REACT LIFECYCLE WORKSHOP 2.0

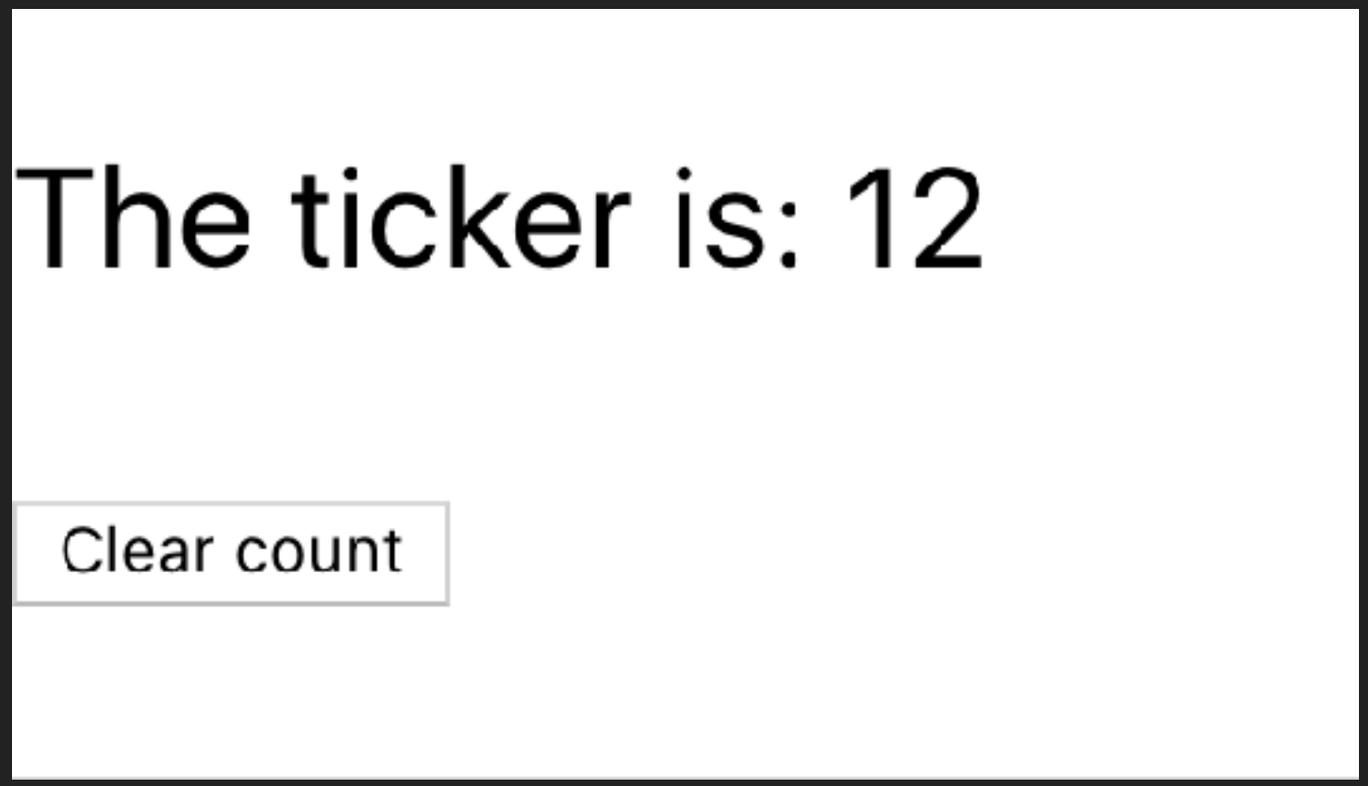
QCC TECH WORKS

CREATE-REACT-APP

- Create a folder named ‘React’ or if you’ve already have a directory you want the project to be installed to
- Type - “npx create-react-app ticker” in our console where you want the folder containing the project to be installed into.
- Follow the suggested prompt of running the commands

  cd ticker
  npm start
- Now we should see our server running and the react-app boilerplate site being displayed.

What we want to accomplish is a ticker that keeps counting up.

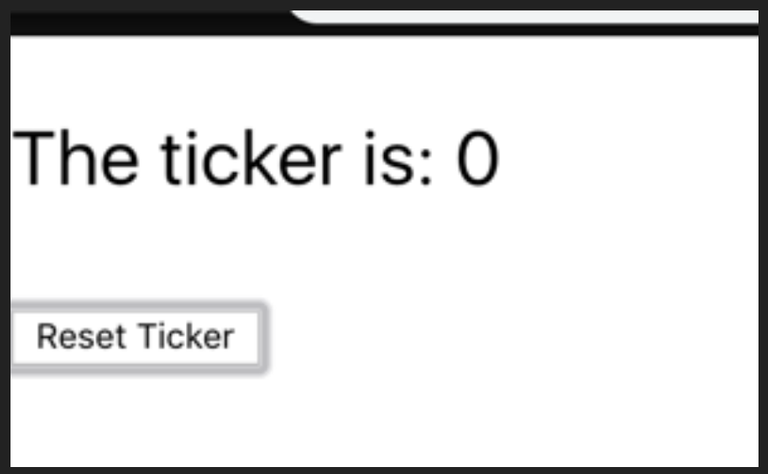


- First, think about what type of component that you'll be creating for this workshop.
- Secondly, think about the functionality of this program and what type of methods we will use to accomplish a ticker that counts up.
- Third, think about how you are going to format this data and how we will display it correctly in JSX.

- For this program we're going to need to create a stateful class component. This will allow us to create state that we can update/use in our program to represent the ticker that is incrementing.

LIFECYCLE WORKSHOP

- The main functionality of this program is to create a ticker that will mount to the DOM and update the count, and renders all updates. Also, you want to be able to clear the count and have it start back at 0.




GET TO CODING!
AWWWWWWW YEEEEAAHH
imgflip.com

- First we need to delete everything in app.js file and start from scratch.

Start off by importing your required libraries

```
1  import React, {Component} from 'react'
```

Then build out the base of your component

```
3  class Ticker extends Component {
4    render() {
5      return(
6        <div></div>
7      )
8    }
9  }
10
11 export default Ticker
```

- Since we know we'll use state for this application let's build our constructor function.

```
4   constructor() {
5     super();
6     this.state = {
7       count: 0
8     }
9   }
```

Just like our counter application we are going to initialize state with the count starting at 0

- Now we need to set up the JSX.
- Think about what lifecycle method will we want to use for the ticker to start incrementing. As well as what method we're going to use to create a ticker incrementing every second.


TIME FOR THAT
"RECAP"
memegenerator.net

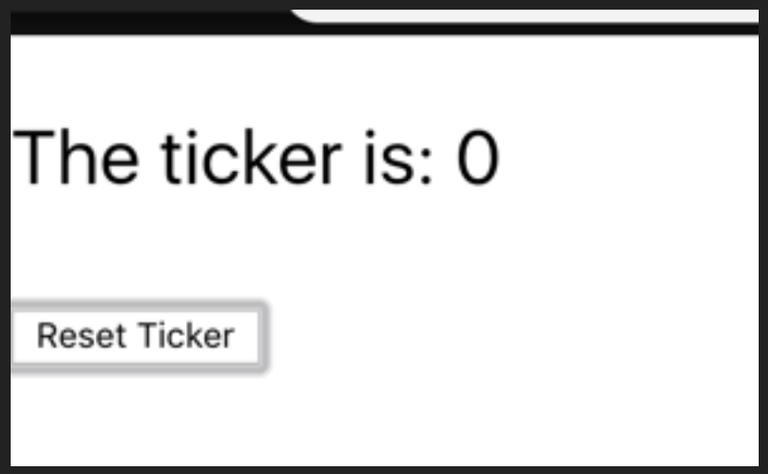
- To accomplish this we are going to use the componentDidMount() method, and place our logic in here to start the ticker incrementing.

```
13    componentDidMount() {
14      //put code here
15    }
```

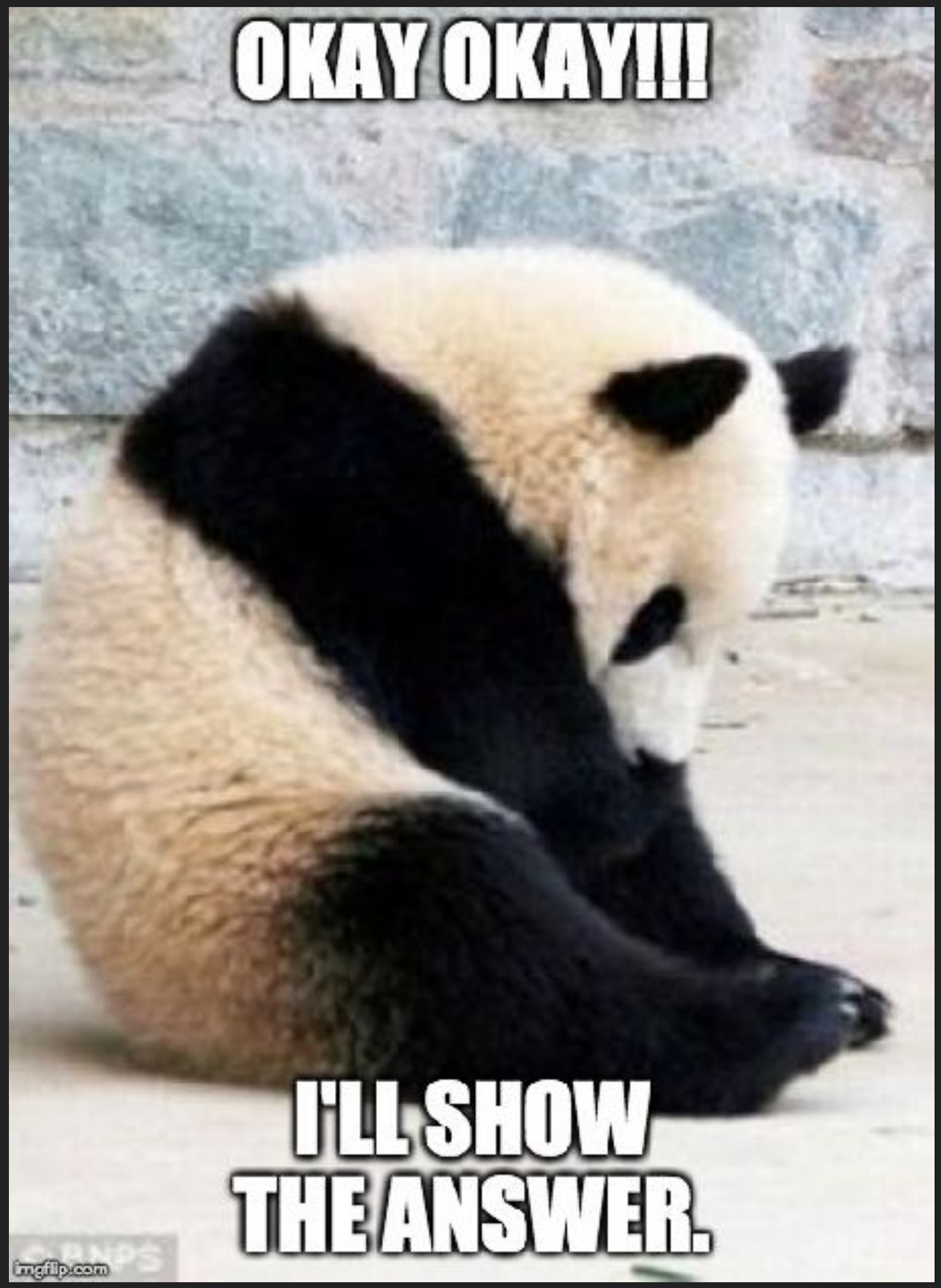
Remember to put methods after the constructor function and before your render method

This is a built in React method, these built-in lifecycle methods follow this format

- Now with your new knowledge of how use setInterval try and build the JSX and have the count reflected as well.



Right now this is all we need.


OKAY OKAY!!!
I'LL SHOW
THE ANSWER.
imgflip.com

- To accomplish this task we want to use the setInterval method to start incrementing the count by 1 every second. And the code that we want the setInterval to execute is the setState method that will change the count by 1.

```
11  componentDidMount() {
12    setInterval(() => {
13      this.setState({
14        count: this.state.count + 1
15      })
16    },1000)
17  }
```

- Now to make the ticker displayed in JSX.

```
20      render() {
21          return(
22              <div>
23                  <p>The ticker number is : {this.state.count}</p>
24              </div>
25          )
26      }
```

- Now you should see your ticker incrementing every second and reflected in your browser.

- BUT WAIT!!! Maybe just maybe you want it only to show the numbers that are divisible by 3. YESS that makes total sense :p.

If we want to accomplish this we’re going to have to use another lifecycle method that will only update when this condition passes.

- Let me introduce you to shouldComponentUpdate()

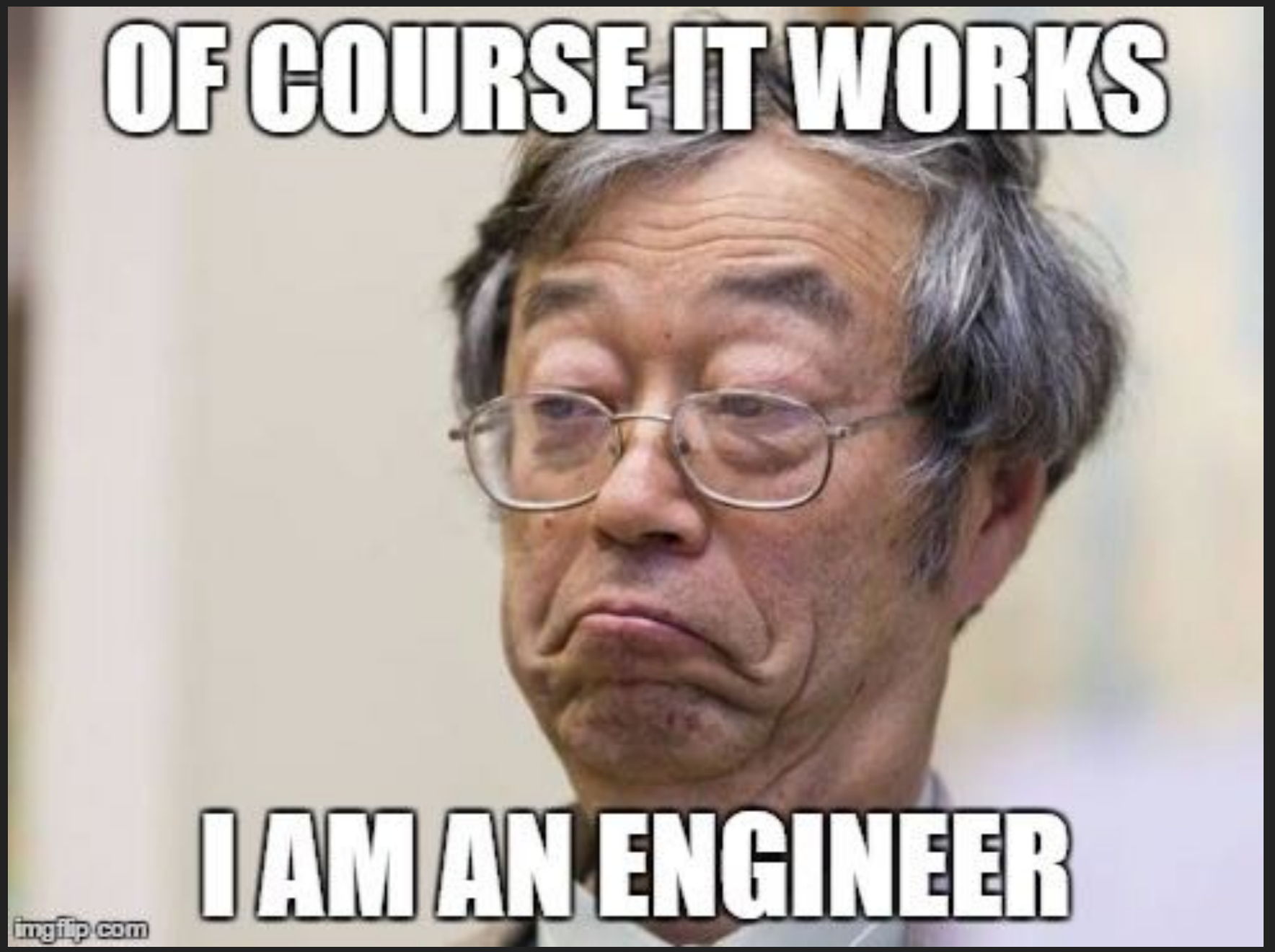
- shouldComponentUpdate() is invoked before rendering when new props or state are being received. It defaults to true. It takes the arguments of nextProps and nextState, and only allows the update if your conditional returns true.

```
19  shouldComponentUpdate (nextProps, nextState) {
20
21  }
```

- Think about what conditional you want to return true so that this method will allow the update to occur.
- We only wanted to show numbers that are multiples of 3.

- How we're going to accomplish this is by checking if the nextState that is being set is going to be divisible by 3

```
19    shouldComponentUpdate (nextProps, nextState) {
20      if (nextState.count % 3 === 0) return true;
21      else return false;
22    }
```


OF COURSE IT WORKS
I AM AN ENGINEER
imgflip.com

- Now I want you to make a reset button that will reset the ticker to 0, without stoping it.
- If you finish that I want you to start working on making a button that will pause the ticker, and then resume the ticker when clicked again. Also make the button show “Ticker is paused” and “Pause the Ticker” when the ticker is running, or is paused.
- If you figured that out start styling it like a boss and show me some awesome CSS ninja skills.