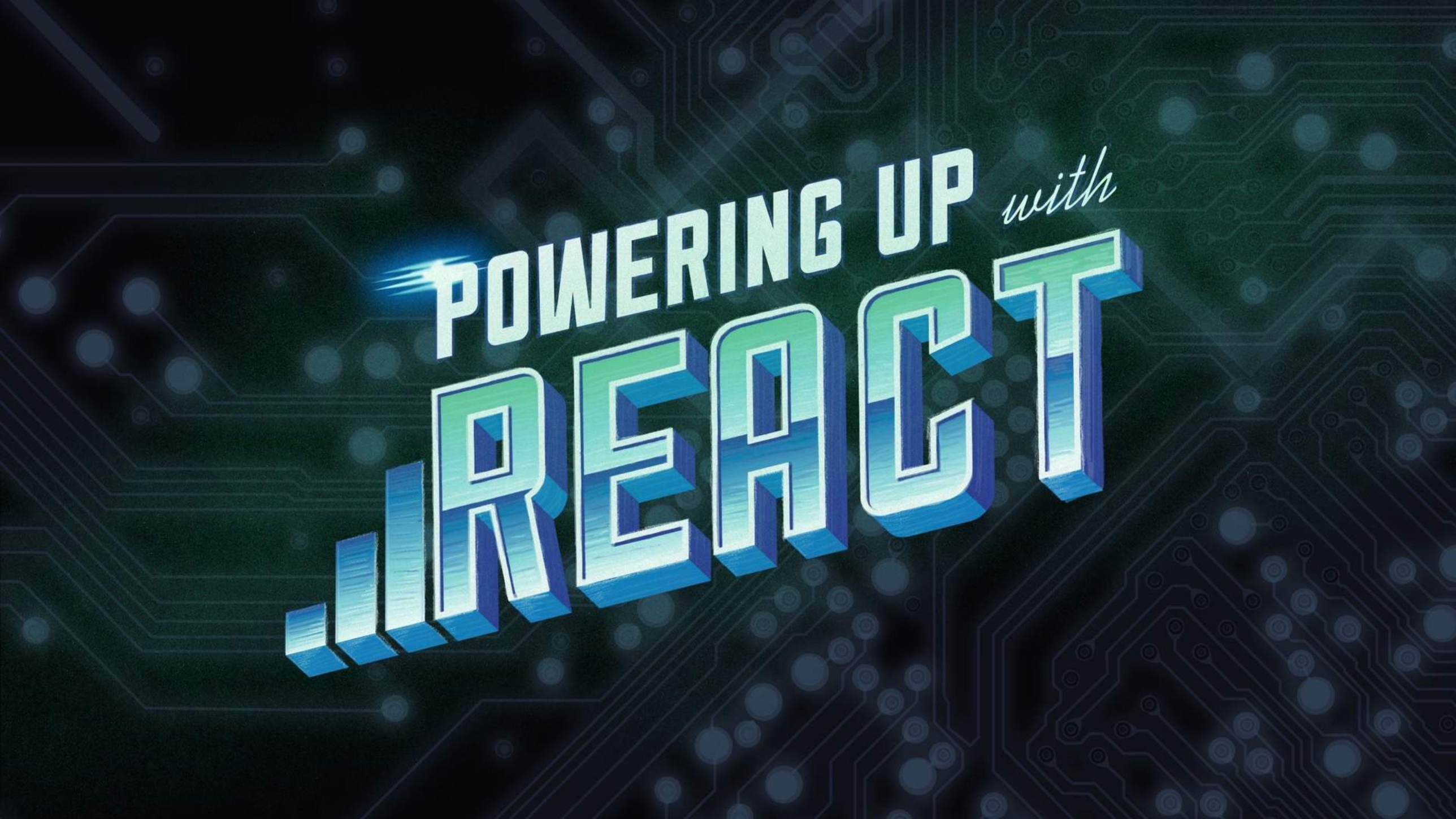

POWERING UP with
REACT

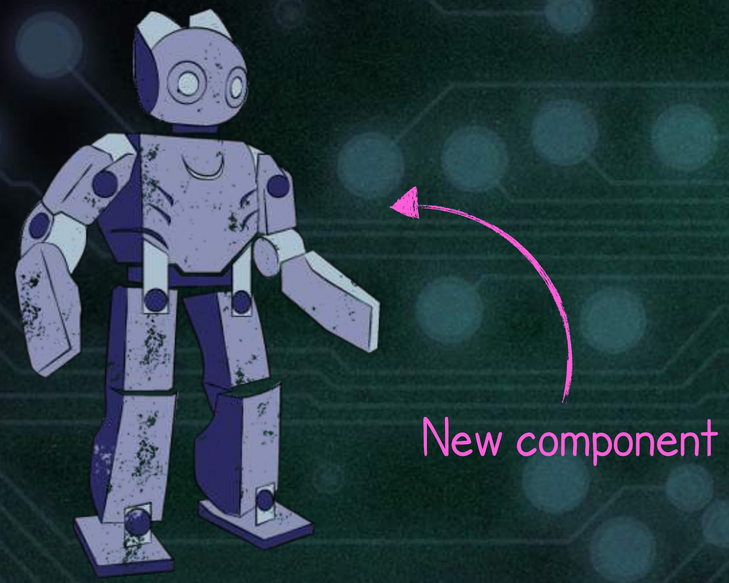
Level 4

Synthetic Events

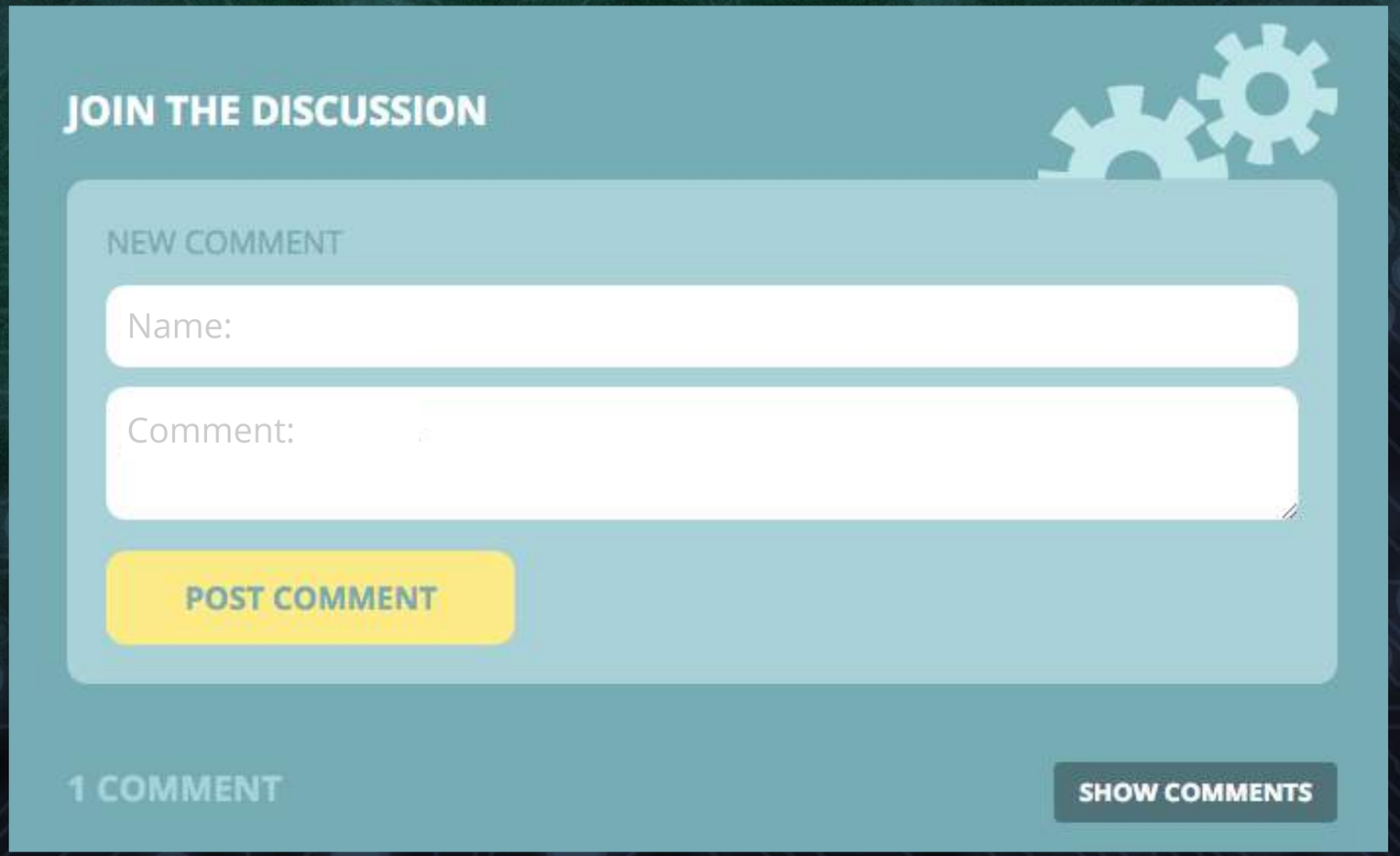
Capturing User Actions

Adding New Comments

We want to let users add new comments to our app.



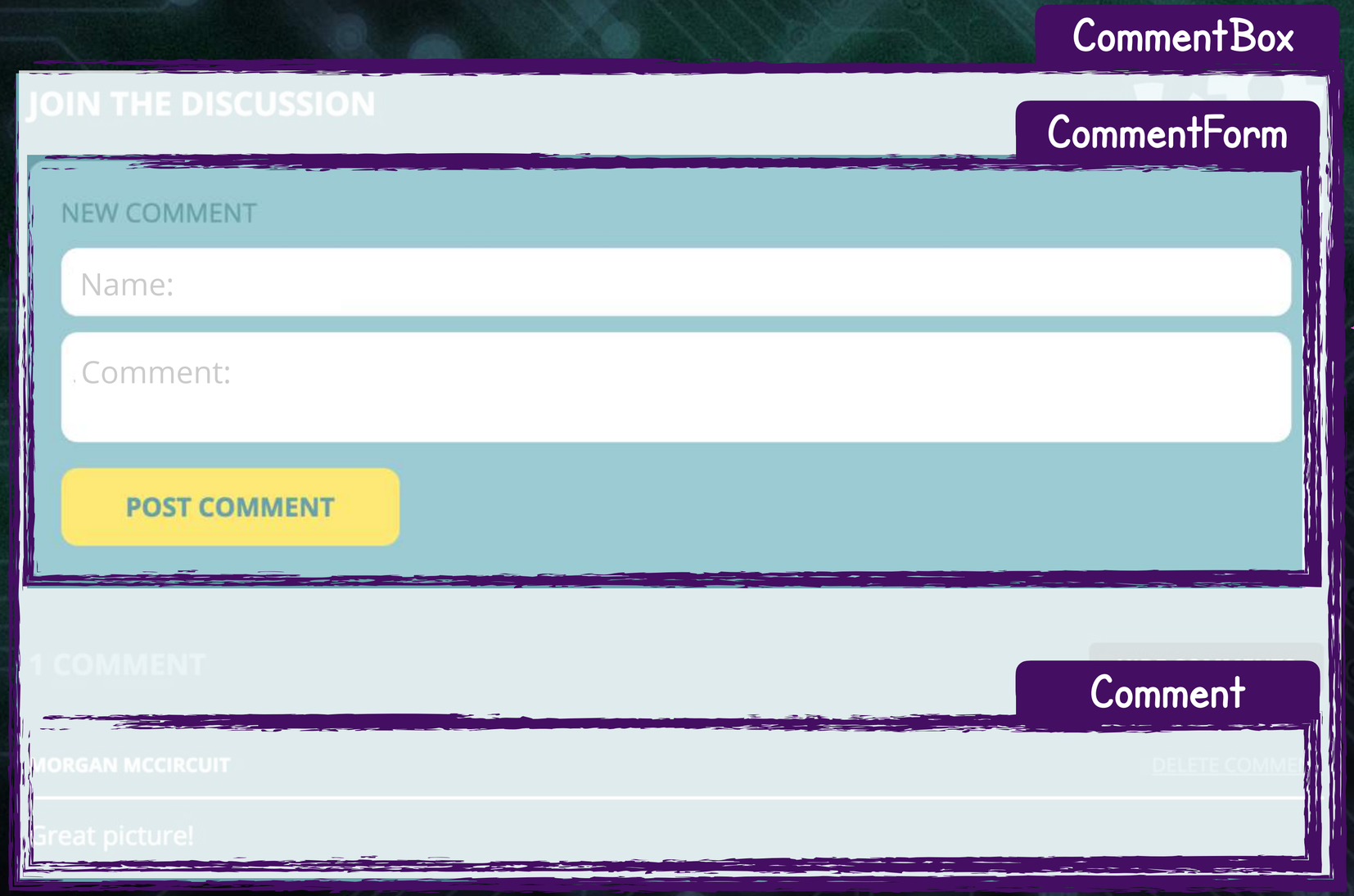
CommentForm



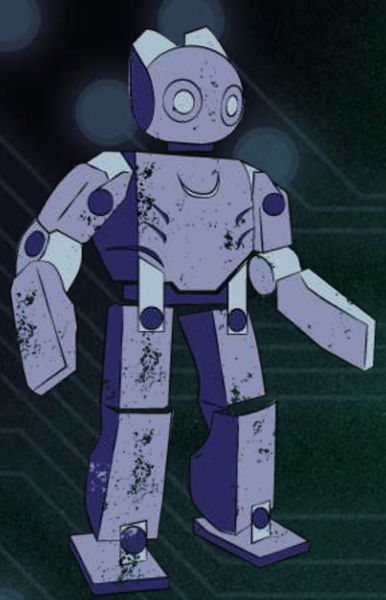
How should we build this new form in React?

New Component: *CommentForm*

CommentForm is a new component that will allow users to add comments to our app.



Coding the *CommentForm* Component



CommentForm

```
class CommentForm extends React.Component {
  render() {
    return (
      <form className="comment-form">
        <label>Join the discussion</label>
        <div className="comment-form-fields">
          <input placeholder="Name:"/>
          <textarea placeholder="Comment:"></textarea>
        </div>
        <div className="comment-form-actions">
          <button type="submit">
            Post comment
          </button>
        </div>
      </form>
    );
  }
}
```

JSX markup for CommentForm

Adding an Event Listener to Our Form

To add an event listener to the form, we use the *onSubmit* prop and pass a handler to it.

```
class CommentForm extends React.Component {
  render() {
    return (
      <form className="comment-form" onSubmit={this._handleSubmit.bind(this)}>
        ...
          <input placeholder="Name:"/>
          <textarea placeholder="Comment:"></textarea>
        ...
      </form>
    );
  }

  _handleSubmit(event) {
    event.preventDefault();
  }
}
```

Adds an event listener to the submit event

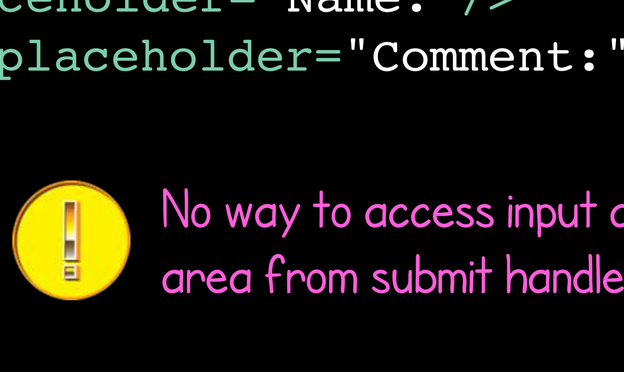
Don't forget to bind event handlers, otherwise this will not work!

Prevents page from reloading

Problem: Can't Access User Input in *handleSubmit()*

```
class CommentForm extends React.Component {
  render() {
    return (
      <form className="comment-form" onSubmit={this._handleSubmit.bind(this)}>
        ...
          <input placeholder="Name:"/>
          <textarea placeholder="Comment:"></textarea>
        ...
      </form>
    );
  }

  _handleSubmit(event) {
    event.preventDefault();
  }
}
```



No way to access input and text area from submit handler

Accessing Form Data From Handler

We can use *refs* to assign form values to **properties** on the component object.

```
class CommentForm extends React.Component {
  render() {
    return (
      <form className="comment-form" onSubmit={this._handleSubmit.bind(this)}>
        ...
          <input placeholder="Name:" ref={(input) => this._author = input}/>
          <textarea placeholder="Comment:" ref={(textarea) => this._body = textarea}>
          </textarea>
        ...
      </form>
    );
  }

  _handleSubmit(event) {
    event.preventDefault();
  }
}
```

We'll use these refs to access values from the input elements.

What Setting the *refs* Is Actually Doing

```
<input placeholder="Name:" ref={(input) => this._author = input}/>
```

is the same as

DOM element passed into callback

```
<input placeholder="Name:" ref={
                                function(input) {
                                 this._author = input;
                                }.bind(this)
                              }/>
```

creates new class property named _author

this refers to CommentForm.

Note: React runs *ref* callbacks on render.

Passing the User Input to the *CommentBox*

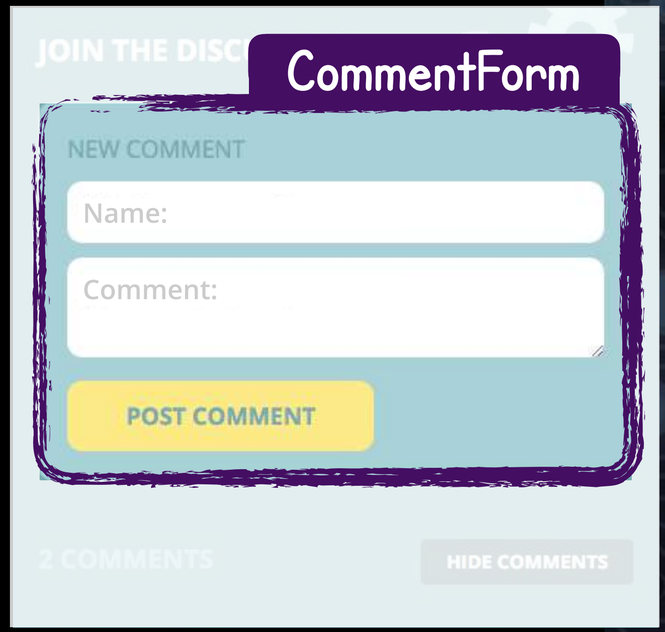
```
class CommentForm extends React.Component {
  render() {
    return (
    ...
      <input placeholder="Name:" ref={(input) => this._author = input}/>
      <textarea placeholder="Comment:" ref={(textarea) => this._body = textarea}>
    ...
    );
  }
  _handleSubmit(event) {
    event.preventDefault();

    let author = this._author;
    let body = this._body;

    this.props.addComment(author.value, body.value);
  }
}
```

Populated from refs in JSX

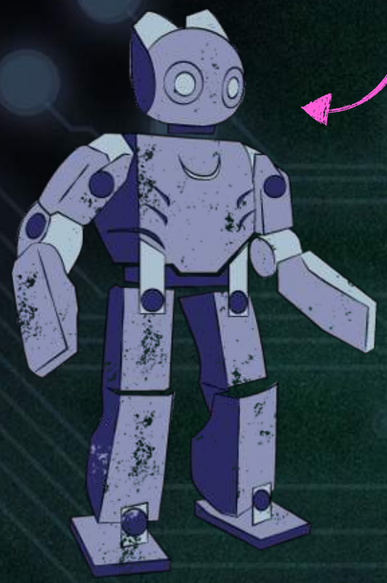
This method has been passed as an argument.



Data About Comments Lives in *CommentBox*

The array of comments is part of the *CommentBox* component, so we need to propagate new comments from *CommentForm* over to *CommentBox*.

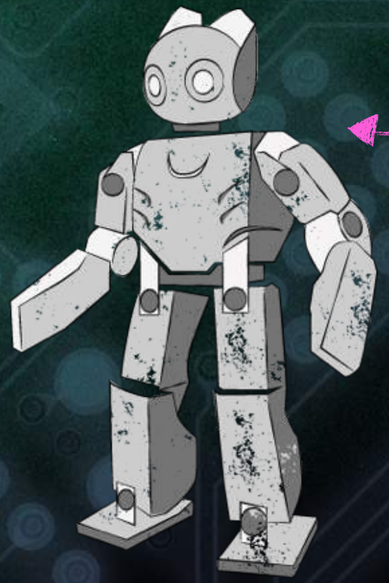
Has the new comment data



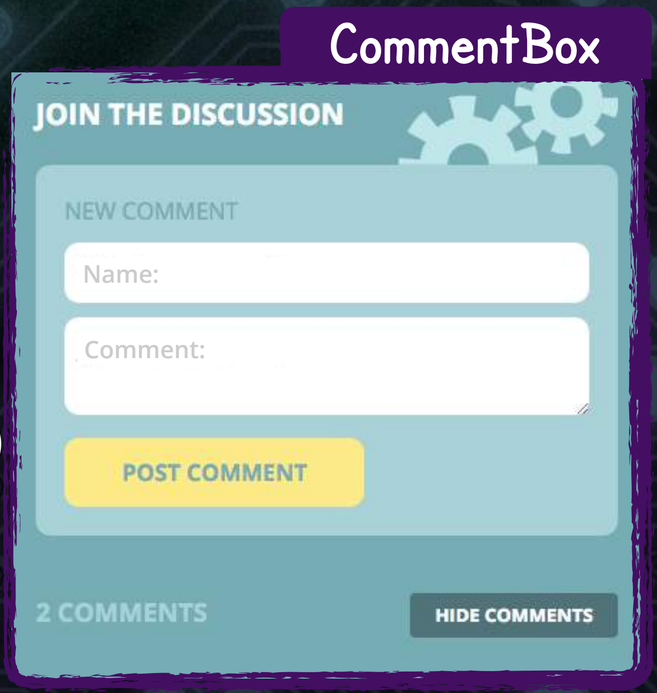
CommentForm

Propagate data about new comment to CommentBox

Has the comments array in its state



CommentBox (Parent)



Let's include *CommentForm* and pass it a callback prop.

Using *CommentForm* to Add Comments

Functions in JavaScript are **first-class citizens**, so we can pass them as **props** to other components.

```
class CommentBox extends React.Component { ...
  render() {
    return(
      <div className="comment-box">
        <CommentForm addComment={this._addComment.bind(this)} />
        ...
      </div>
    );
  }

  _addComment(author, body) {

  }

}
```

Using the newly created CommentForm component...

...and pas

animate second

animate first

...gets triggered by CommentForm when a new comment is added.

Adding Functionality to Post Comments

```
class CommentBox extends React.Component { ...
  render() {
    return(
      <div className="comment-box">
        <CommentForm addComment={this._addComment.bind(this)} />
        ...
      </div>
    );
  }
  ...
  _addComment(author, body) {
    const comment = {
      id: this.state.comments.length + 1,
      author,
      body
    };
    this.setState({ comments: this.state.comments.concat([comment]) });
  }
}
```

New comment object

New array references help React stay fast. So concat works better than push here.

Updates state when function is called by adding new comment

Comments Are Not Part of the State

Currently, we're defining an array every time the *_getComments* method is called. Let's move this data to the **state**.

```
class CommentBox extends React.Component {
  ...
   _getComments() {
     const commentList = [
       { id: 1, author: 'Morgan McCircuit', body: 'Great picture!' },
       { id: 2, author: 'Bending Bender', body: 'Excellent stuff' }
     ];

     ...
   }

}
```

Defining a variable can help us with prototyping, but it's time to change this!

Moving Comments to the State

Since comments will **change over time**, they should be part of the component's state.

```
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' }
      ]
    };
  }
  ...
}
```

Now part of the component's state

Rendering Comments From the State

Let's use the comments from the state object to render our component.

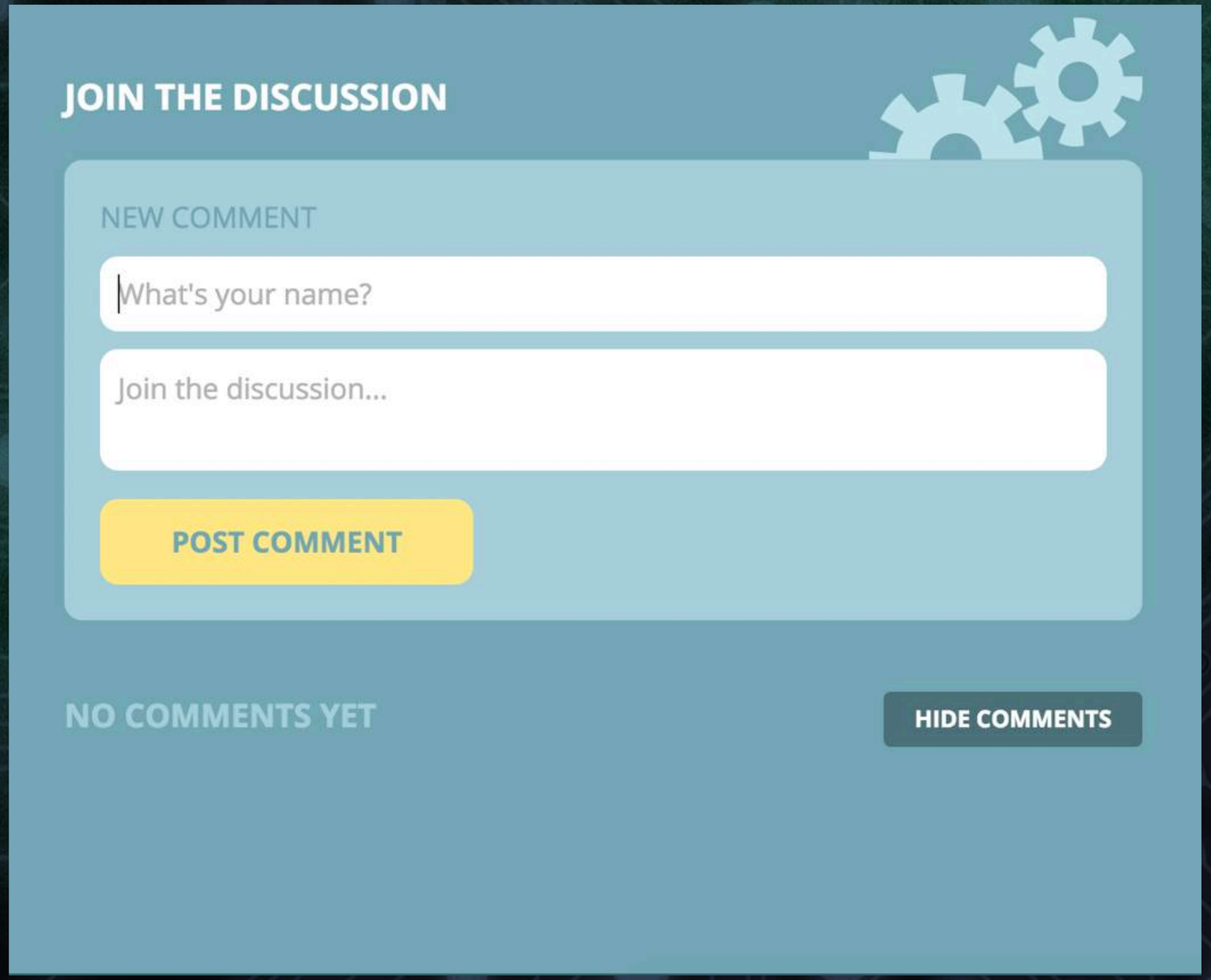
```
class CommentBox extends React.Component {
  ...
   _getComments() {

     return this.state.comments.map((comment) => {
        return (
          <Comment
            author={comment.author}
            body={comment.body}
            key={comment.id} />
        );
      });

   }
}
```

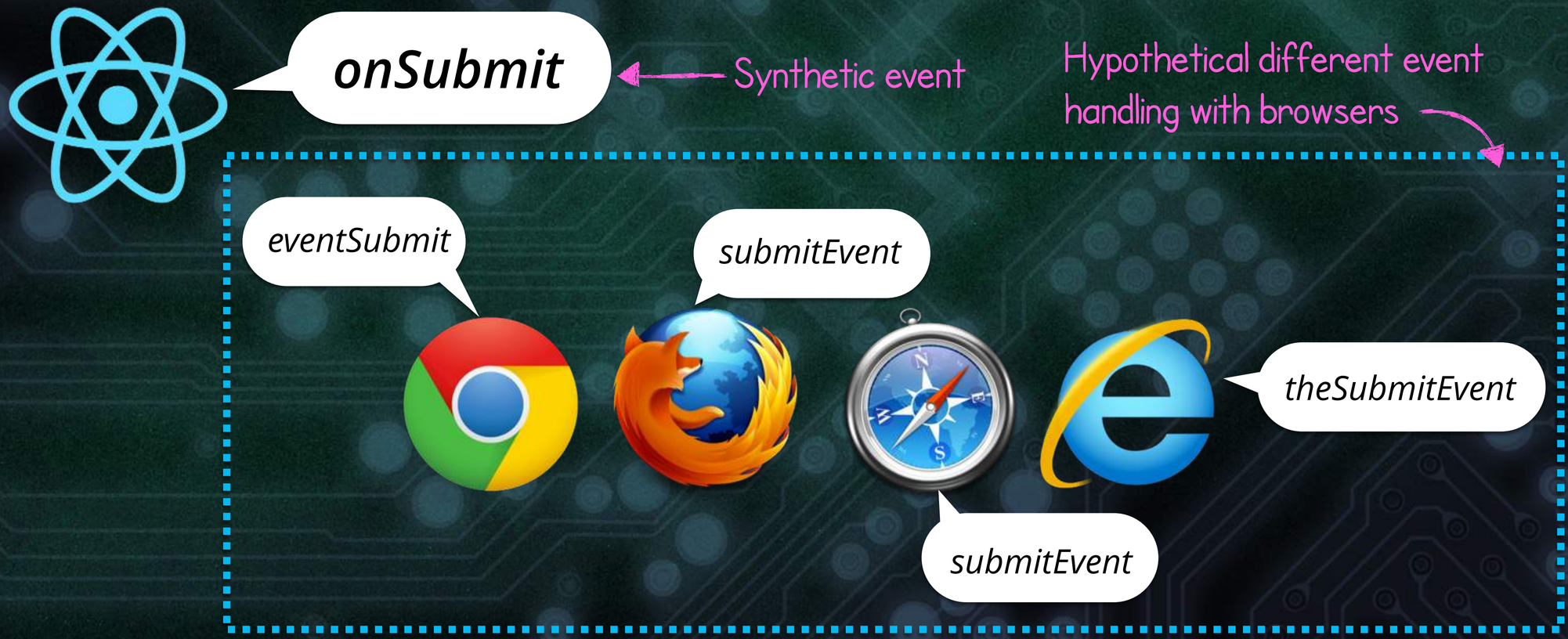
Reading from component's state

Demo: *CommentForm* Working



Review: Event Handling in React

In order to ensure events have **consistent properties across different browsers**, React wraps the browser's native events into **synthetic events**, consolidating browser behaviors into one API.



Synthetic events are my jam!

For the full list of browser events supported by React, visit http://go.codeschool.com/react-events

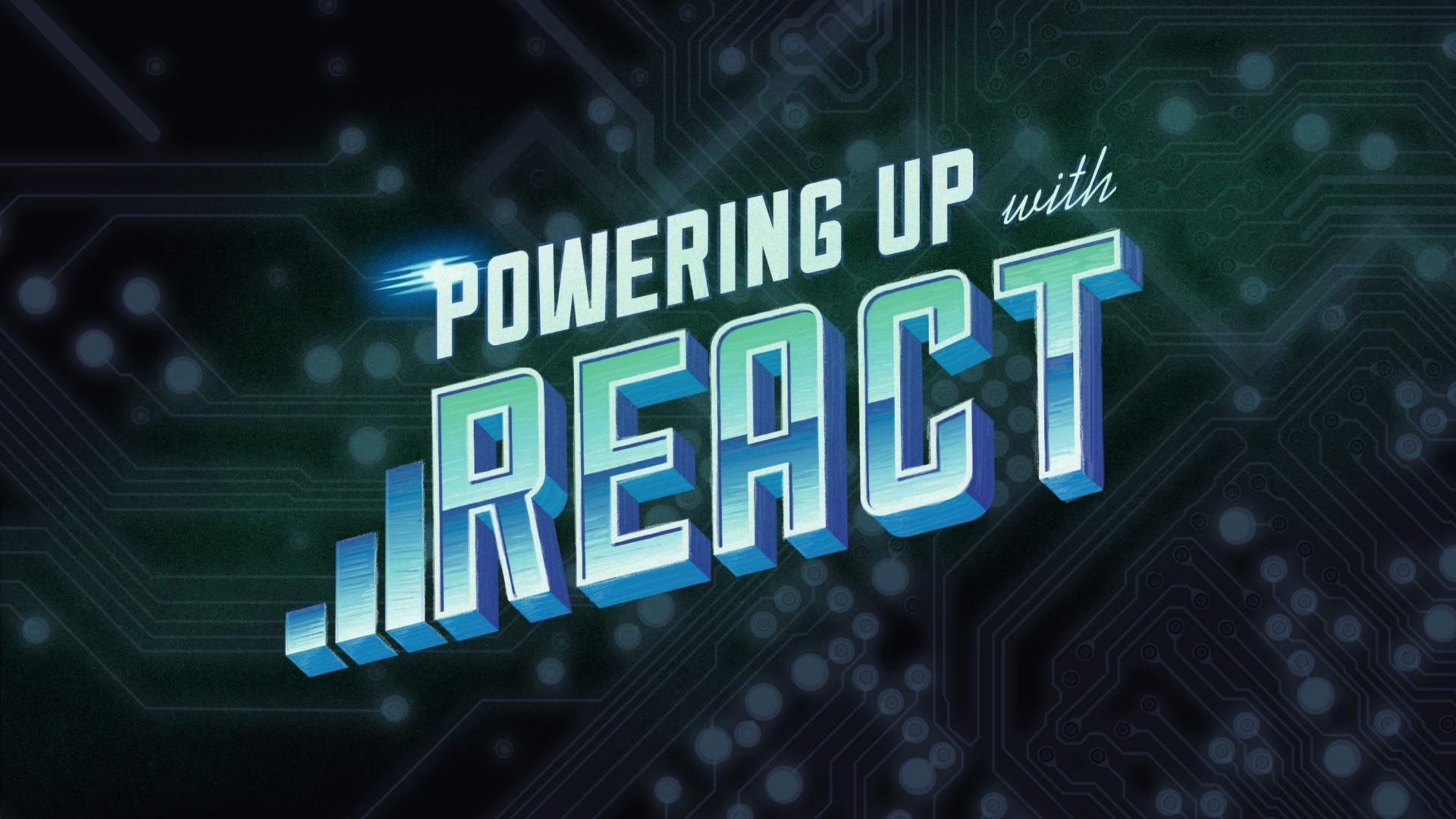
Quick Recap

We use React's event system to capture user input, including form submissions and button clicks.

Refs allow us to reference DOM elements in our code after the component has been rendered.

Parent components can pass callback functions as props to child components to allow two-way communication.

Synthetic events are a cross-browser wrapper around the browser's native event.


POWERING UP with
REACT