POWERING UP *with* REACT

Level 5 – Section 1

Talking to Remote Servers

Using Lifecycle Methods to Load Comments

Comments Are Static

In the real world, we'd want to pull comments from an API instead of hard-coding the data.

```
class CommentBox extends React.Component {

  constructor() {
    super();

    this.state = {
      showComments: false,
      comments: [
        { id: 1, author: 'Morgan McCircuit', body: 'Great picture!' },
        { id: 2, author: 'Bending Bender', body: 'Excellent stuff' }
      ]
    };
  }
  ...
}
```

Hard-coded data

Loading Comments From a Remote Server

Let’s set the initial state of *comments* as an empty array so we can later populate it with data from an API server.

```
class CommentBox extends React.Component {

  constructor() {
    super();

    this.state = {
      showComments: false,
      comments: [ ]
    };
  }
  ...
}
```

Initialized to an empty array

Adding jQuery as a Dependency

jQuery will help us make Ajax requests. We can download it from the jQuery website and include it in our HTML page.

Project Folder

- index.html
- components.js
- vendors
  - react.js
  - react-dom.js
  - babel.js
  - jquery.js

Download it from the jQuery website

index.html

```
<!DOCTYPE html>
<html>
  <body>
    <div id="story-app"></div>
    <script src="vendors/react.js"></script>
    <script src="vendors/react-dom.js"></script>
    <script src="vendors/jquery.js"></script>
    <script src="vendors/babel.js"></script>
    <script type="text/babel"
            src="components.js"></script>
  </body>
</html>
```

Brush up on your **Ajax skills** with our jQuery: The Return Flight course

How to Fetch Data in a Component

Let's write a class method that will make Ajax requests in the *CommentBox* component.

```
class CommentBox extends React.Component {
  ...
  _fetchComments() {
    jQuery.ajax({
      method: 'GET',
      url: '/api/comments',




    });
  }
}
```

Makes call to the remote server

Setting State With Data From a Remote Server

We call the *setState* method when data is received from the API server.

```
class CommentBox extends React.Component {
  ...
  _fetchComments() {
    jQuery.ajax({
      method: 'GET',
      url: '/api/comments',
      success: (comments) => {
        this.setState({ comments })
      }
    });
  }
}
```

Arrow function preserves the this binding to our class

...this refers to CommentBox

Deciding Where to Call *_fetchComments()*

```
...

class CommentBox extends React.Component {
  render() {
  }

  _fetchComments() {
    ...
  }

}
```

That means we can't call _fetchComments() from render – we'll get an infinite loop!

fetchComments calls setState, which calls render()

We need to call *_fetchComments* **before** *render()* is called.

TODO: make codeshcool short url

React's Lifecycle Methods

Lifecycle methods in React are functions that get called while the component is rendered for the first time or about to be removed from the DOM.

We should call _fetchComments here!

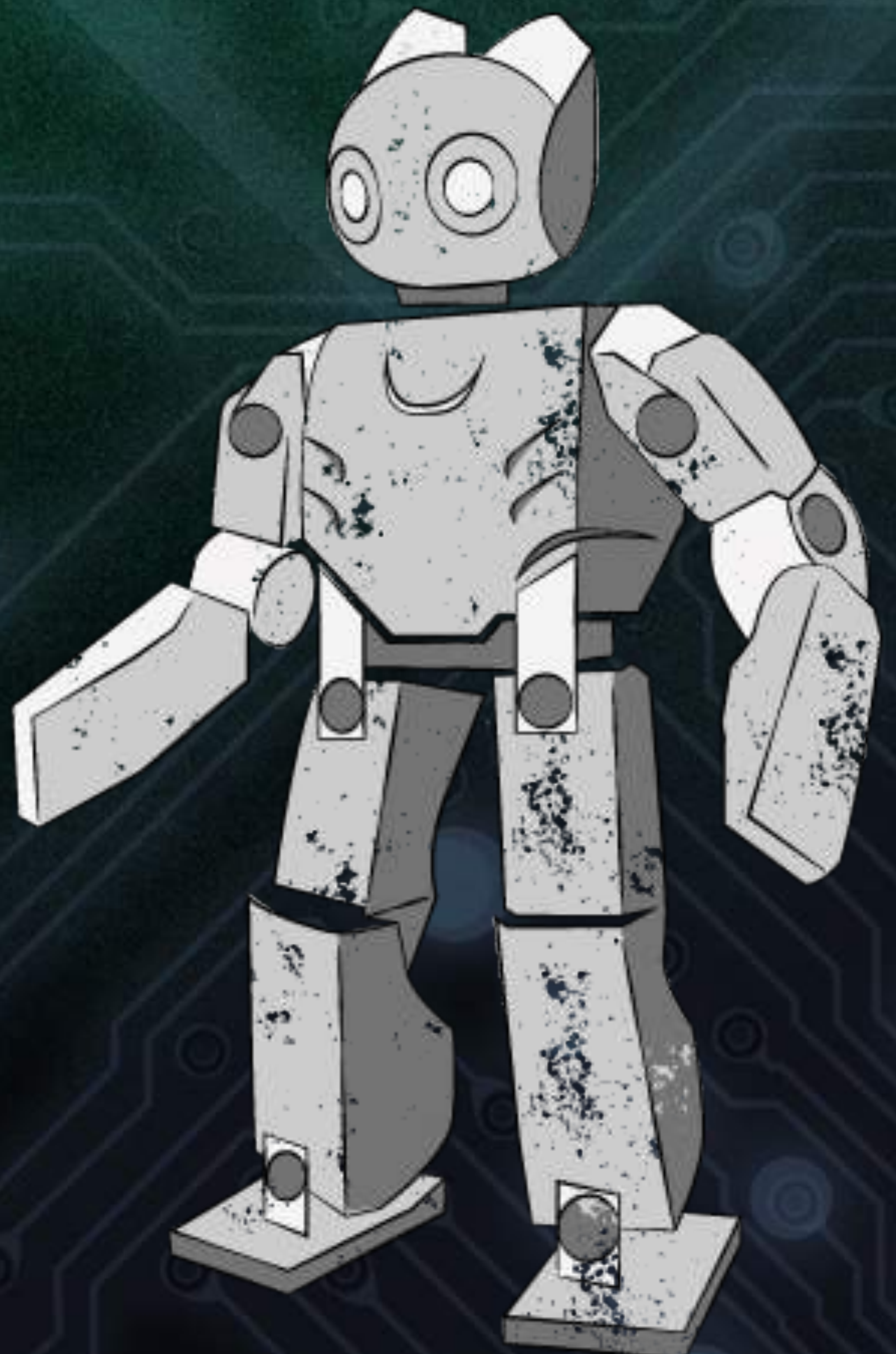
constructor()

componentWillMount()

render()

componentDidMount()

componentWillUnmount()



Note: In React, **mounting** means rendering for the first time.

For a full list of React's lifecycle methods, visit http://go.codeschool.com/react-lifecycle-methods

Fetching Data on the Mounting Phase

The *componentWillMount* method is called **before** the component is rendered to the page.

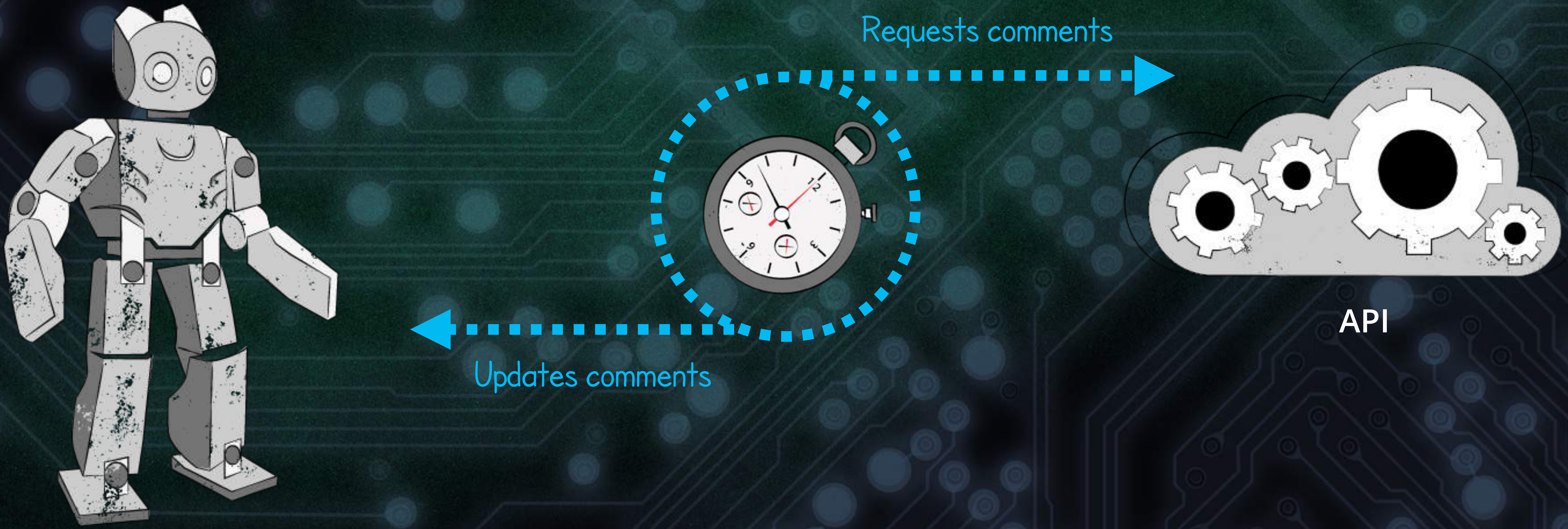
```
class CommentBox extends React.Component {
  ...
  componentWillMount() {
    _fetchComments();
  }

  _fetchComments() {
    jQuery.ajax({
      method: 'GET',
      url: '/api/comments',
      success: (comments) => {
        this.setState({ comments })
      }
    });
  }
}
```

Fetch comments from server before component is rendered.

Getting Periodic Updates

In order to check whether new comments are added, we can periodically check the server for updates. This is known as **polling**.



Polling Data on the Mounting Phase

The *componentDidMount* method is called **after** the component is rendered to the page.

```
...

class CommentBox extends React.Component {
  ...
  componentDidMount() {
    setInterval(() => this._fetchComments(), 5000);
  }
}
```

Polling the server every five seconds

5,000 milliseconds is equal to five seconds

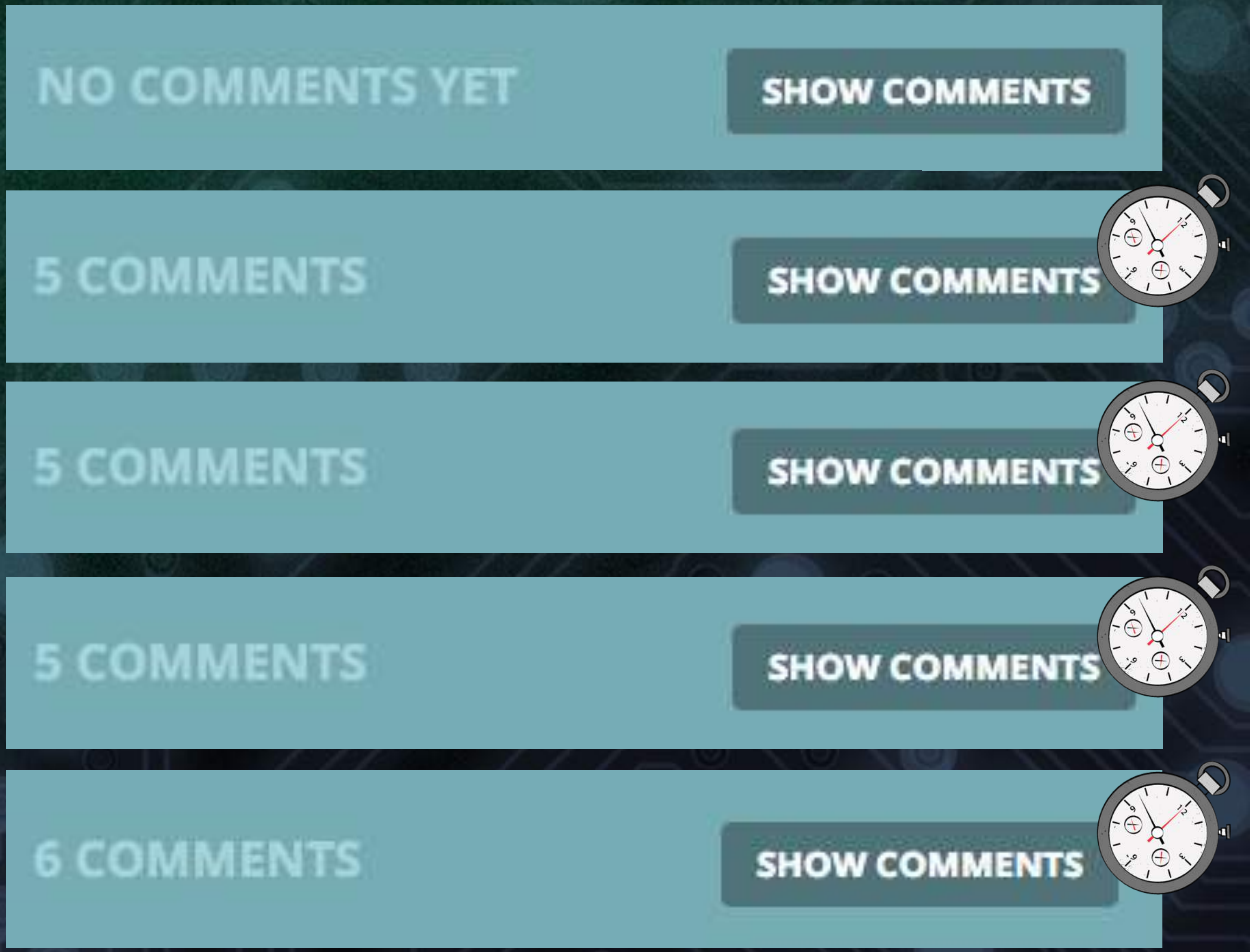
Updating Component With New Comments

React optimizes the rendering process by **only updating the DOM** when **changes are detected** on the resulting markup.

New state value after initial Ajax request → DOM change happens

No new state value after second periodic Ajax request → No DOM change

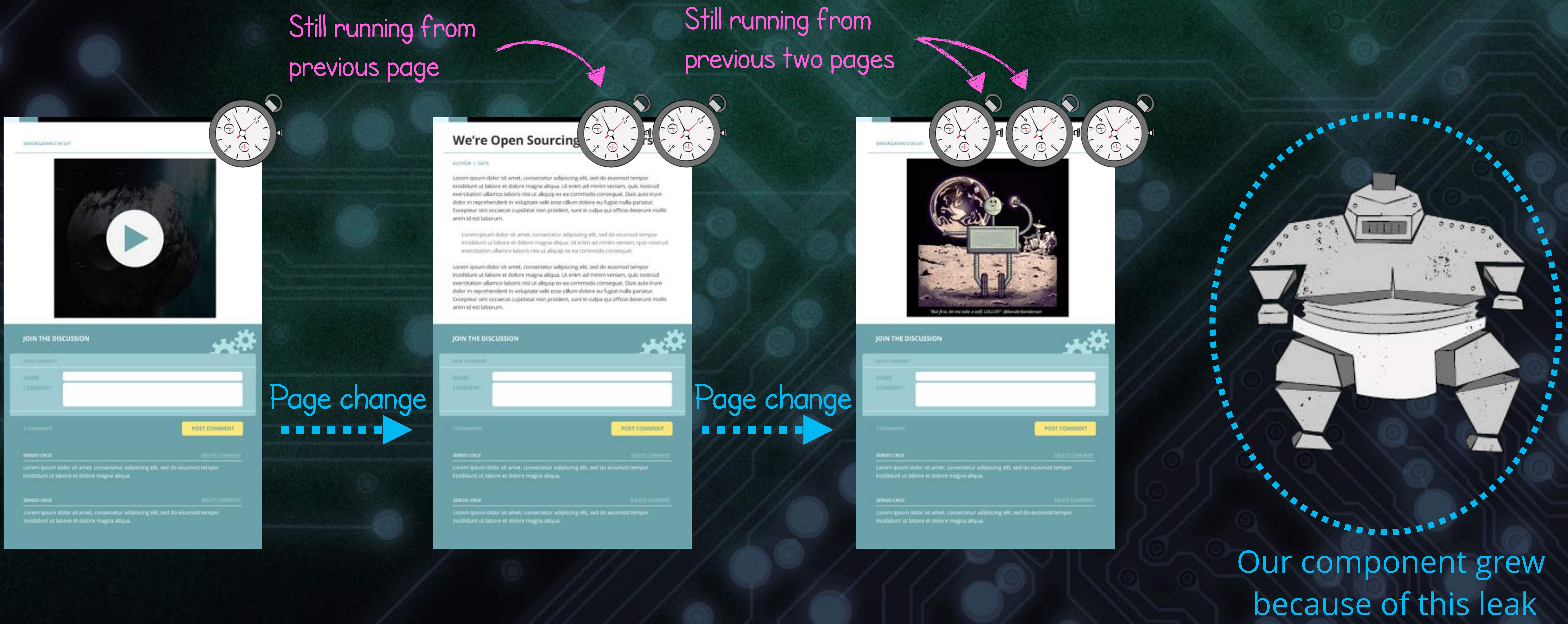
New state value after third periodic Ajax request → DOM change happens



Note: render() is called after each Ajax response because setState is in the response function.

Memory Leaks on Page Change

Page changes in a single-page app environment will cause each *CommentBox* component to keep loading new comments every five seconds, even when they're no longer being displayed.


Still running from previous page
Still running from previous two pages
We're Open Sourcing
JOIN THE DISCUSSION
POST COMMENT
Page change
Page change
Our component grew because of this leak

Preventing Memory Leaks

Each component is responsible for removing any timers it has created. We will remove the *timer* on the *componentWillUnmount* method.

Store timer as object property

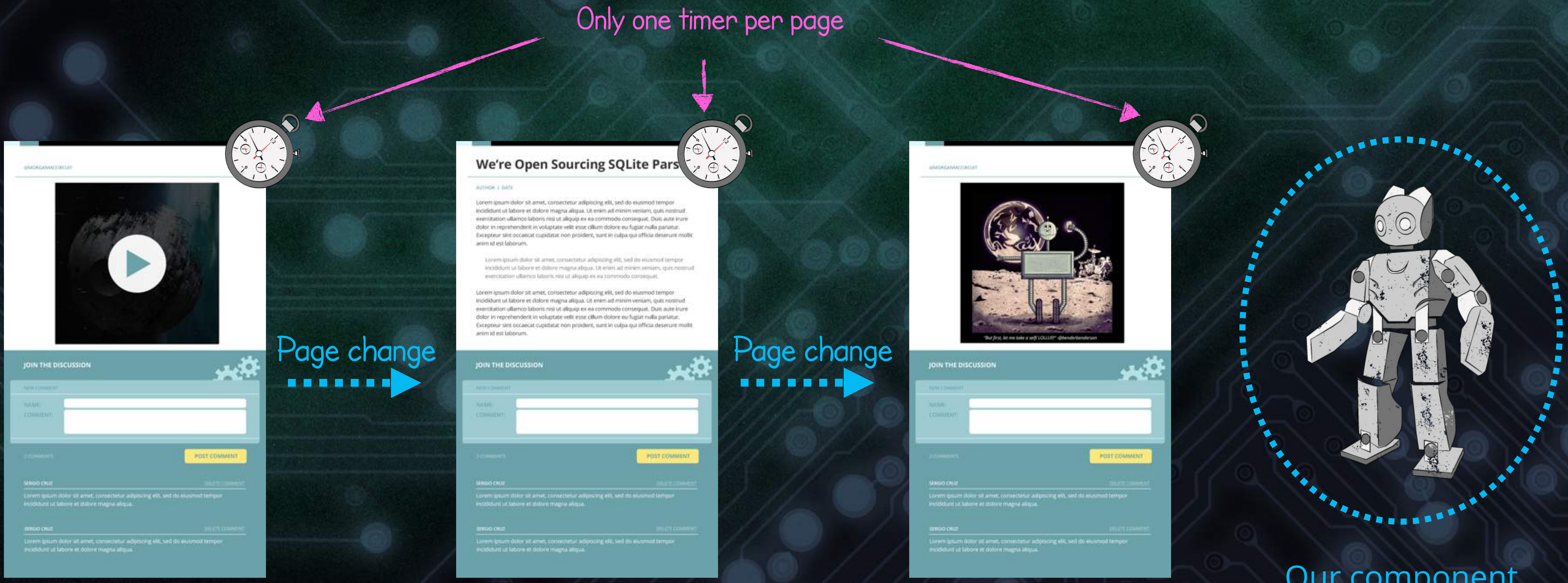
```
...
class CommentBox extends React.Component {
  ...
  componentDidMount() {
    this._timer = setInterval(
     () => this._fetchComments(),
     5000
    );
  }

  componentWillUnmount() {
    clearInterval(this._timer);
  }
}
```

Run when component is about to be removed from the DOM

Memory Leak Is Gone

Our app can be freely navigated through now, without causing multiple unnecessary calls to the API.



Our component is smaller again!

Reviewing the Steps for Loading Comments

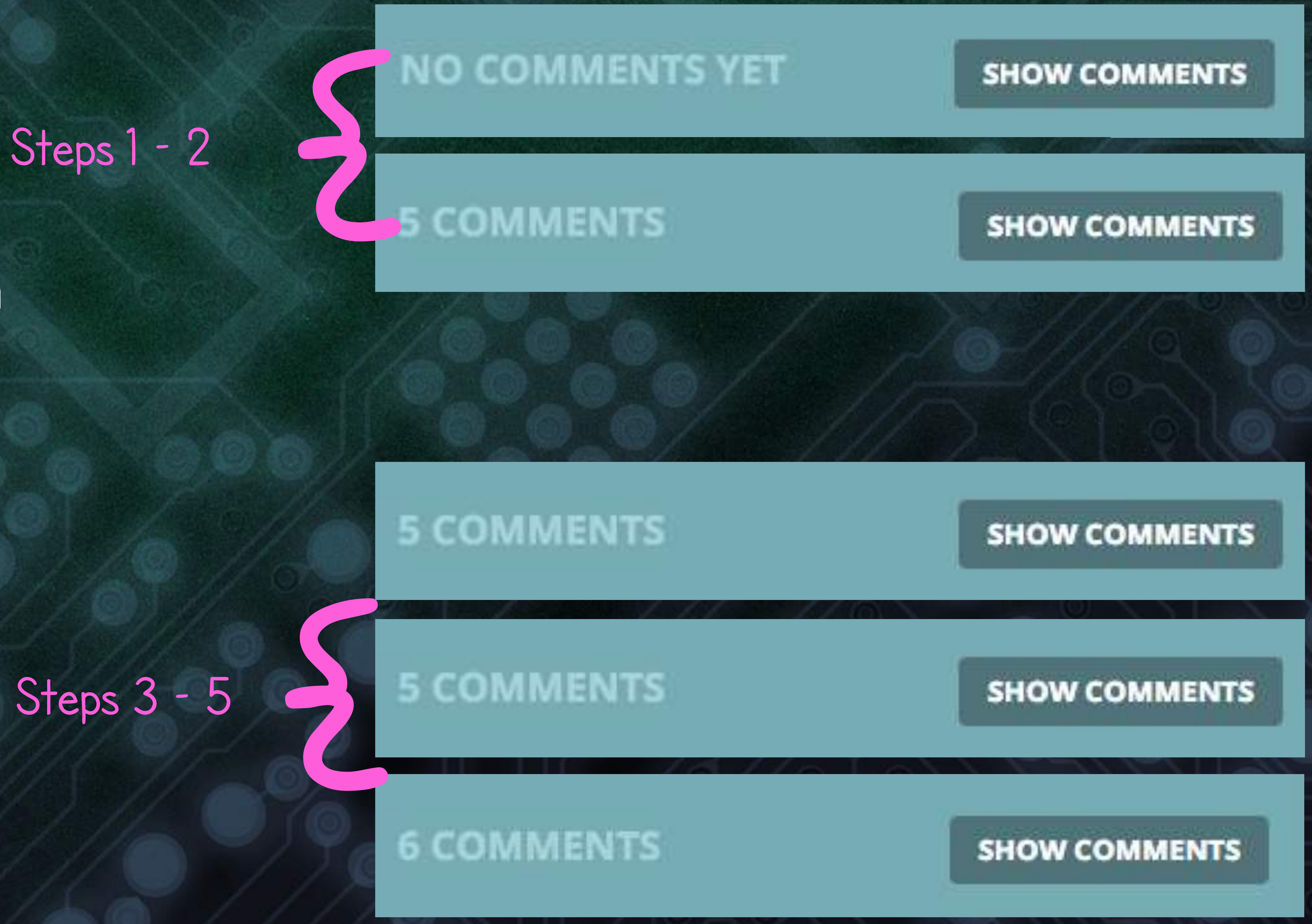
1 - *componentWillMount()* is called.

2 - *render()* is called and *CommentBox* is mounted.

3 - Component waits for API response and when it is received, *setState()* is called, causing *render()* to be called again.

4 - *componentDidMount()* is called, causing *this._fetchComments()* to be triggered **every five seconds.**

5 - *componentWillUnmount()* is called when the component is about to be removed from the DOM and clears the *fetchComments* timeout.



Quick Recap on Lifecycle Methods

Lifecycle methods in React are functions that get called during certain *phases* that components go through.

componentWillMount() is called **before** the component is rendered.

componentDidMount() is called **after** the component is rendered.

componentWillUnmount() is called immediately before the component is **removed from the DOM.**

More lifecycle methods at
http://go.codeschool.com/react-lifecycle-methods

POWERING UP *with* REACT