**Tuesday May 14th, 2019 Daily Coding Journal**

8:44 — I’m going to spend this morning pomodoro session learning about markdown.

8:45 — The following notes on markdown are from this video:

<https://www.youtube.com/watch?v=HUBNt18RFbo>

8:47 — Markdown is commonly used for readme files (such as those on Github), as well as by many forums and blog posts.

8:48 — The file extension used by markdown is .md

8:49 — Markdown is used by many static site generators.

8:50 — Here are some of the ways we can format using markdown:

Lists

Emphasis/Italics

Links

Blocks of code

Block quotes

Images

Horizontal rules

8:50 — There are some various “flavors” of markdown such as PHP markdown and a GitHub variation of markdown that may “extend” markdown or add additional features.

8:51 — The author recommends using VSCode with an extension called auto open markdown previewer. Though there are many other tools one could use such as MarkPad, HarooPad, MarkdownPad2, Typora, etc.

8:54 — The author summarizes markdown as being basically a very lightweight markup language.

8:55 — Now, we’re going to get into some of the basic syntax of Markdown.

8:56 — We can use the hashtag symbol to create headings. One hashtag symbol will create h1 text, two hashtags will create h2 text, three hashtags will create h3 text, and so on up until a whopping six hashtags gives us h6 text.

# This is h1 text.

## Here’s some h2 text.

### This text would display as h3 text

#### h4 text here

##### Here’s some h5 text.

###### Bring on the h6 text.

9:00 — To italicize text we can surround it using single asterisk symbols or single plain underscores.

\*Italicize me\*

\_I am italicized\_

9:01 — To make text bold or “strong” we surround it using double asterisks or double plain underscores.

\*\*I am bold\*\*

\_\_I am strong\_\_

9:02 — For strikethroughs we surround our text with doubles tildes

~~I am a mistake~~

9:04 — For a horizontal rule (which is basically something we can use as a separator between two pieces of content), the syntax is three hyphens or three underscores.

Content #1

---

Content #2

\_\_\_

Content #3

9:06 — Another important thing to note is that if we want to escape any special characters, we can do so with a back slash (in the same manner we may do so in JavaScript RegExp)

10:45 — Let’s continue taking notes. For block quotes we use the greater than symbol.

> This line will display as a block quote.

10:47 — For links in markdown we use brackets and parentheses. The anchor text goes in brackets and the address goes in parentheses. If you’d like a title to be displayed when users hover over your link, you can also include that within quotations within the parentheses. Example…

Please be sure to read the [following article]

(<http://cameronchardukian.com/nexstand-k2-review/> “Your title here”)

10:53 — Using markdown we can also create unordered lists. To do this, we use a single asterisk and list a single item on each line.

\* Apple

\* Orange

\* Pear

It’s also possible to create nested lists. For example, we could write things that you should not consume before your physical exam:

\* Starchy foods

\* Calorically dense food

\* Supplements such as:

\* PEDs

\* Mega-doses of vitamins

\* etc.

10:58 — Markdown also makes it possible for us to create ordered lists. We do this by writing the number one followed by a dot. An important thing to note is that we’re literally writing the number one every single time. We don’t have to write one, two, three as we may have to in some other word processing applications.

Markdown will increment our ordered list items for us under the hood. Example syntax:

The most important players on an American football team are the following:

1. Quarterback

1. Designated pass rusher

1. A lockdown cornerback

11:07 — Using markdown we can also create inline code blocks. We do that using backticks.``

`<p> While in this example I used p tags, you could also have used the backticks with any other example of code. <p/>`

11:10 — Let’s talk about images. To display images using markdown we use syntax that is very similar to links. We put our anchor text in square brackets, our url or path to the image in parentheses, and an optional title attribute in single or double quotes. Let’s look at a practical example using the above syntax:

[Curry](<http://cameronchardukian.com/wp-content/uploads/2018/11/IMG_6749-1024x768.jpg> “Maya Mall Curry”)

11:14 — I’m going to take a quick break to eat breakfast/brunch before the restaurant closes. When I come back I’ll focus on some of markdown’s syntax as used with GitHub.

14:43 — I’ve finished lunch and took a nap amongst other obligations. Now I’m going to look at some Github Markdown syntax for the next 15 minutes.

14:45 — On GitHub we can indicate that we’re going to display code by using triple backticks.

```

npm install

```

Another cool thing we can do is indicate which programming language the code is written in. For example, if we write javascript after the three opening backticks the code would change colors based on syntax specific to the JavaScript language:

```javascript

function fullName(first, last) {

let myFullName = `${first} ${last}`;

```

We could also do the same with python or other languages.

14:55 — GitHub also allows us to do tables. To do that we can use the vertical bar or ‘pipe’ character on our keyboards.

| Name | Phone |

| ------------------ | ------------------------ |

| Cameron | 123-1234-1234 |

While the above may have displayed or look not aligned 100% correctly, that’s because I’m writing this in a sticky note application. Starting tomorrow I’ll start writing all of my daily journal entries in markdown now that I’ve been exposed to all it has.

14:59 — I’ve got to go now. We’ll finish the video in the next pomodoro session I think! :)

16:42 — I’m back. Let’s finish that video up now.

16:44 — Another thing we can do using Github’s version of markdown is create task lists. Perhaps we want to show tasks that are completed with checks, and tasks that are not yet completed without checks. We can do that by using square brackets with an X nested inside the brackets if the task has already been completed and plain square brackets if the task is yet to be completed. For example…

[X] Task #1 is completed

[X] Task #2 is completed

[ ] Task #3 is not yet completed.

16:54 — Well, I’ve finished the entire video now! Learning the basics of markdown really only takes 1-1.5 hours. Getting comfortable with all of the syntax probably takes another hour or two of practice and I’m confident that within a week of using the syntax you’ll probably be able to easily commit to memory most of the important things.

16:56 — Now that we’ve gotten markdown out of the way, let’s get back to the Scrimba *Learn React for Free* course!

17:11 — I’ve gotten to what feels like the final boss in an intensely difficult game. Trying to actually figure out how to create a function with a const, .map( ) method, setState, amongst other things.

17:17 — The following code logs out the todos of the previous state just as one would expect.

handleChange(id) {

this.setState(prevState => {

console.log(prevState.todos)

})

}

17:18 — For whatever reason, however, I can’t seem to use dot notation any further, however.

handleChange(id) {

this.setState(prevState => {

console.log(prevState.todos.text)

})

}

For example only logs null to the console.

17:19 — Why is this the case? I don’t know. Anyway, I’m going to watch the video solution once and then call it quits for this pomodoro session.

17:23 — Of course I knew in the above code that I needed to use .map( ) to actually solve the problem, but I was using console.log for debugging purposes to try to get some ideas about how I could solve the problem on my own. In any case, I’m pretty brain dead at this point. I’m going to take a break and try to come back to the react course later tonight.

20:02 — I’m back.

20:14 — Me oh my, I think I’m on to something.

handleChange(id) {

const legos = this.state.todos.map(thing => {

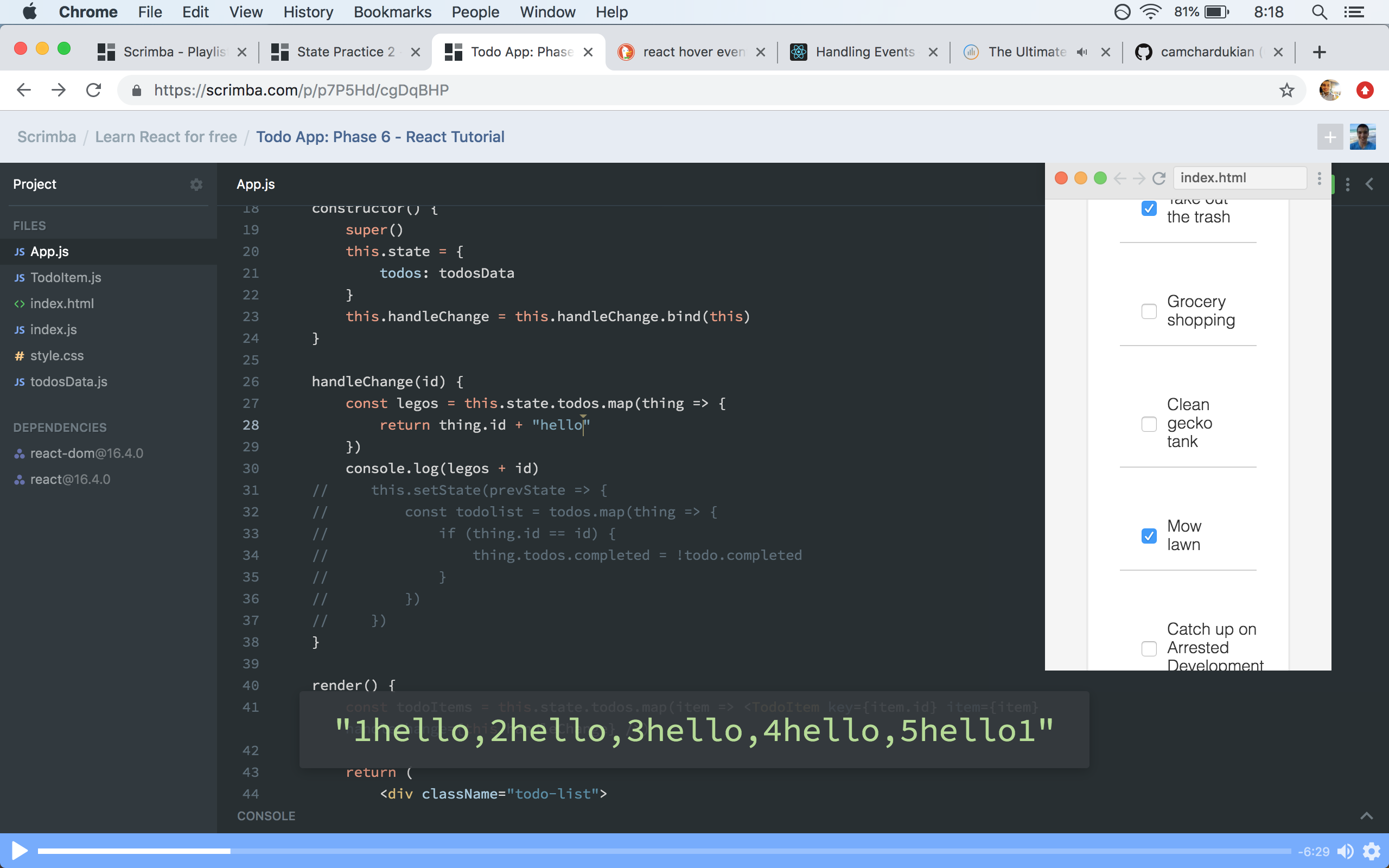
return thing.id + 3

})

console.log(legos)

}

20:15 — In the above code, let’s forget the horrible variable name of legos lol. While I was previously having a problem where everything was coming up as nulls, now I’m getting the above code to return some values. Let me just switch up the logic a bit, and I think everything will be looking good.

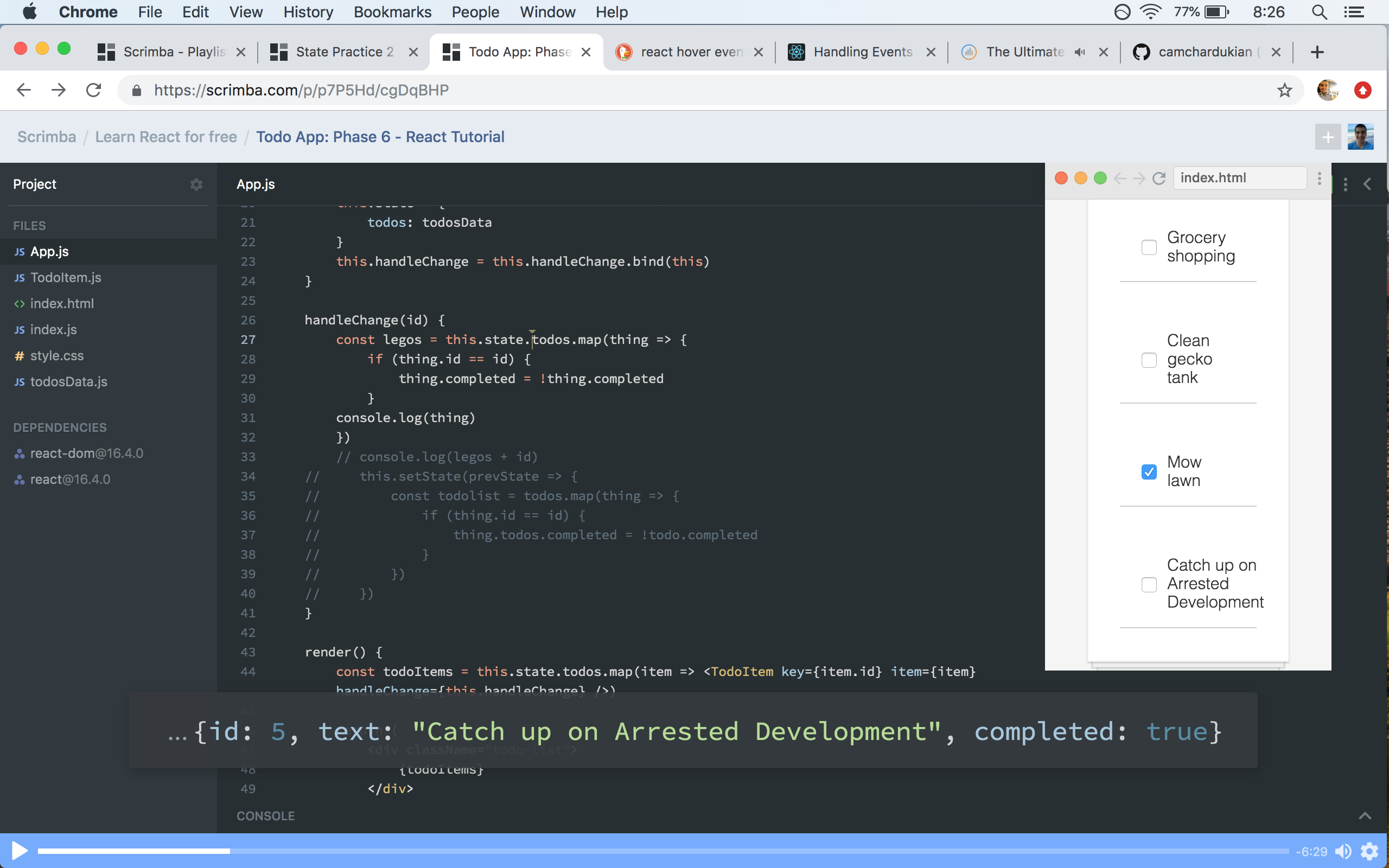


20:26 — Getting closer… The problem I’m facing now is trying to figure out how I should structure all of my return statements.

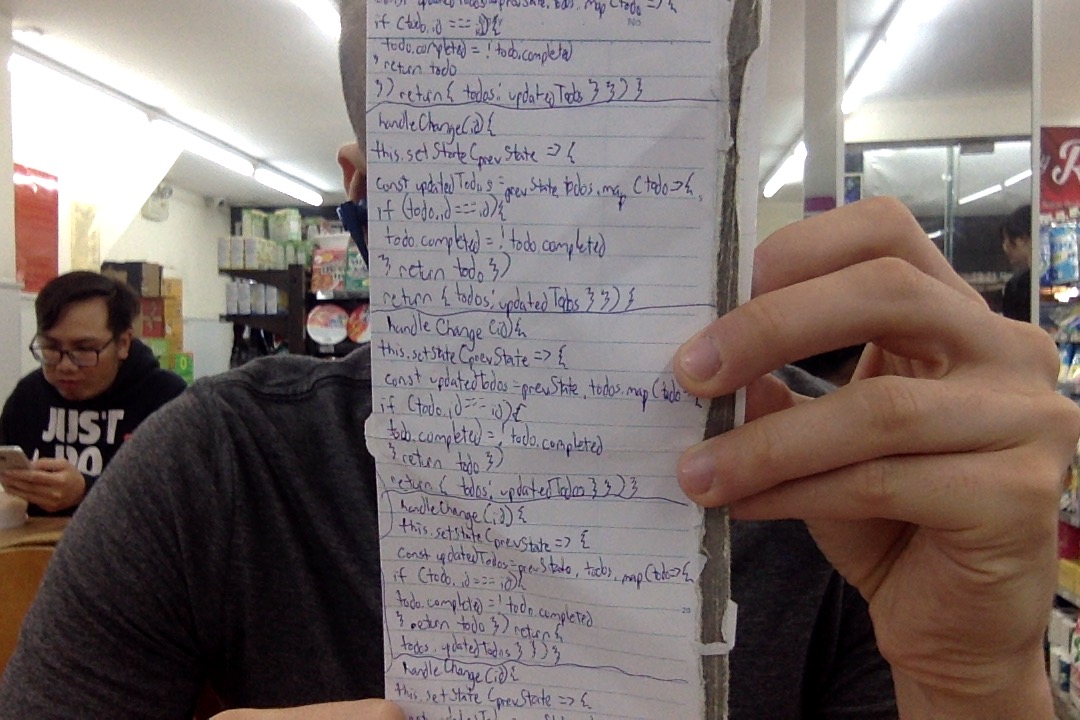
20:29 — Oh my god. I’ve been making such a noob error. I was reading my console and for the last 10 minutes or so I’ve been thinking that my program has had some problem where for some reason it only outputs the last item or more appropriately speaking the last “thing” being mapped. It turns out, however, that my console was outputting everything it’s just that Scrimba’s user interface won’t print everything you’ve console.logged unless you click the console.log button at the bottom of the page.

20:32 — With that being said, it has just come to my attention that I’ve been trying to set the state directly instead of using the setState method. This could also be causing some errors.

20:35 — It’s getting late. I’m going to watch the video again.

20:38 — I’m going to spend a few minutes writing code by hand.

20:50 — It’s been a solid day. See you again tomorrow. I’m excited to start doing this journal using markdown tomorrow!

**Total time spent coding today: 2 hours 54 minutes**

**Total time spent coding thus far in May 2019: 27 hours 32 minutes**

**Total lifetime hours of coding: 523 hours 26 minutes**