INTRODUCTION TO REACT

HOW IT STARTED…

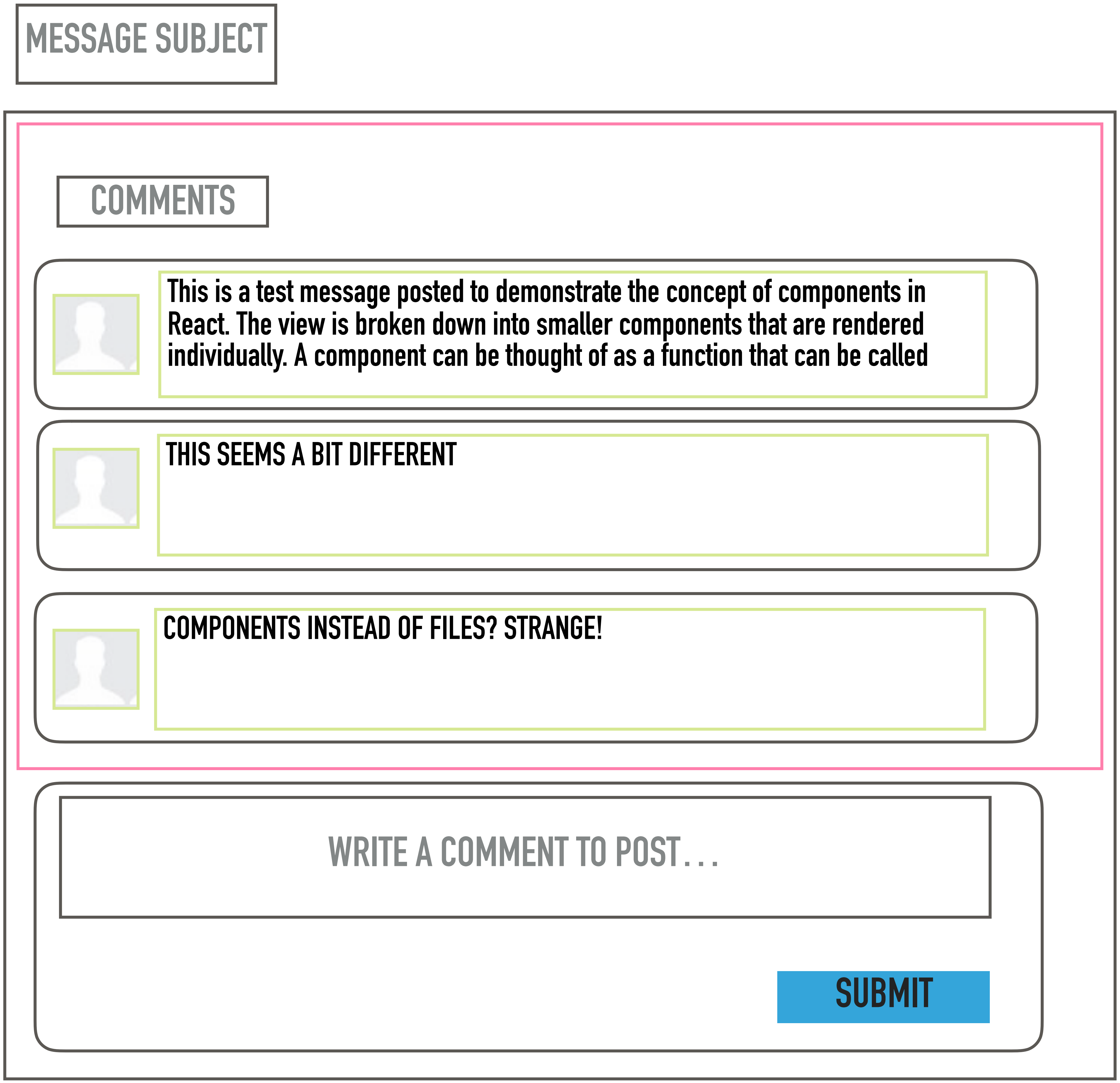
WHAT IS REACT ??

“A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES”

REACT GIVES YOU...

- Powerful Components
- Unidirectional Data Flow
- Explicit Mutation of the view
- A lightweight Virtual DOM

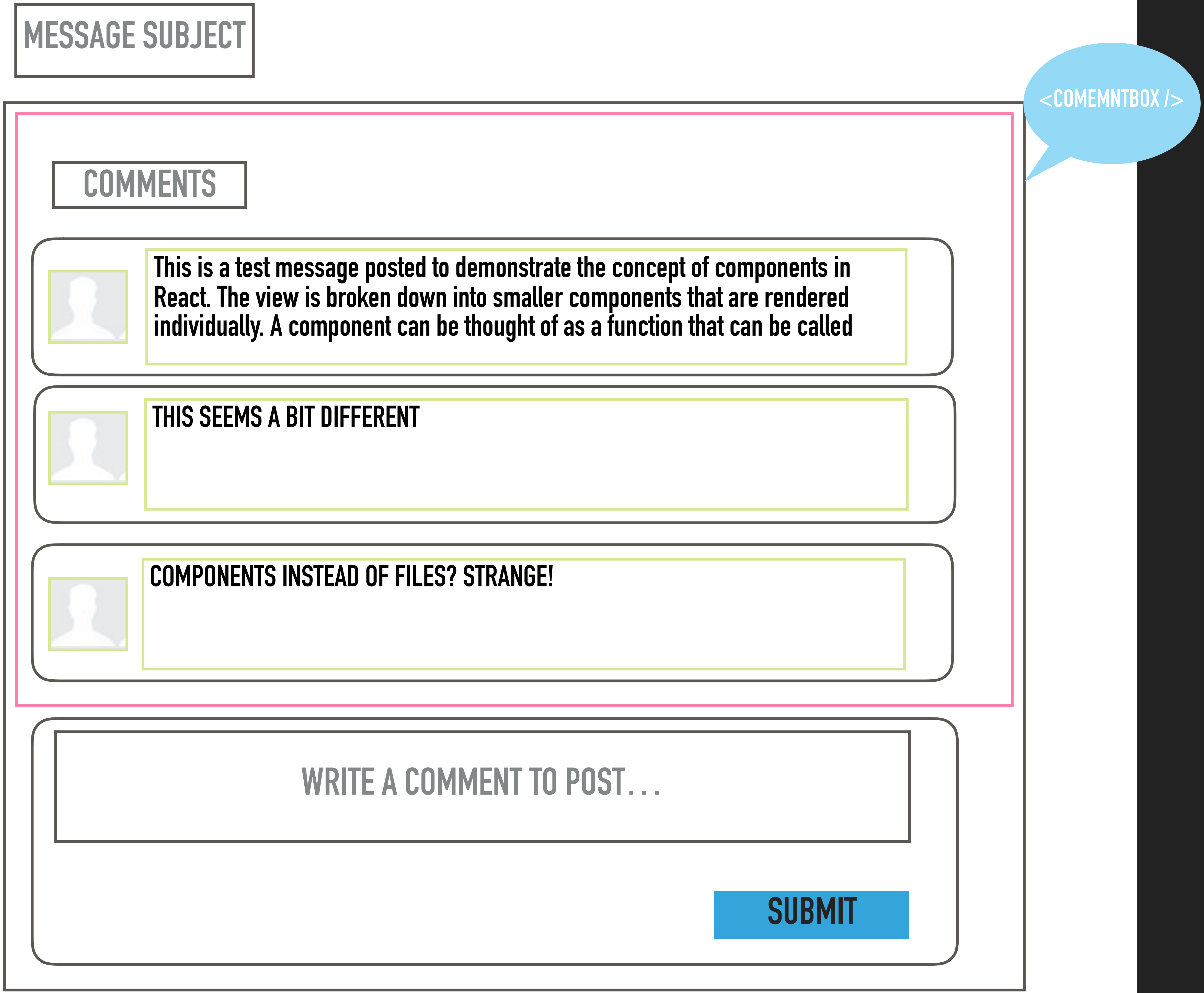
BUILD COMPONENTS, NOT TEMPLATES.



This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called

THIS SEEMS A BIT DIFFERENT

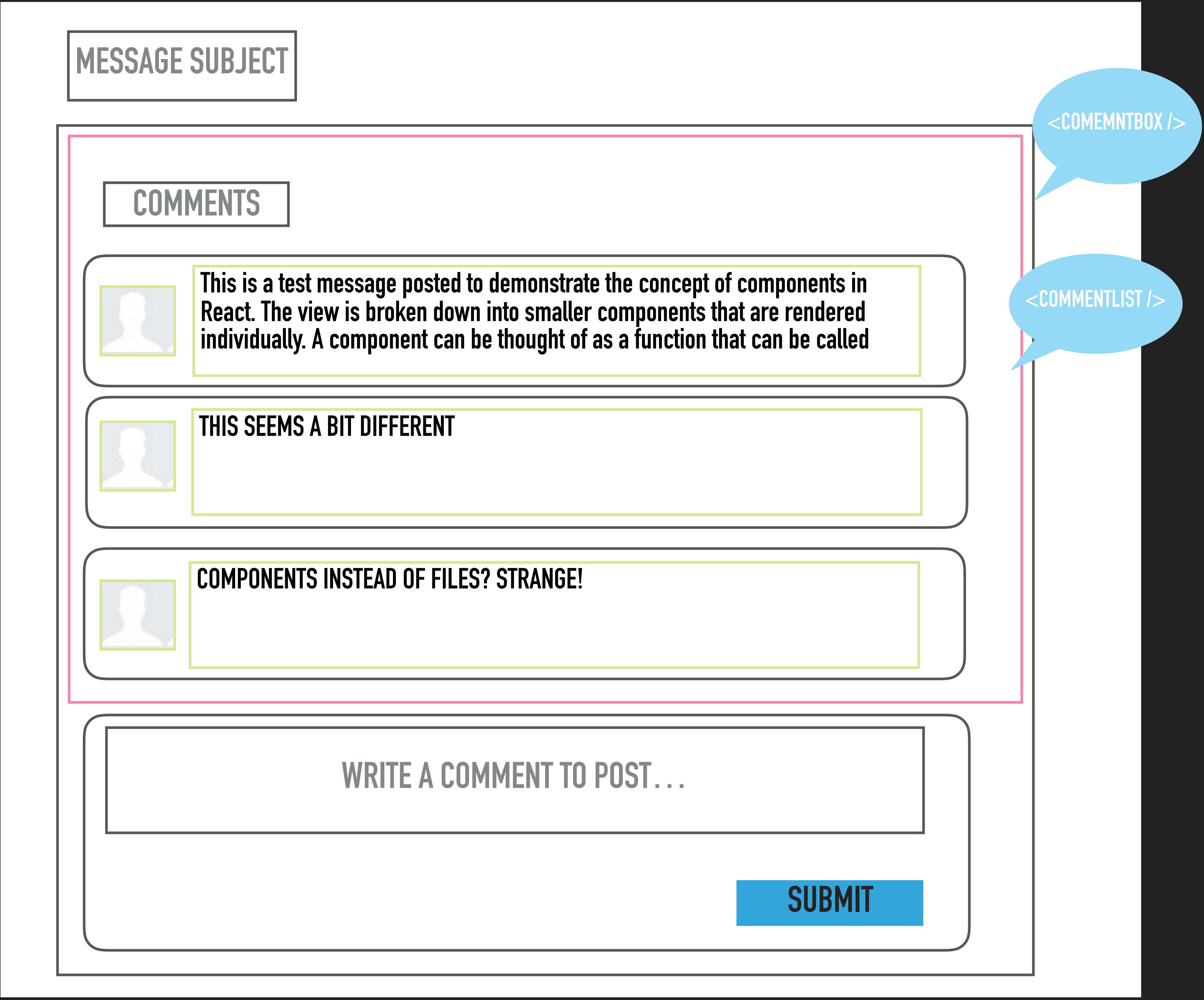
COMPONENTS INSTEAD OF FILES? STRANGE!



This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called

THIS SEEMS A BIT DIFFERENT

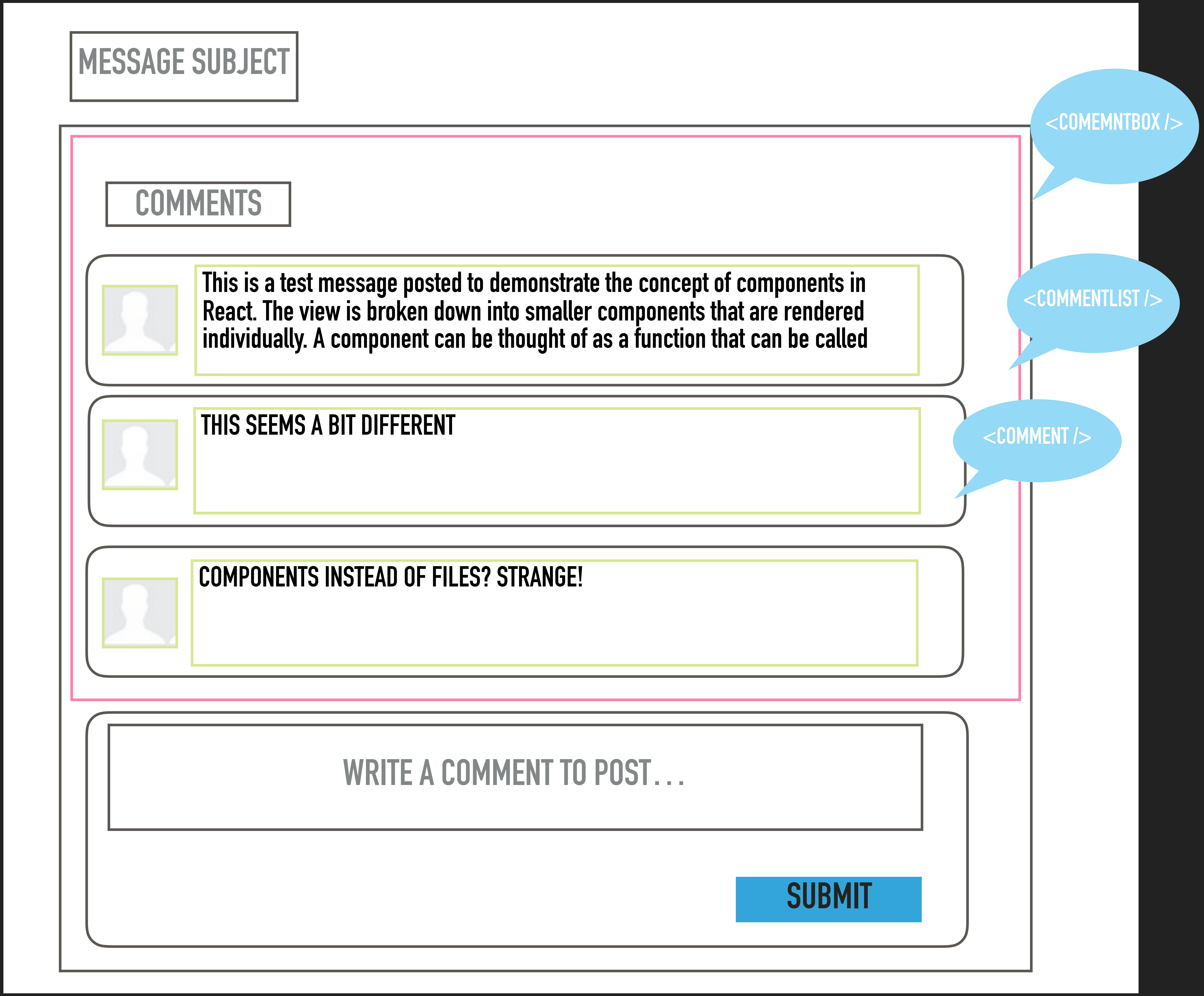
COMPONENTS INSTEAD OF FILES? STRANGE!



This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called

THIS SEEMS A BIT DIFFERENT

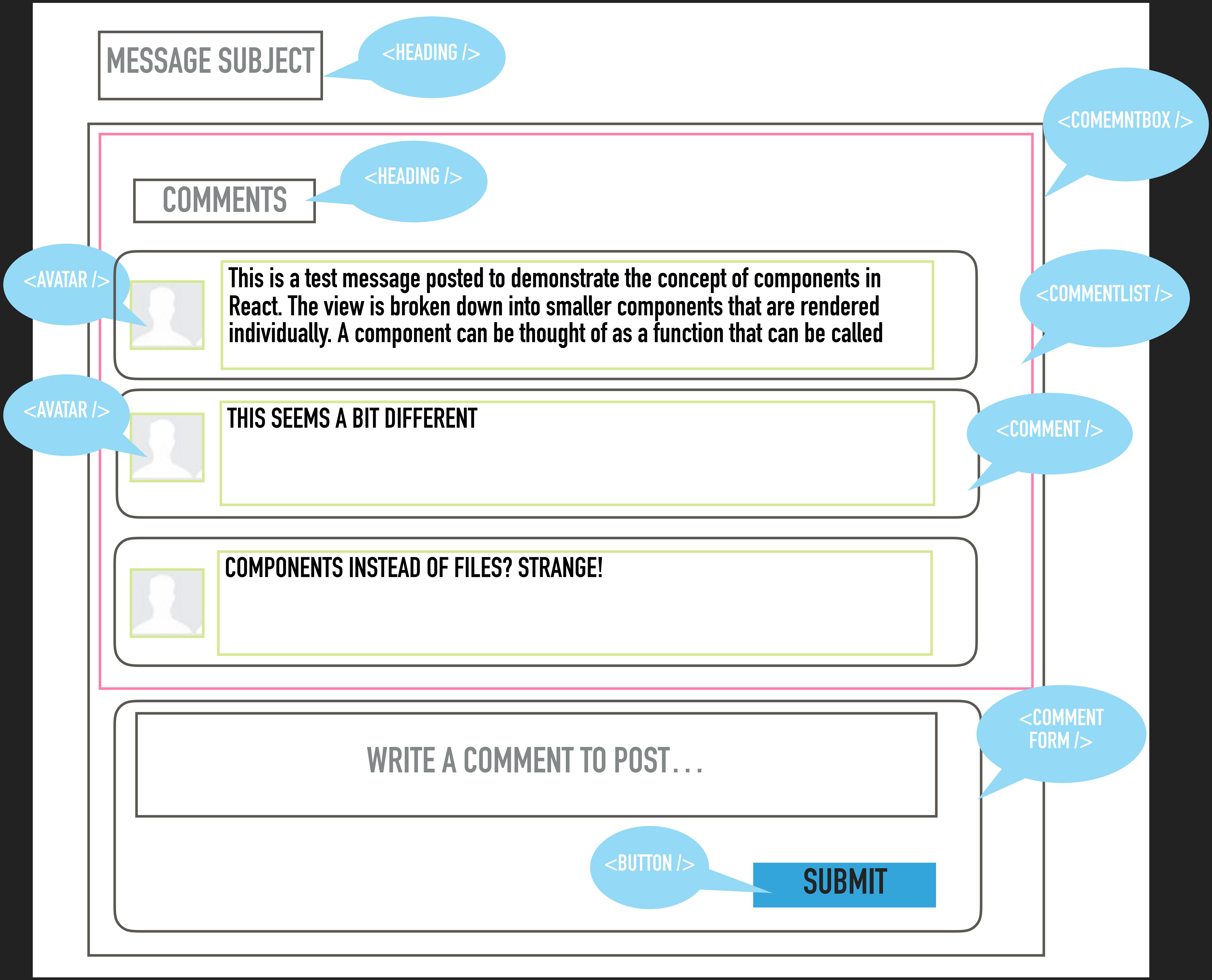
COMPONENTS INSTEAD OF FILES? STRANGE!



This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called

THIS SEEMS A BIT DIFFERENT

COMPONENTS INSTEAD OF FILES? STRANGE!



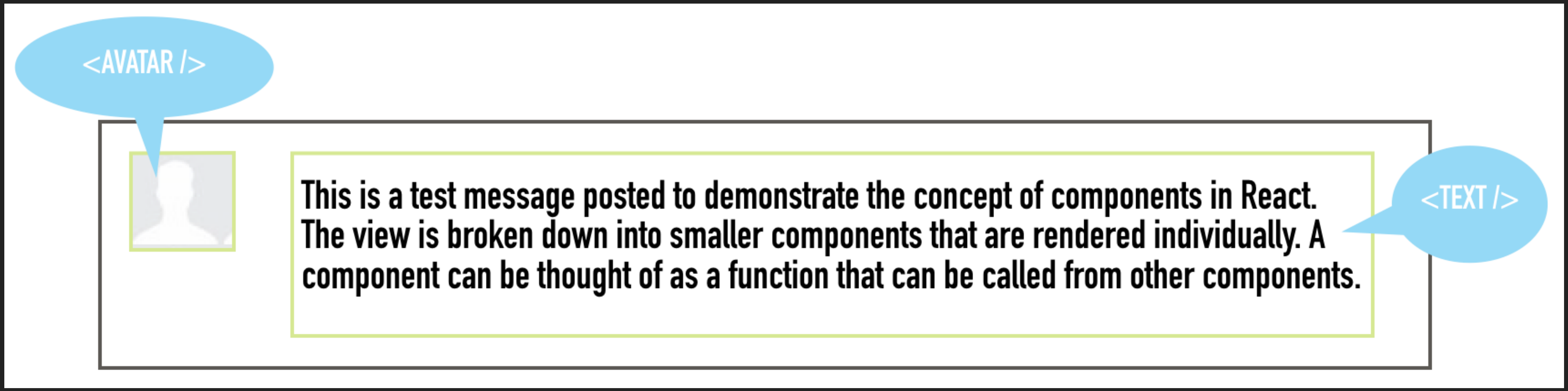
This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called

THIS SEEMS A BIT DIFFERENT

COMPONENTS INSTEAD OF FILES? STRANGE!

WRITE A COMMENT TO POST…

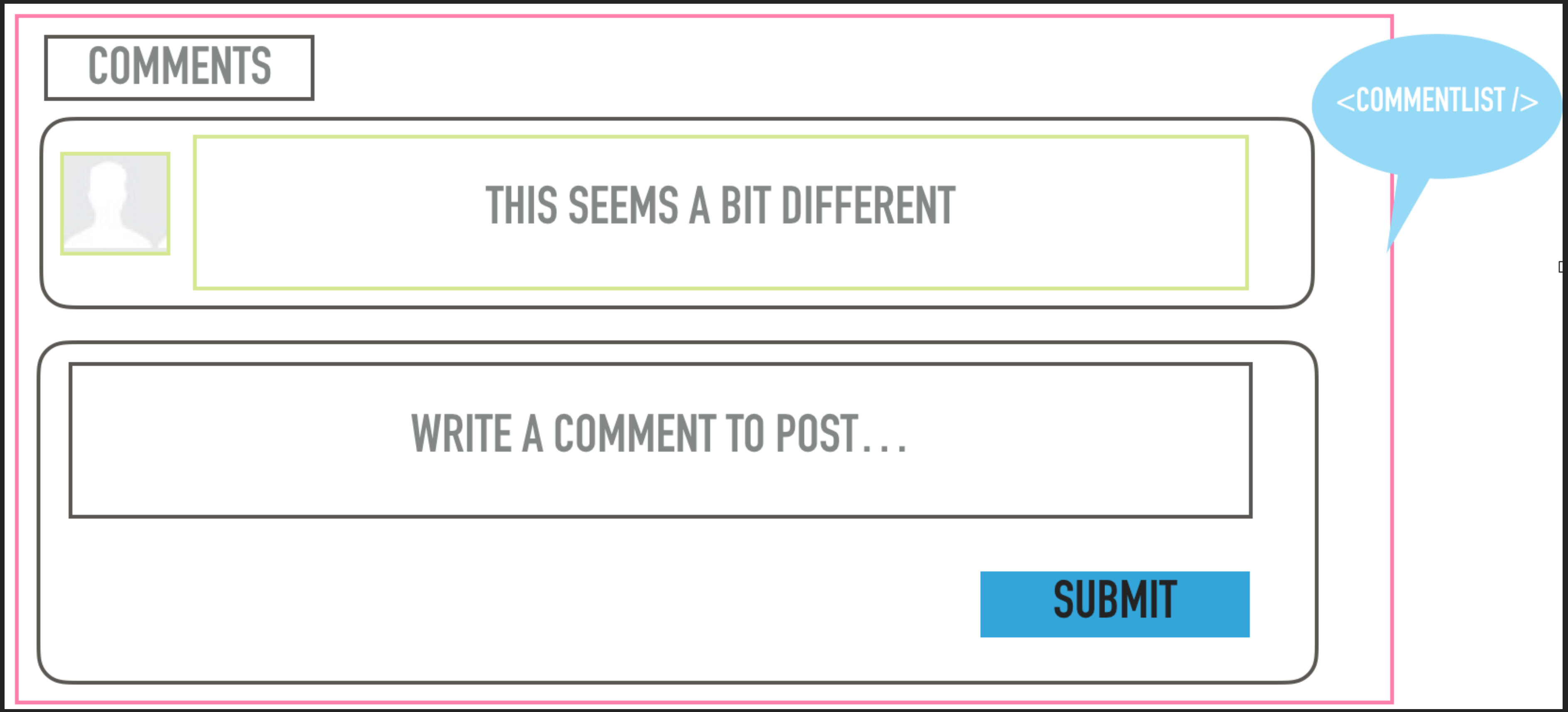
PROPS - UNIDIRECTIONAL DATA FLOW



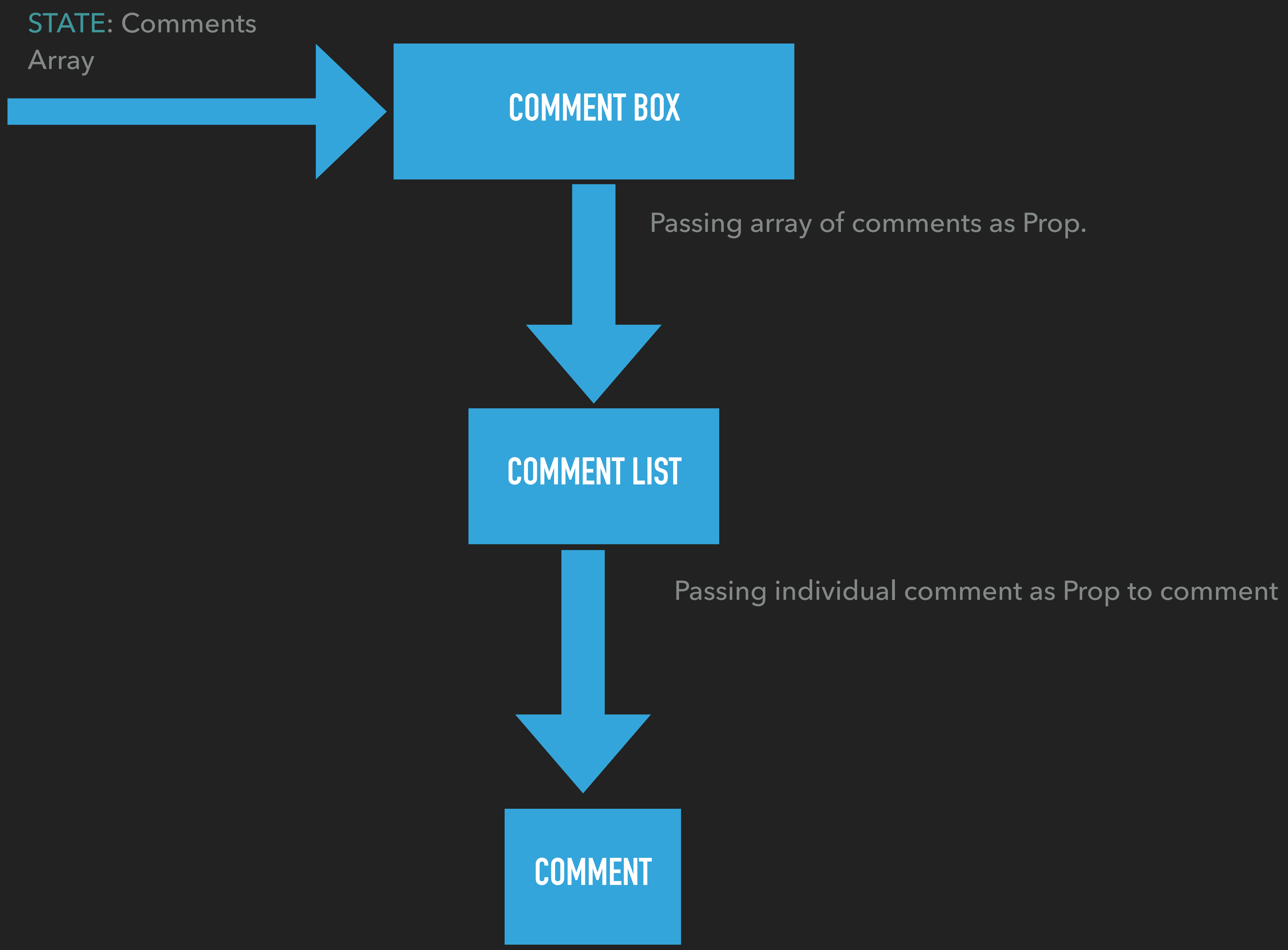
This is a test message posted to demonstrate the concept of components in React. The view is broken down into smaller components that are rendered individually. A component can be thought of as a function that can be called from other components.

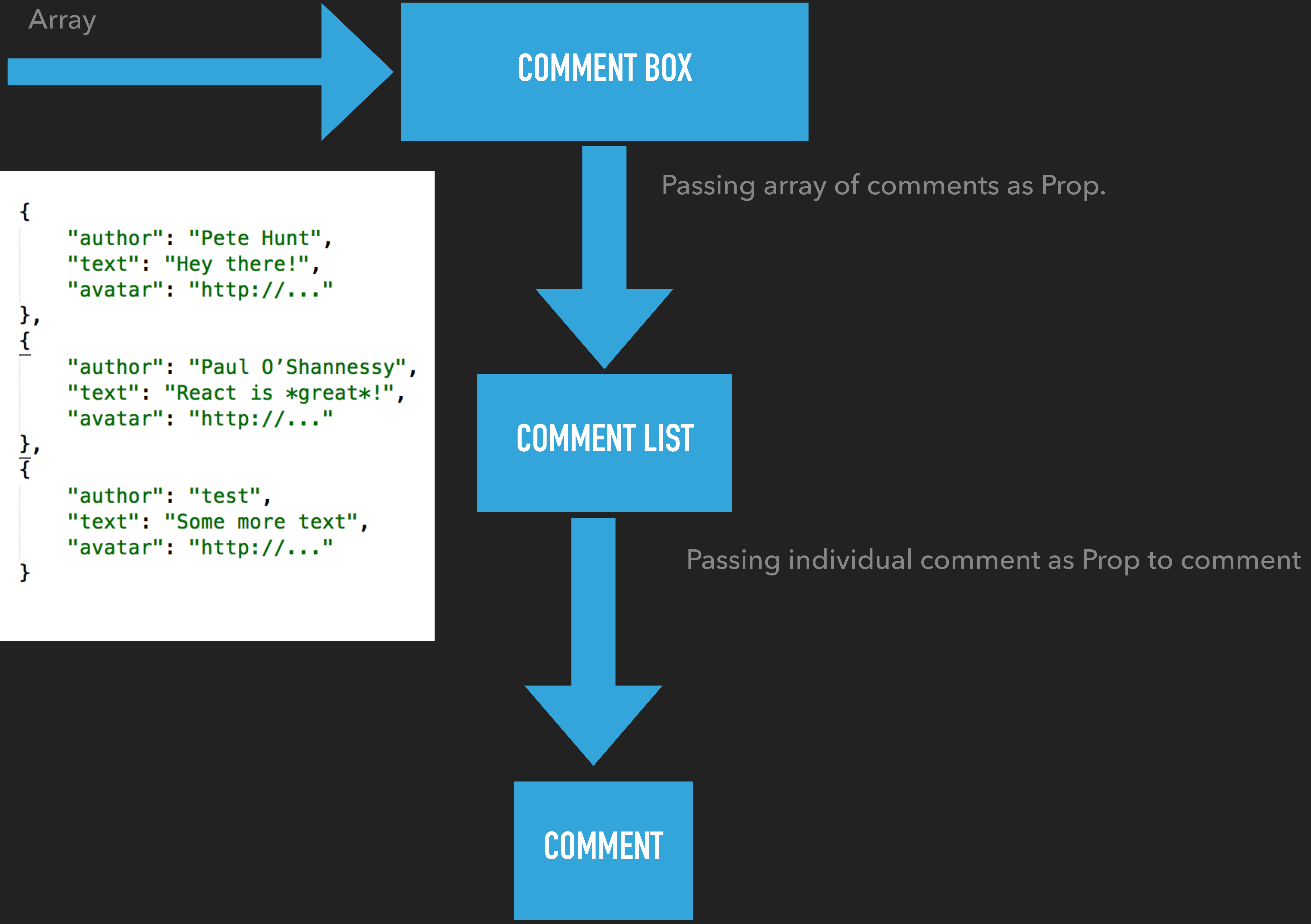
- Immutable
- Idempotent
- Data flows through the components via props.
- E.g. Avatar URL, Comment body.

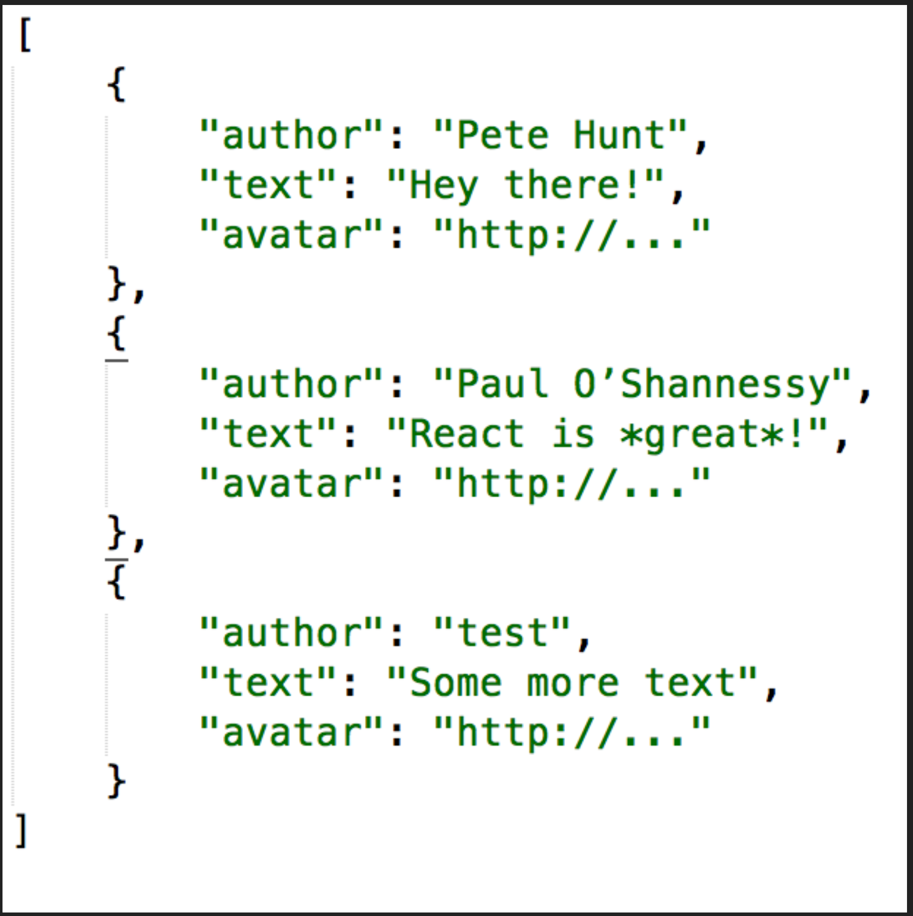
STATE - EXPLICIT MUTATION

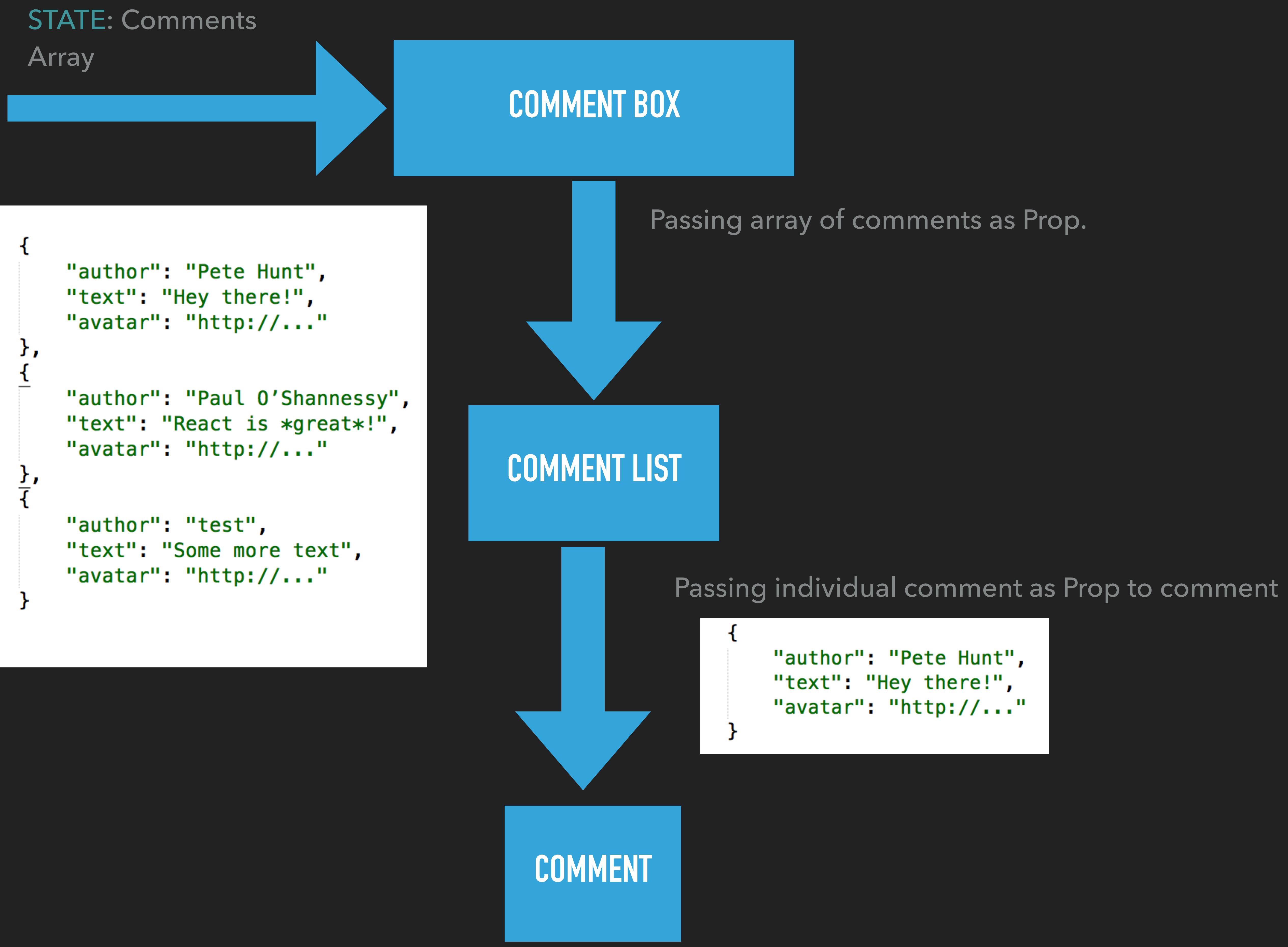


- Data that can be modified by user input or ajax call or time.
- When the state is changed, the component subtree is re-rendered.
- E.g. : Array of comments in CommentBox component.


STATE: Comments Array
COMMENT BOX
Passing array of comments as Prop.
COMMENT LIST
Passing individual comment as Prop to comment
COMMENT


STATE: Comments Array
COMMENT BOX
Passing array of comments as Prop.
COMMENT LIST
Passing individual comment as Prop to comment
COMMENT


```
[
    {
        "author": "Pete Hunt",
        "text": "Hey there!",
        "avatar": "http://..."
    },
    {
        "author": "Paul O'Shannessy",
        "text": "React is *great*!",
        "avatar": "http://..."
    },
    {
        "author": "test",
        "text": "Some more text",
        "avatar": "http://..."
    }
]
```


STATE: Comments Array
COMMENT BOX
Passing array of comments as Prop.
COMMENT LIST
Passing individual comment as Prop to comment
COMMENT

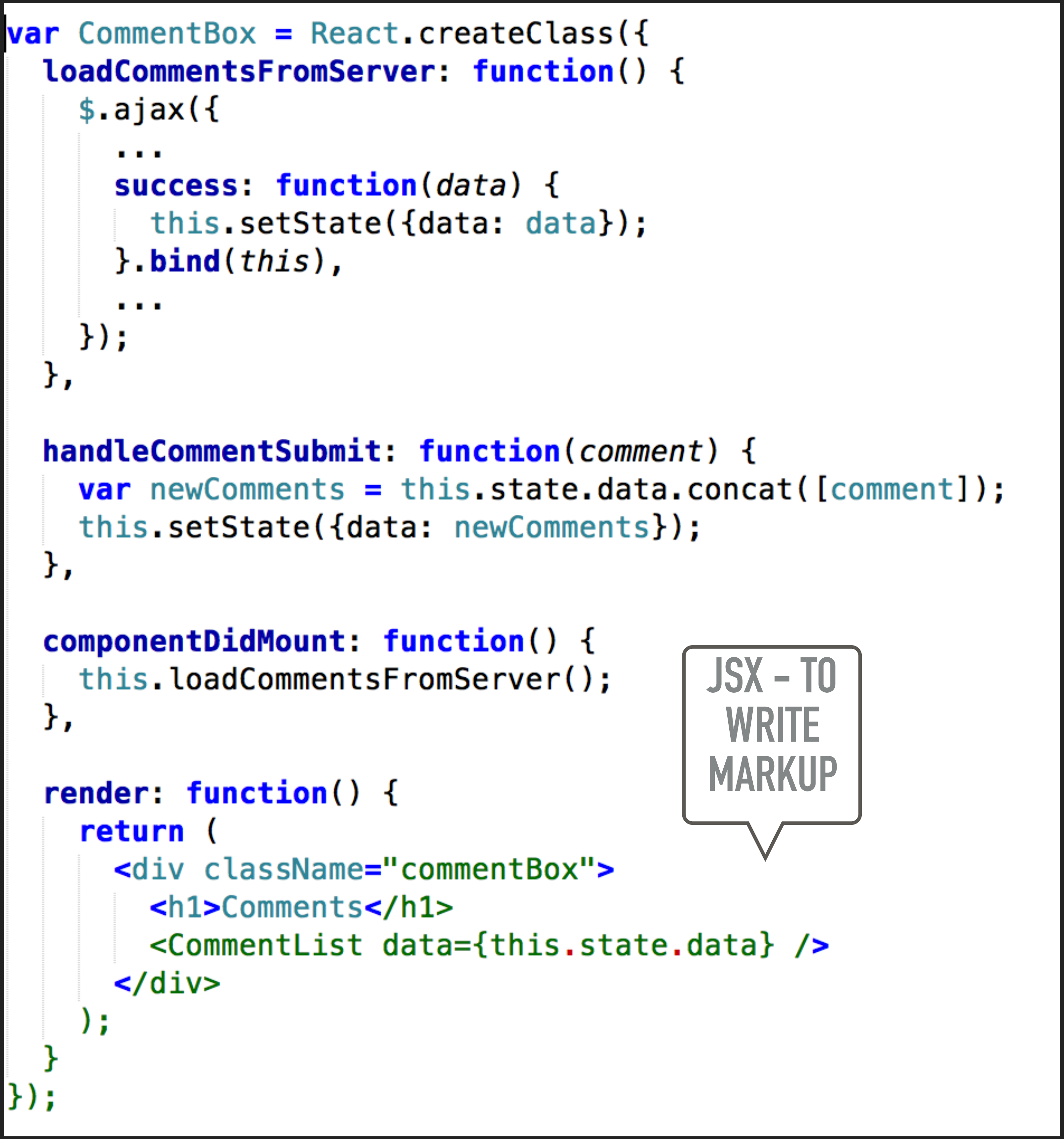
```
[
    {
        "author": "Pete Hunt",
        "text": "Hey there!",
        "avatar": "http://..."
    },
    {
        "author": "Paul O'Shannessy",
        "text": "React is *great*!",
        "avatar": "http://..."
    },
    {
        "author": "test",
        "text": "Some more text",
        "avatar": "http://..."
    }
]
```

```
{
    "author": "Pete Hunt",
    "text": "Hey there!",
    "avatar": "http://..."
}
```

THE RENDER METHOD

- Examines the props and the state of its component and returns a single element.
- Called on initial render and upon state change.

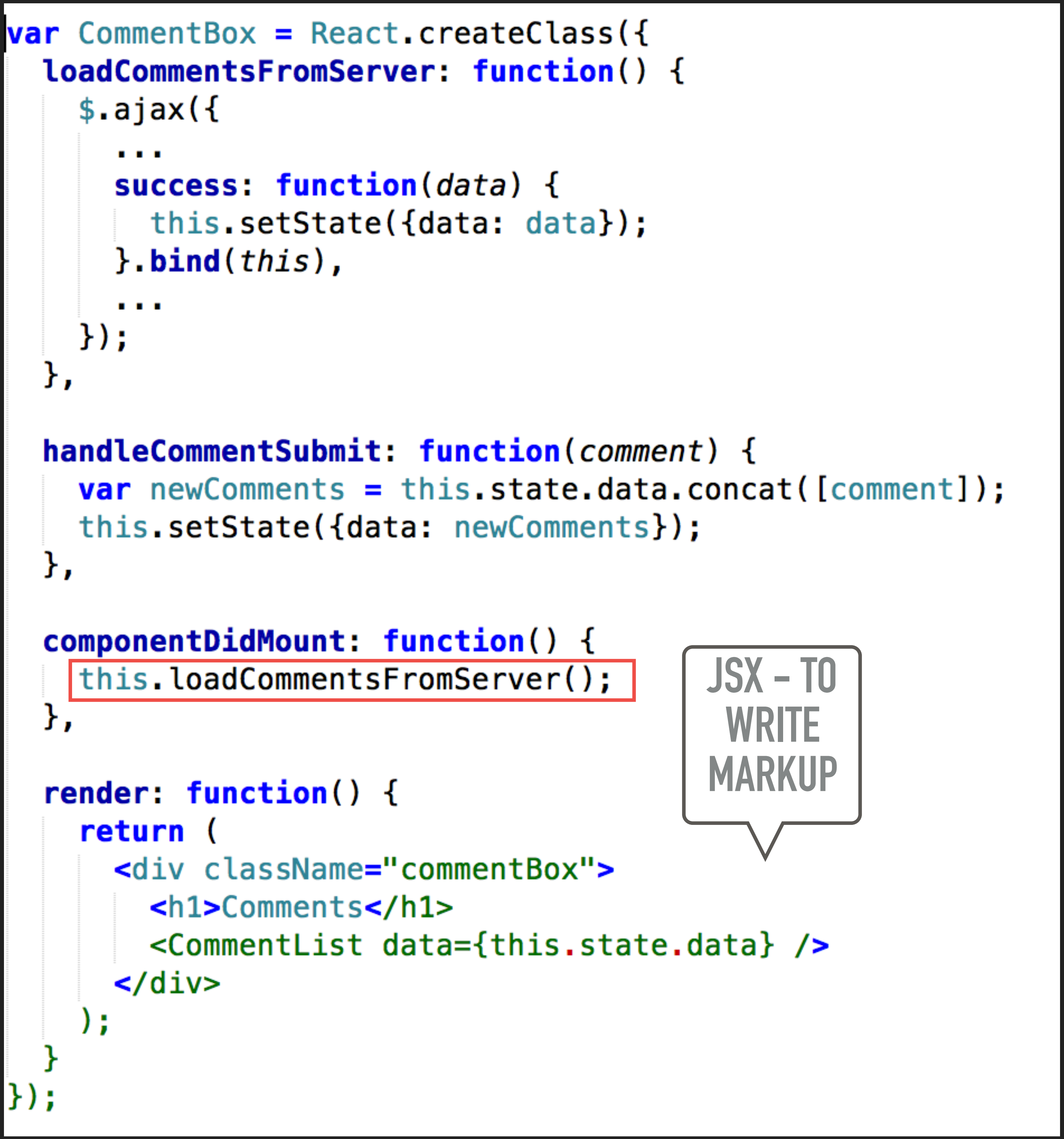
COMMENTBOX



```
var CommentBox = React.createClass({
  loadCommentsFromServer: function() {
    $.ajax({
      ...
      success: function(data) {
        this.setState({data: data});
      }.bind(this),
      ...
    });
  },

  handleCommentSubmit: function(comment) {
    var newComments = this.state.data.concat([comment]);
    this.setState({data: newComments});
  },

  componentDidMount: function() {
    this.loadCommentsFromServer();
  },

  render: function() {
    return (
      <div className="commentBox">
        <h1>Comments</h1>
        <CommentList data={this.state.data} />
      </div>
    );
  }
});
```

JSX - TO WRITE MARKUP

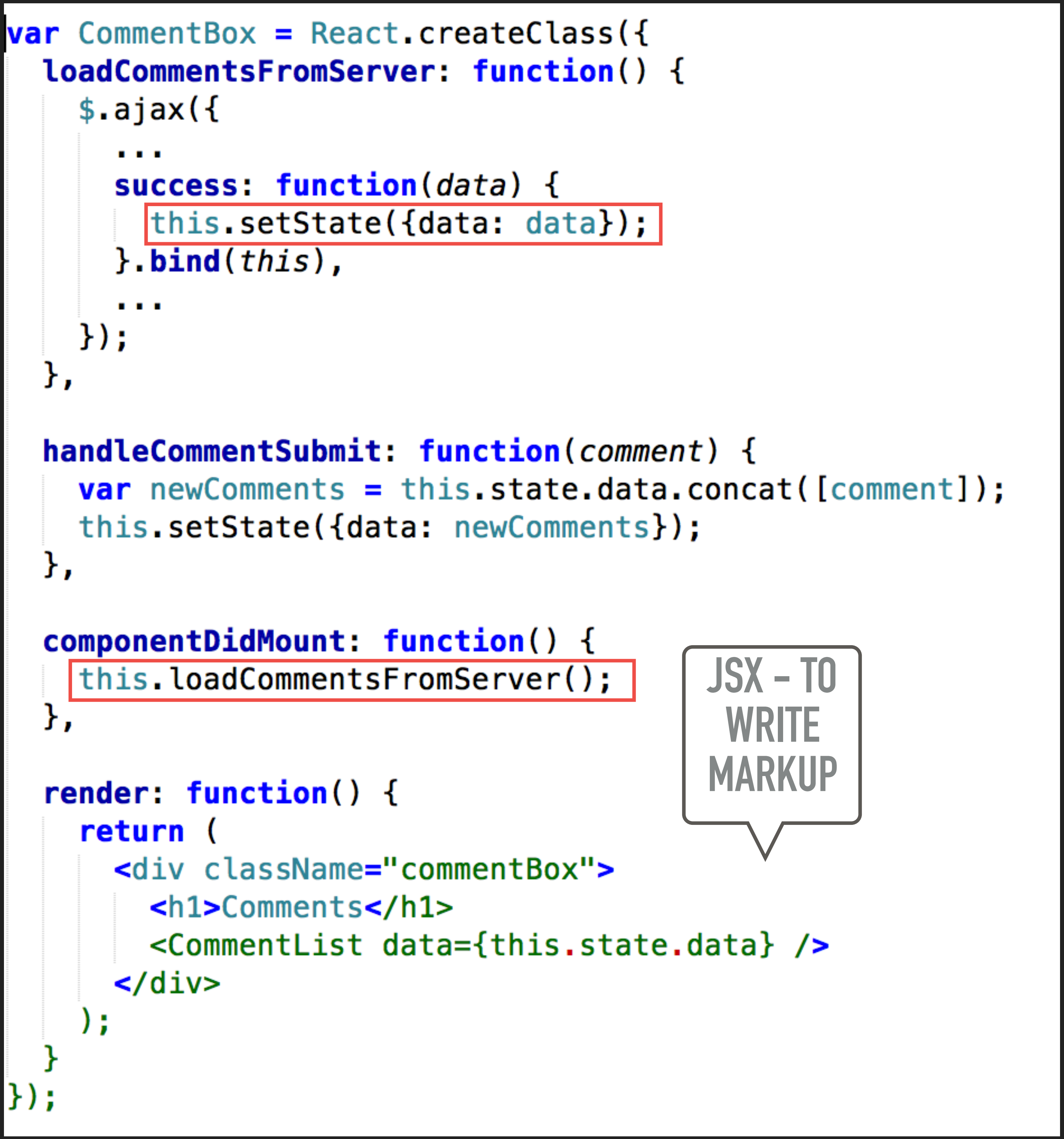
COMMENTBOX



```
var CommentBox = React.createClass({
  loadCommentsFromServer: function() {
    $.ajax({
      ...
      success: function(data) {
        this.setState({data: data});
      }.bind(this),
      ...
    });
  },

  handleCommentSubmit: function(comment) {
    var newComments = this.state.data.concat([comment]);
    this.setState({data: newComments});
  },

  componentDidMount: function() {
    this.loadCommentsFromServer();
  },

  render: function() {
    return (
      <div className="commentBox">
        <h1>Comments</h1>
        <CommentList data={this.state.data} />
      </div>
    );
  }
});
```

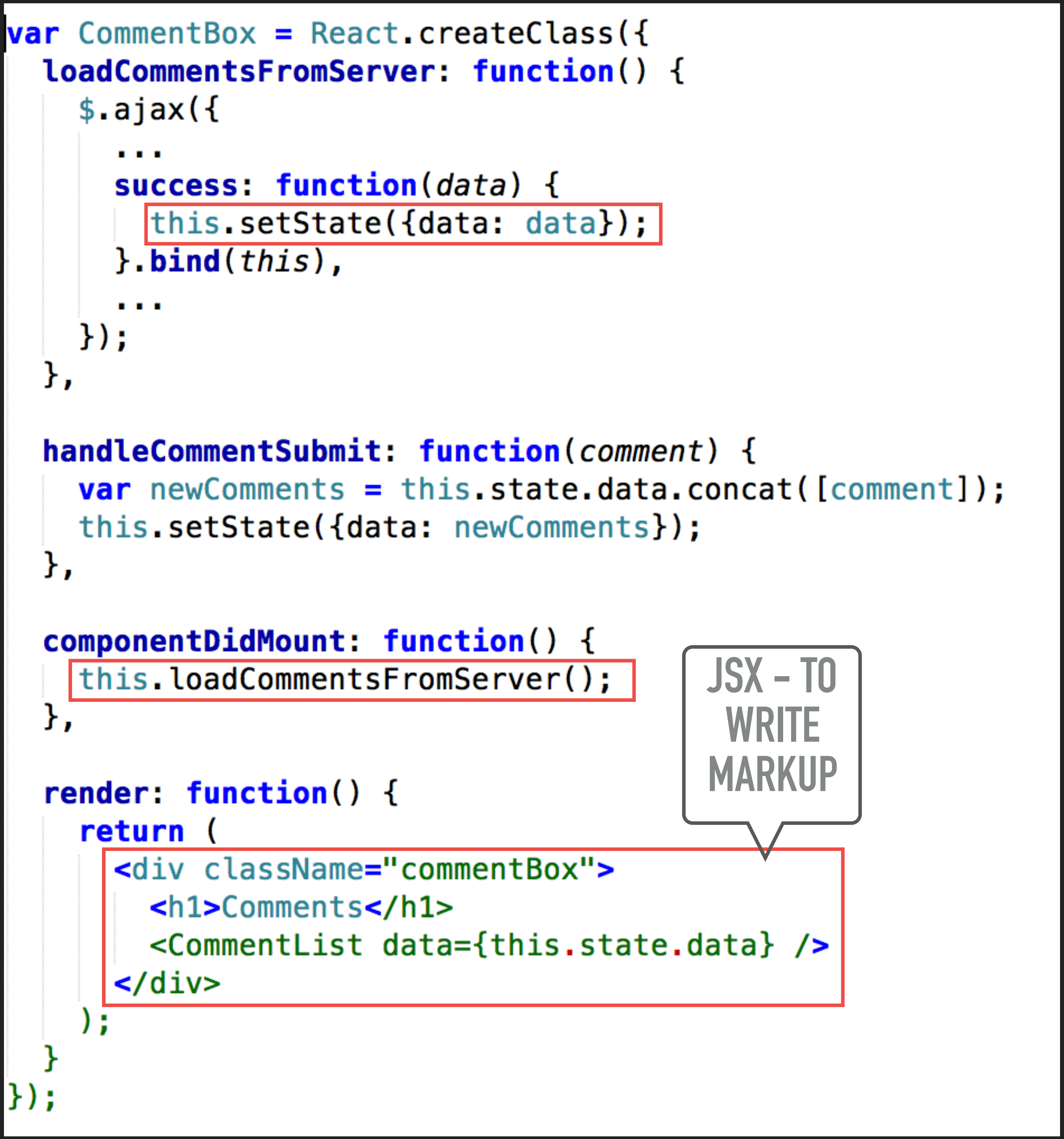
JSX - TO WRITE MARKUP

COMMENTBOX



```
var CommentBox = React.createClass({
  loadCommentsFromServer: function() {
    $.ajax({
      ...
      success: function(data) {
        this.setState({data: data});
      }.bind(this),
      ...
    });
  },

  handleCommentSubmit: function(comment) {
    var newComments = this.state.data.concat([comment]);
    this.setState({data: newComments});
  },

  componentDidMount: function() {
    this.loadCommentsFromServer();
  },

  render: function() {
    return (
      <div className="commentBox">
        <h1>Comments</h1>
        <CommentList data={this.state.data} />
      </div>
    );
  }
});
```

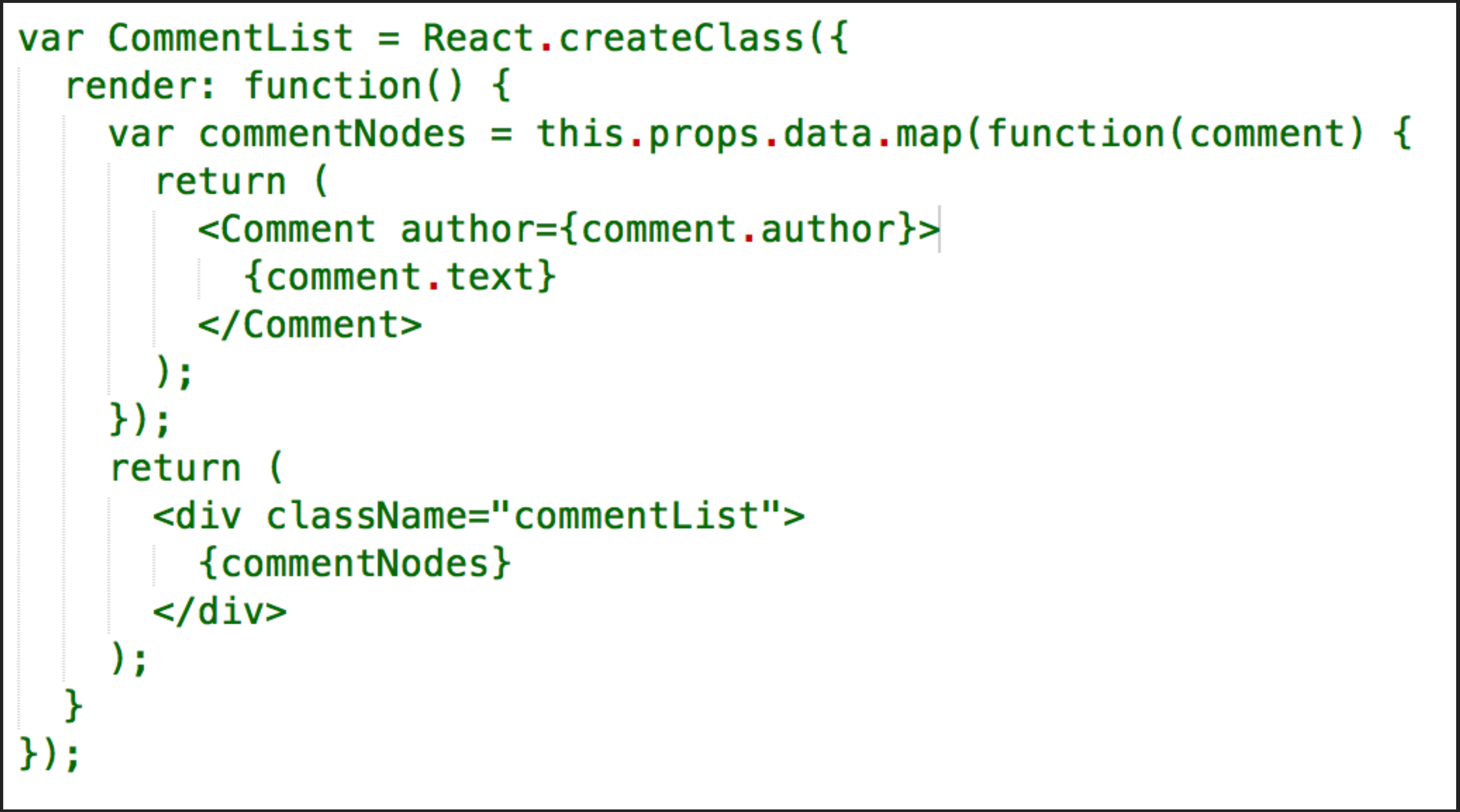
COMMENTBOX



```
var CommentBox = React.createClass({
  loadCommentsFromServer: function() {
    $.ajax({
      ...
      success: function(data) {
        this.setState({data: data});
      }.bind(this),
      ...
    });
  },

  handleCommentSubmit: function(comment) {
    var newComments = this.state.data.concat([comment]);
    this.setState({data: newComments});
  },

  componentDidMount: function() {
    this.loadCommentsFromServer();
  },

  render: function() {
    return (
      <div className="commentBox">
        <h1>Comments</h1>
        <CommentList data={this.state.data} />
      </div>
    );
  }
});
```

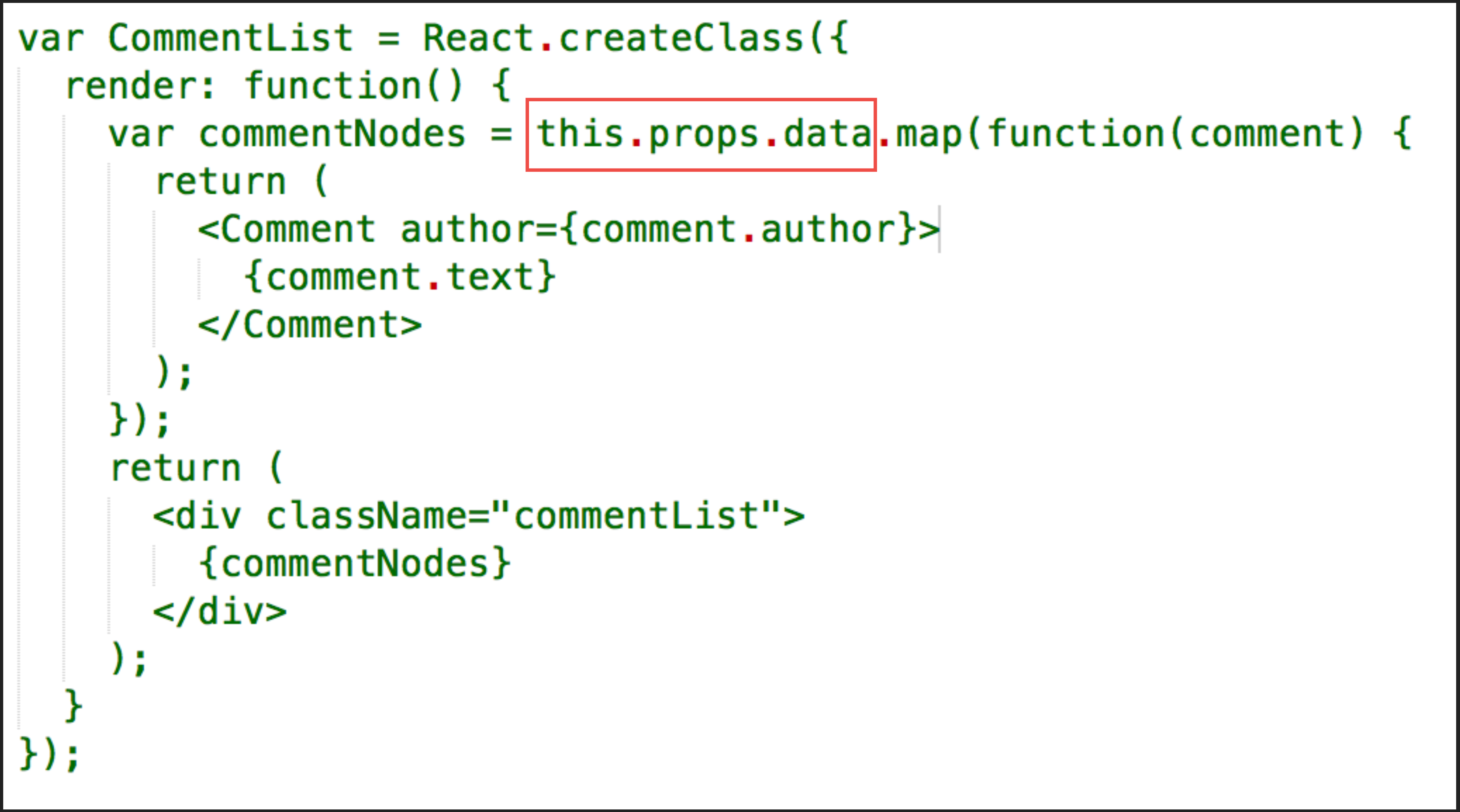
JSX - TO WRITE MARKUP

COMMENT LIST



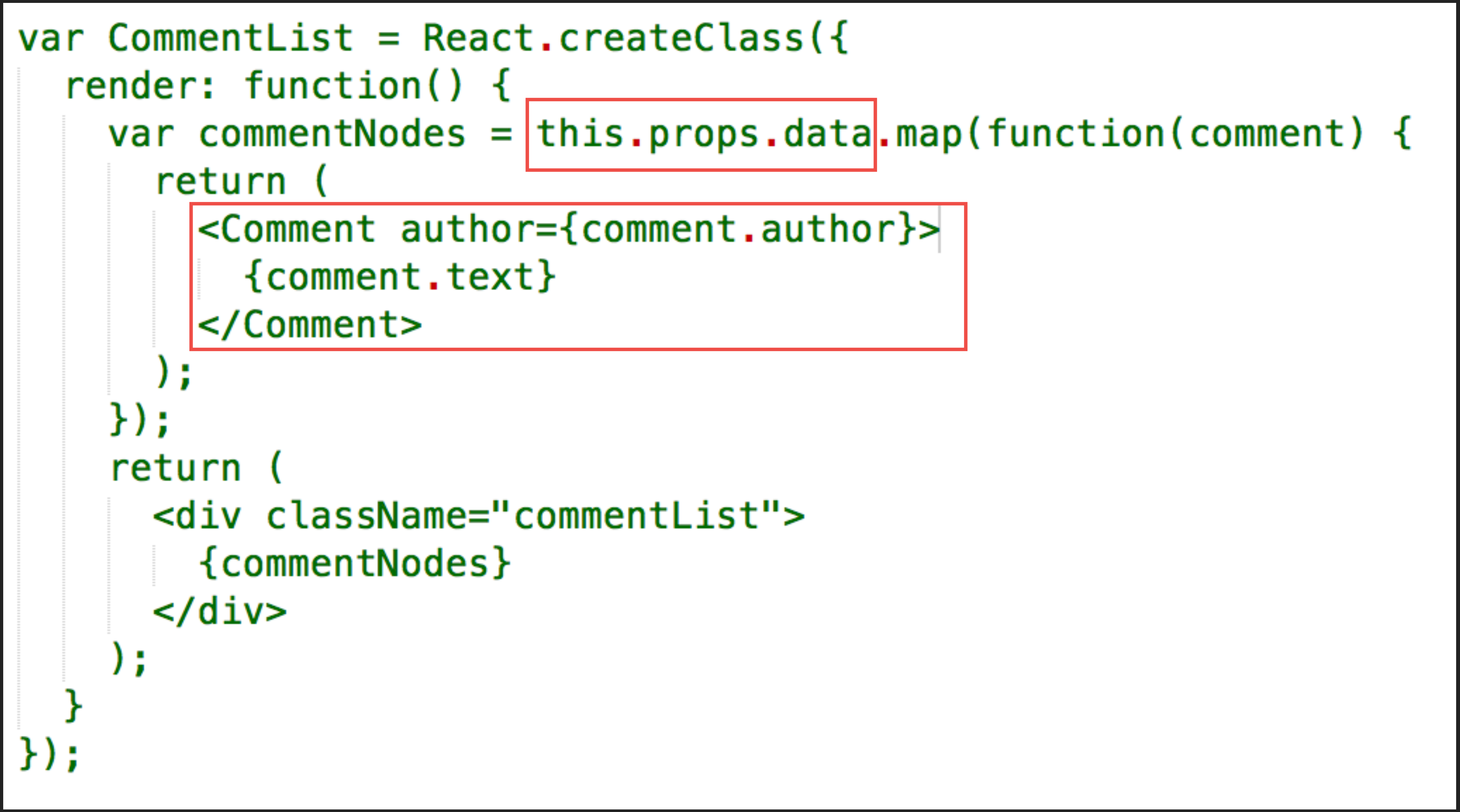
```
var CommentList = React.createClass({
  render: function() {
    var commentNodes = this.props.data.map(function(comment) {
      return (
        <Comment author={comment.author}>
          {comment.text}
        </Comment>
      );
    });
    return (
      <div className="commentList">
        {commentNodes}
      </div>
    );
  }
});
```

COMMENT LIST



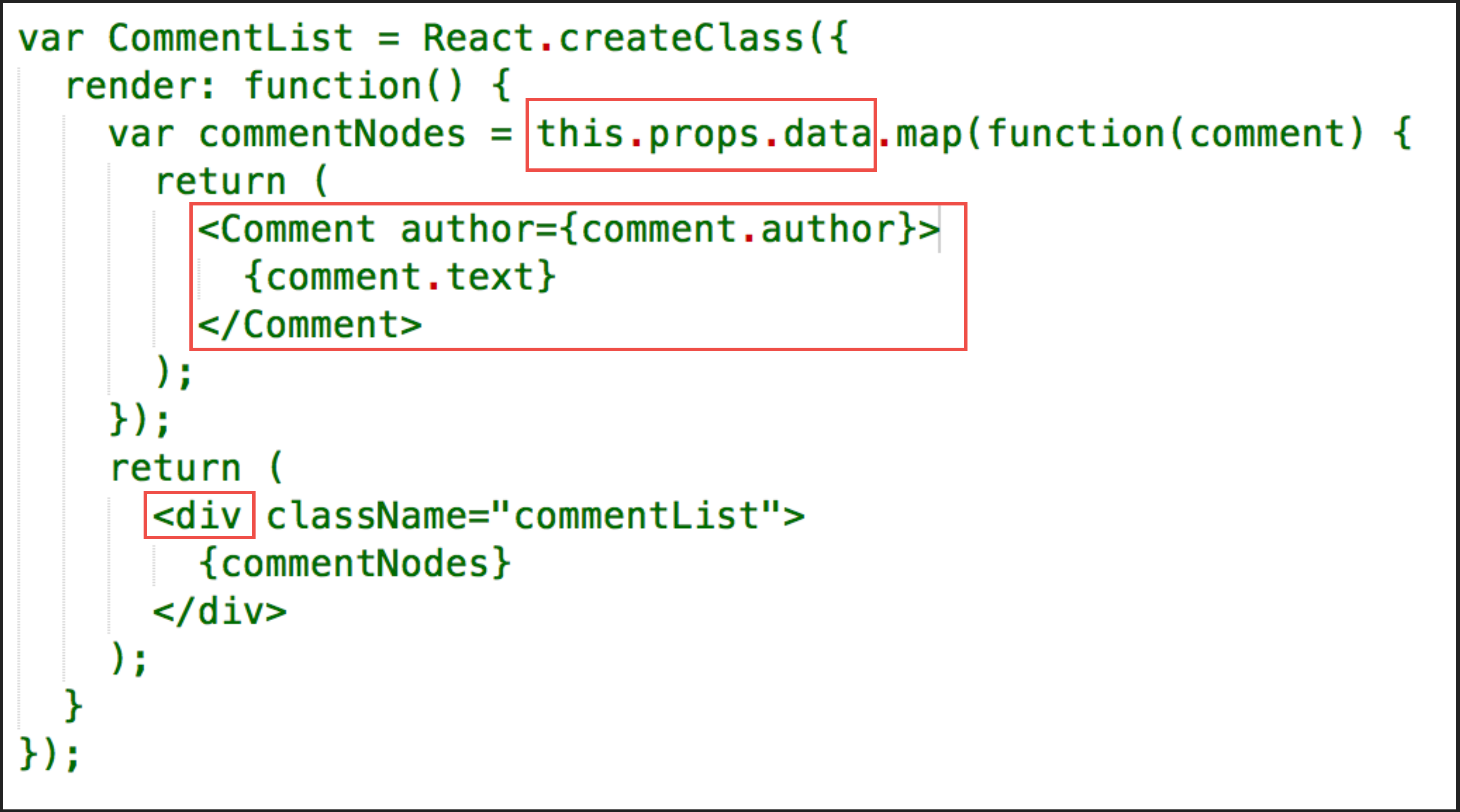
```
var CommentList = React.createClass({
  render: function() {
    var commentNodes = this.props.data.map(function(comment) {
      return (
        <Comment author={comment.author}>
          {comment.text}
        </Comment>
      );
    });
    return (
      <div className="commentList">
        {commentNodes}
      </div>
    );
  }
});
```

COMMENT LIST



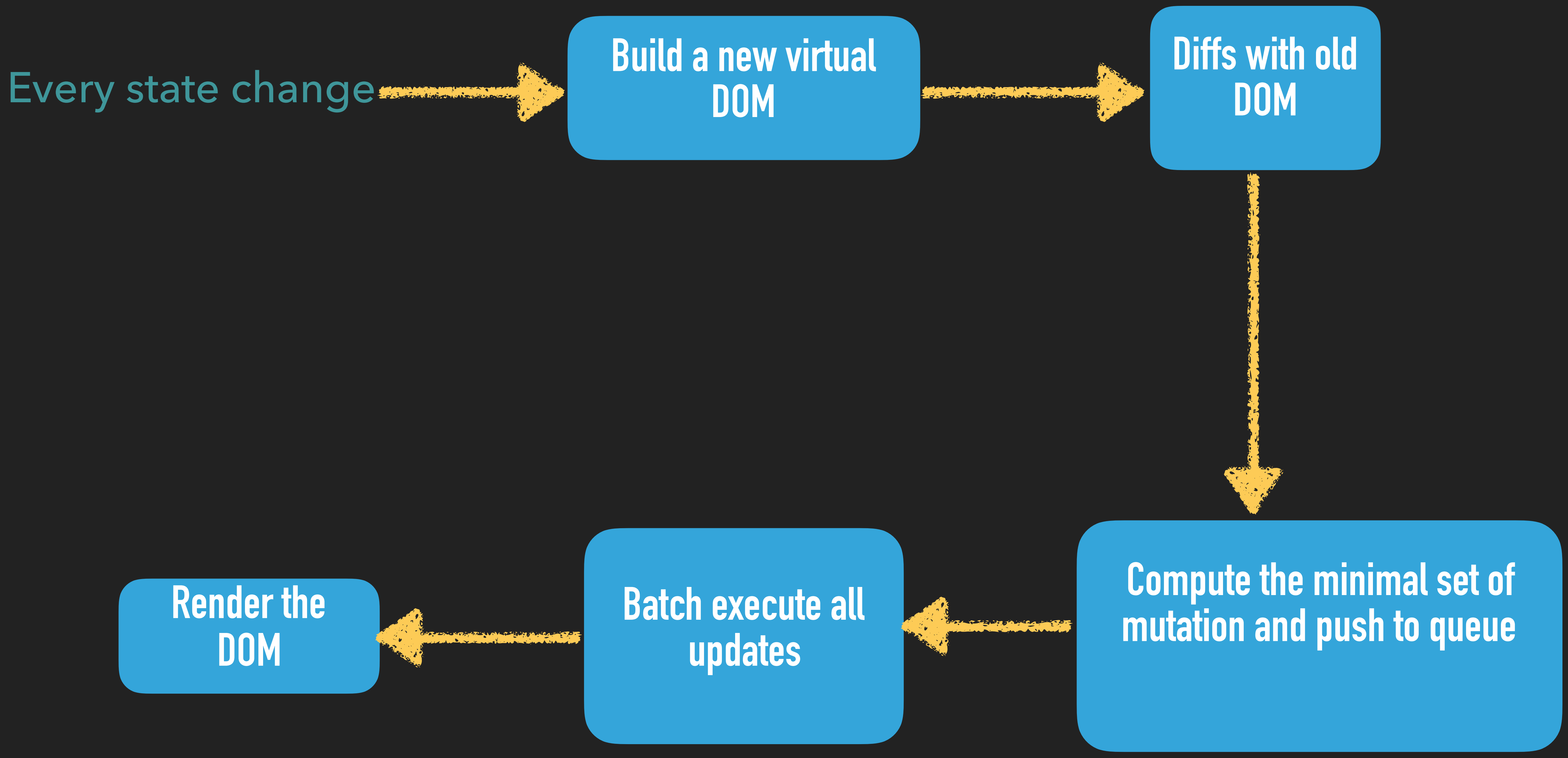
```
var CommentList = React.createClass({
  render: function() {
    var commentNodes = this.props.data.map(function(comment) {
      return (
        <Comment author={comment.author}>
          {comment.text}
        </Comment>
      );
    });
    return (
      <div className="commentList">
        {commentNodes}
      </div>
    );
  }
});
```

COMMENT LIST



```
var CommentList = React.createClass({
  render: function() {
    var commentNodes = this.props.data.map(function(comment) {
      return (
        <Comment author={comment.author}>
          {comment.text}
        </Comment>
      );
    });
    return (
      <div className="commentList">
        {commentNodes}
      </div>
    );
  }
});
```

VIRTUAL DOM


Every state change
Build a new virtual DOM
Diffs with old DOM
Compute the minimal set of mutation and push to queue
Batch execute all updates
Render the DOM

WHY REACT ??

REACT IS DIFFERENT

Exposes DOM specific methods

REACT-DOM

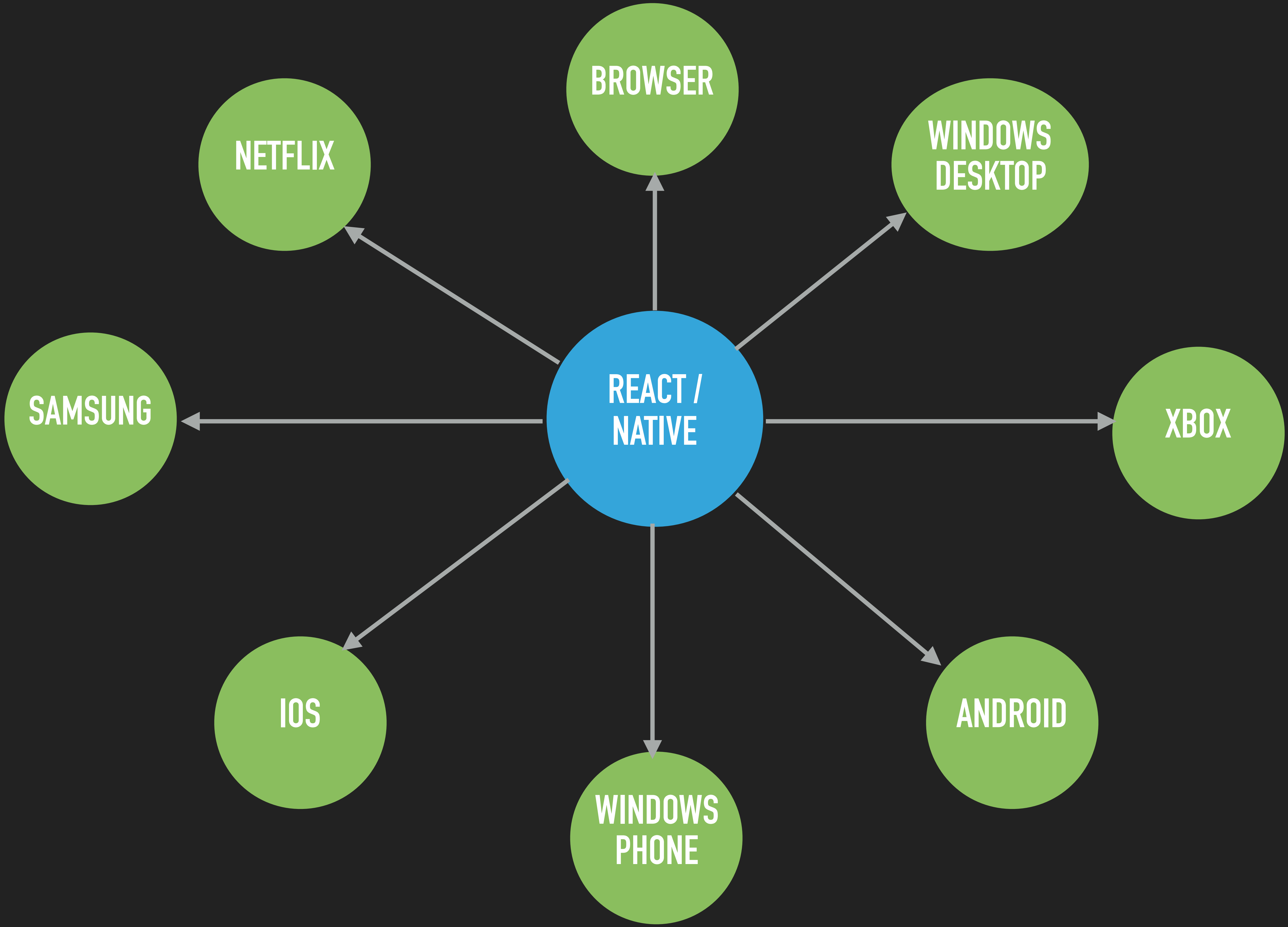
REACT

Core react. No assumptions about the View layer

A framework for building native apps using React

REACT-NATIVE

LEARN ONCE, WRITE EVERYWHERE



WHY REACT FOR DEVELOPERS?

- Shallow learning curve
- Great debugging tools
- Easy to reason about. Easier to think.
- Less time between writing the code and deploying
- Learn once, write everywhere
- Backend frameworks going the API way…

THANK YOU!

REFERENCES

- https://facebook.github.io/react/docs/getting-started.html
- https://gist.github.com/staltz/868e7e9bc2a7b8c1f754 - A good view on reactive programming.
- http://tonyfreed.com/blog/what_is_virtual_dom
- https://code.facebook.com/videos/1507312032855537/oscon-2014-react-s-architecture/ – Very old ppt on origins of react.
- http://tadeuzagallo.com/blog/react-native-bridge/