTML generated by JS in functions like this are called **partials.**

The Button component

This is imported at the top of the file. The functionality is in a js file: components > button > index.js

We imported it as Button, and we can access the render file inside that index.js file through a tag <Button />. (Kind of like using a Java object!)

Consider this code:

```
<div class= { style_iphone.container }>

   { this.state.display ? <Button class={
style_iphone.button } clickFunction={
this.fetchWeatherData }/ > : null }

</div>
```

This is an if-else statement with ? :

```
this.state.display ? <Button class={
style_iphone.button } clickFunction={
this.fetchWeatherData }/ > : null
```

Check if `this.state.display` is true (at first, it is)

If it is, display the `Button` tag.

If not, don’t display anything.

Consider the function that runs if we click the button:

```
<Button class={ style_iphone.button }
clickFunction={ this.fetchWeatherData }/ >
```

`fetchWeatherData()` is defined in the main `index.js` file! It asks the API for weather data, and parses the response. Then, it hides the button to display data.

A note on API keys

You’ll have to register for your own at Weather Underground (there is a limit per day so we can’t all use the key in the boilerplate!)

More info in the Resources Google doc.

This still seems hard.

(╯°□°)╯︵ ┻━┻

Maybe, but it’s worth it and you can do it.

┬──┬ ノ(゜-゜ノ)

Breathe in

Tip 1: Plan to iterate.

What’s the minimum viable product (MVP)?

What can we add on once we have that done?

What else? What’s after that?

Tip 2: Pseudo code before anything else.

Write out in plain English what your app needs to do, for each iteration.

Actual coding should always come after good pseudo coding!

Tip 3: Leave the styling and design until after functionality is done.

Get your functionality done without worrying about how it looks. THEN concentrate on what it looks like and how that design serves the stakeholder - doing both at once is unproductive.

Tip 4: Google Developer Tools are useful.

In Chrome, hit `alt + cmd + I` to open the developer tools.

You’ll be able to see your HTML and styles and try things out. This is a huge time saver.

Assignment 2: Implementation.

30% Design

30% Implementation

30% Extension

10% Participation

30%: Implementation

Criteria:

- Functionality and code quality
- Understanding of components
- Commenting of code
- Ability to explain your code
- Crediting

30%: Design

Criteria:

- Clarity
- Aesthetics
- Rationale

30%: Extension

Criteria:

- Ambition
- Originality
- Fit to stakeholder

10%: Participation

This is decided as a group.

Criteria:

- Participation
- Communication
- Effort

Breathe in