React

An introduction

`whoami`

- software engineer @ HackerOne
- apprentice of all trades (hardware, vms, backend, frontend)
- works mostly in CoffeeScript
- @jjoos / jan@deelstra.org

What is React?

```
curl https://github.com/facebook/react/blob/master/README.md
```

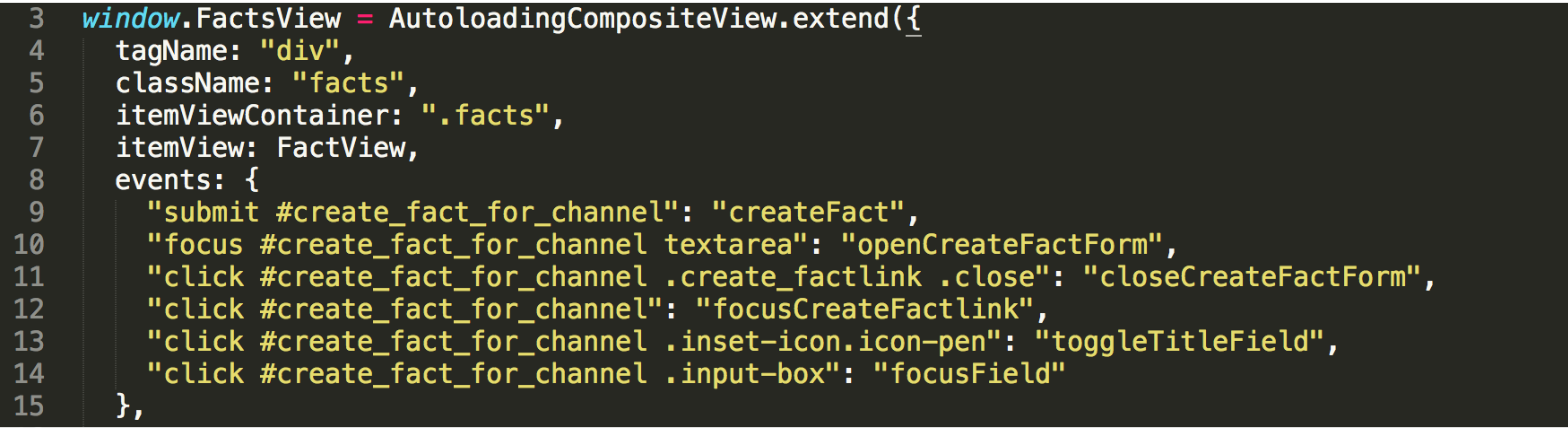
Interest in React

- xhp is not adopted
- released one year ago
- Facebook, Instagram, Airbnb, Khan Academy, Mozilla, Github, etc.
- New frameworks using React concepts: `mithril`, `elm`, `mercury` and `ractive`

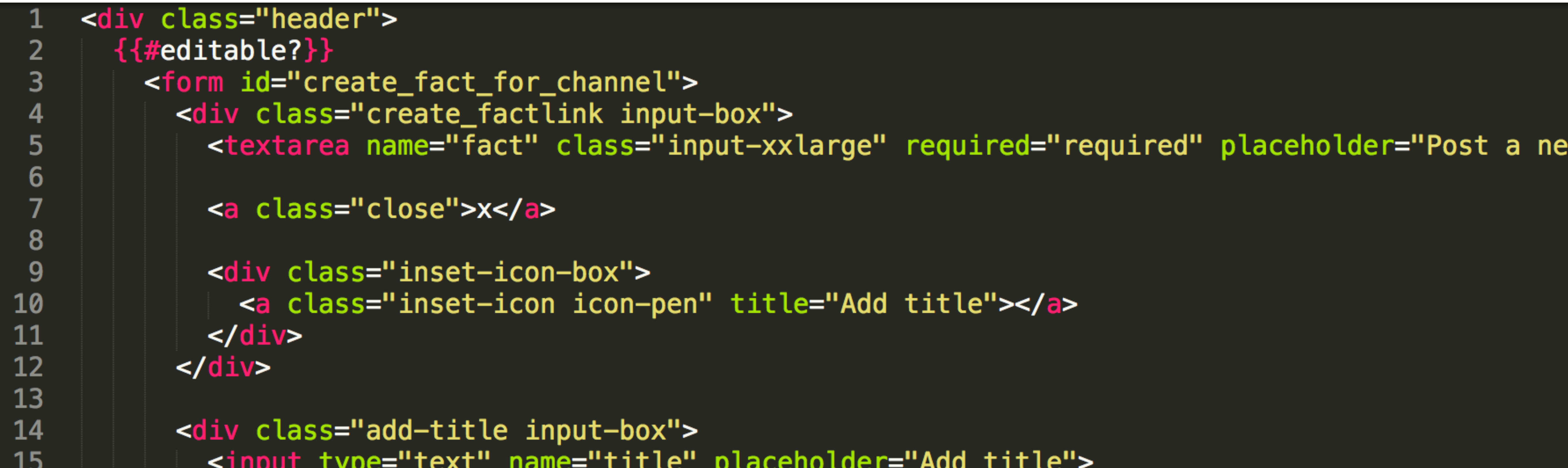
“I’ve seen React as the replacement for the views component of Angular.”

–Tyler Renelle

Display logic and templates

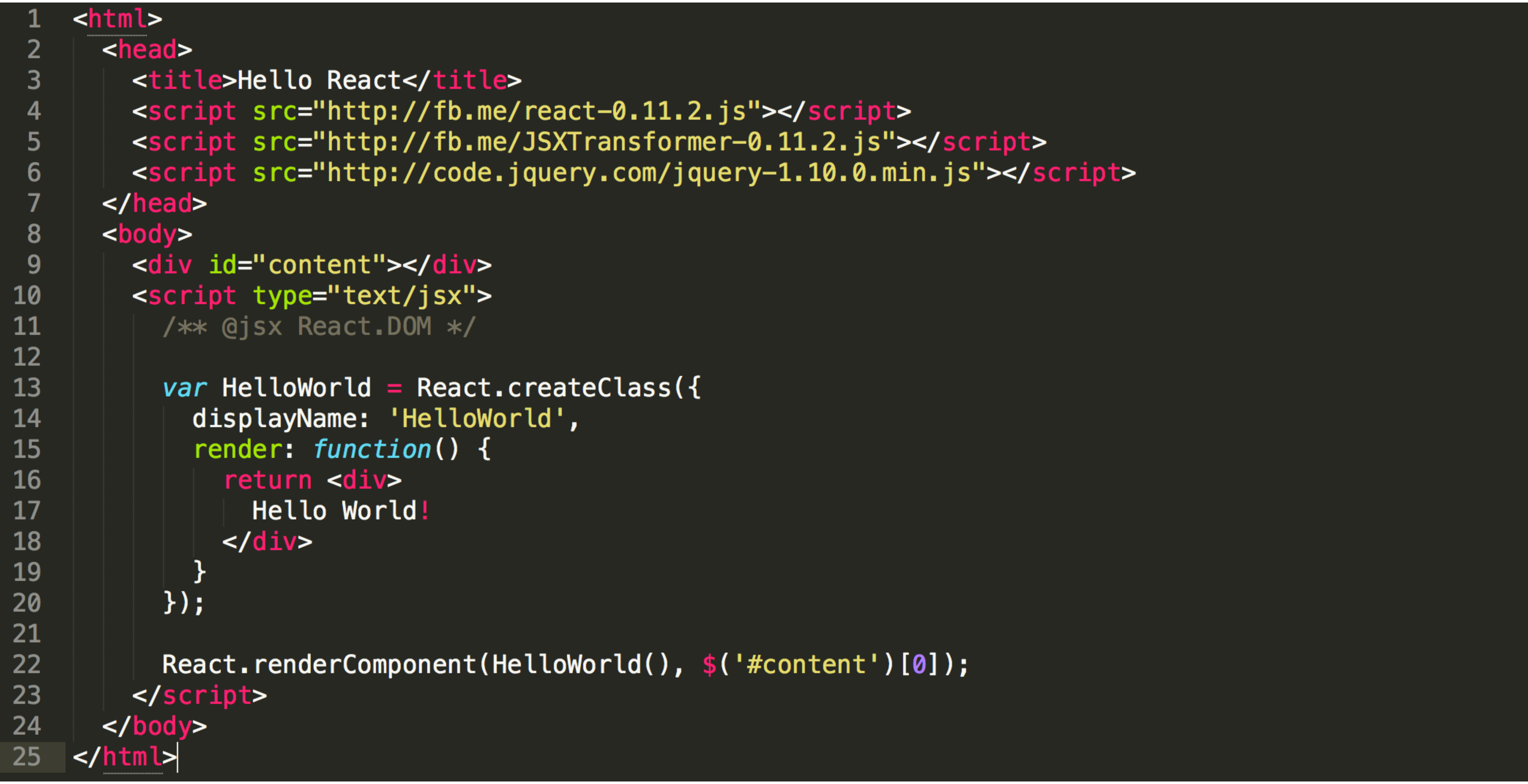


```
window.FactsView = AutoloadingCompositeView.extend({
  tagName: "div",
  className: "facts",
  itemViewContainer: ".facts",
  itemView: FactView,
  events: {
    "submit #create_fact_for_channel": "createFact",
    "focus #create_fact_for_channel textarea": "openCreateFactForm",
    "click #create_fact_for_channel .create_factlink .close": "closeCreateFactForm",
    "click #create_fact_for_channel": "focusCreateFactlink",
    "click #create_fact_for_channel .inset-icon.icon-pen": "toggleTitleField",
    "click #create_fact_for_channel .input-box": "focusField"
  },
```



```
<div class="header">
  {{#editable?}}
    <form id="create_fact_for_channel">
      <div class="create_factlink input-box">
        <textarea name="fact" class="input-xxlarge" required="required" placeholder="Post a ne

        <a class="close">x</a>

        <div class="inset-icon-box">
          <a class="inset-icon icon-pen" title="Add title"></a>
        </div>
      </div>

      <div class="add-title input-box">
        <input type="text" name="title" placeholder="Add title">
```

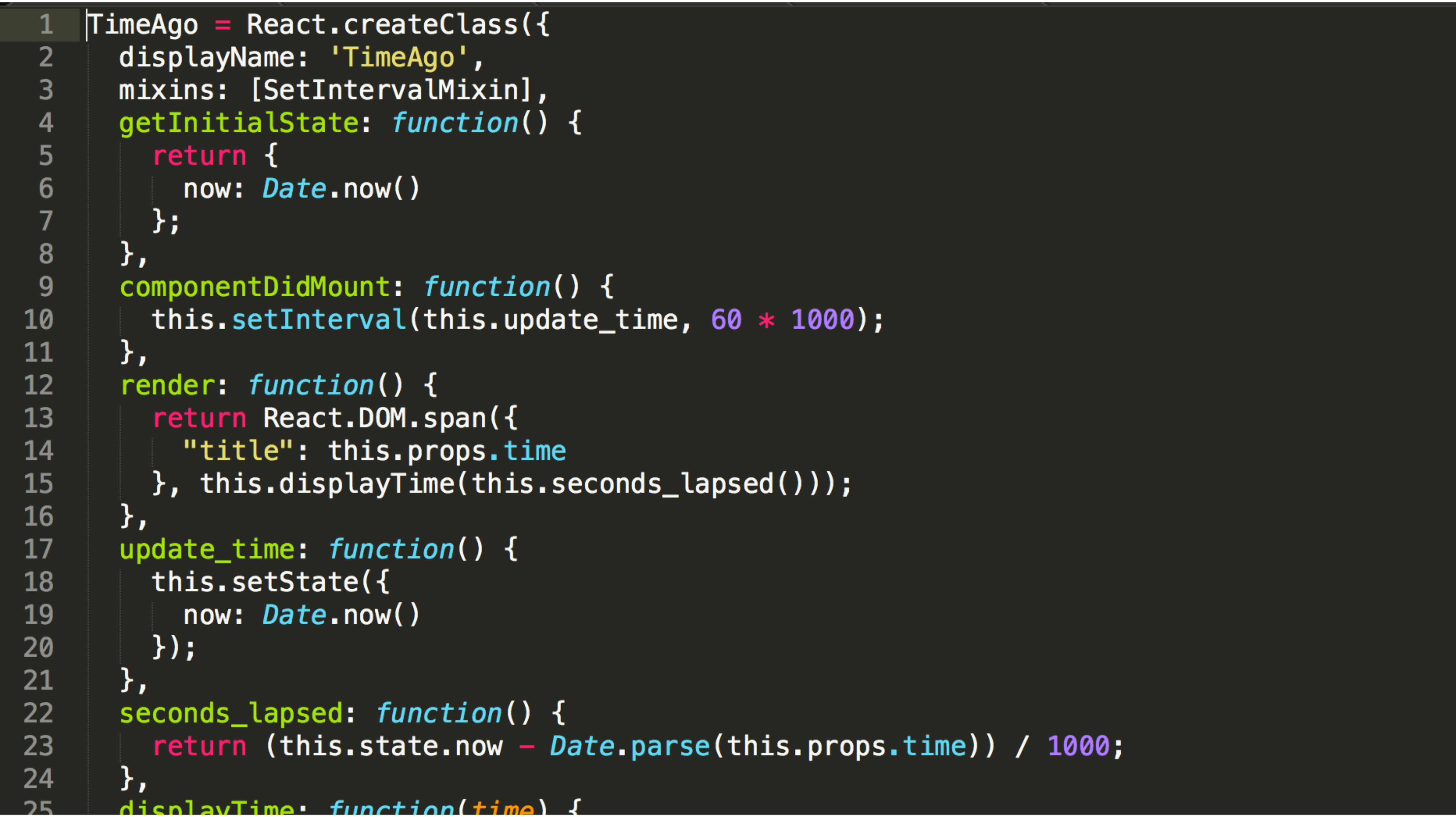
Getting started



```
<html>
  <head>
    <title>Hello React</title>
    <script src="http://fb.me/react-0.11.2.js"></script>
    <script src="http://fb.me/JSXTransformer-0.11.2.js"></script>
    <script src="http://code.jquery.com/jquery-1.10.0.min.js"></script>
  </head>
  <body>
    <div id="content"></div>
    <script type="text/jsx">
      /** @jsx React.DOM */

      var HelloWorld = React.createClass({
        displayName: 'HelloWorld',
        render: function() {
          return <div>
            Hello World!
          </div>
        }
      });

      React.renderComponent(HelloWorld(), $('#content')[0]);
    </script>
  </body>
</html>
```

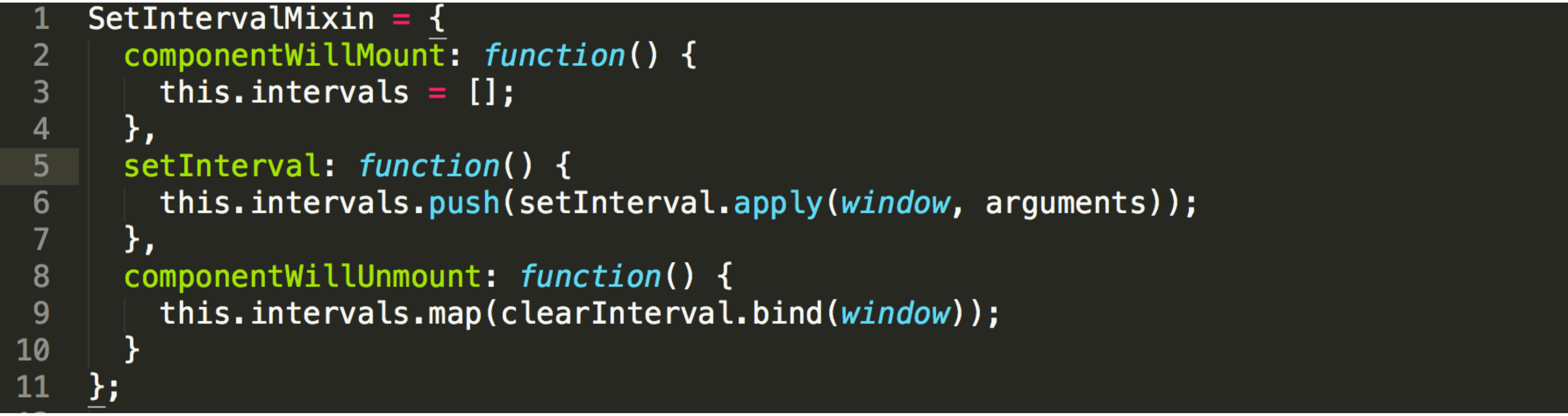
http://facebook.github.io/react/docs/tutorial.html

More advanced



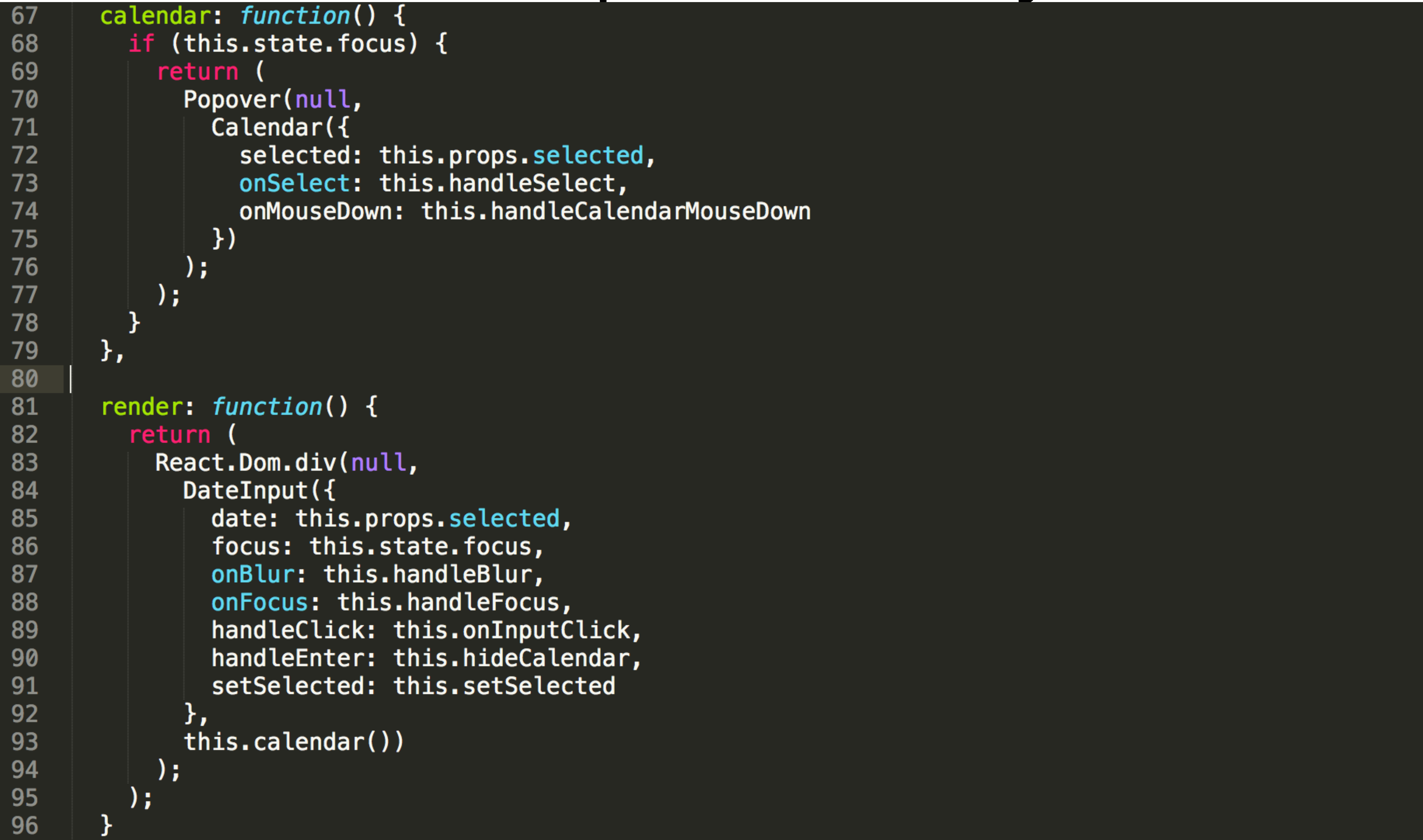
```
TimeAgo = React.createClass({
  displayName: 'TimeAgo',
  mixins: [SetIntervalMixin],
  getInitialState: function() {
    return {
      now: Date.now()
    };
  },
  componentDidMount: function() {
    this.setInterval(this.update_time, 60 * 1000);
  },
  render: function() {
    return React.DOM.span({
      "title": this.props.time
    }, this.displayTime(this.seconds_lapsed()));
  },
  update_time: function() {
    this.setState({
      now: Date.now()
    });
  },
  seconds_lapsed: function() {
    return (this.state.now - Date.parse(this.props.time)) / 1000;
  },
  displayTime: function(time) {
```

Mixins



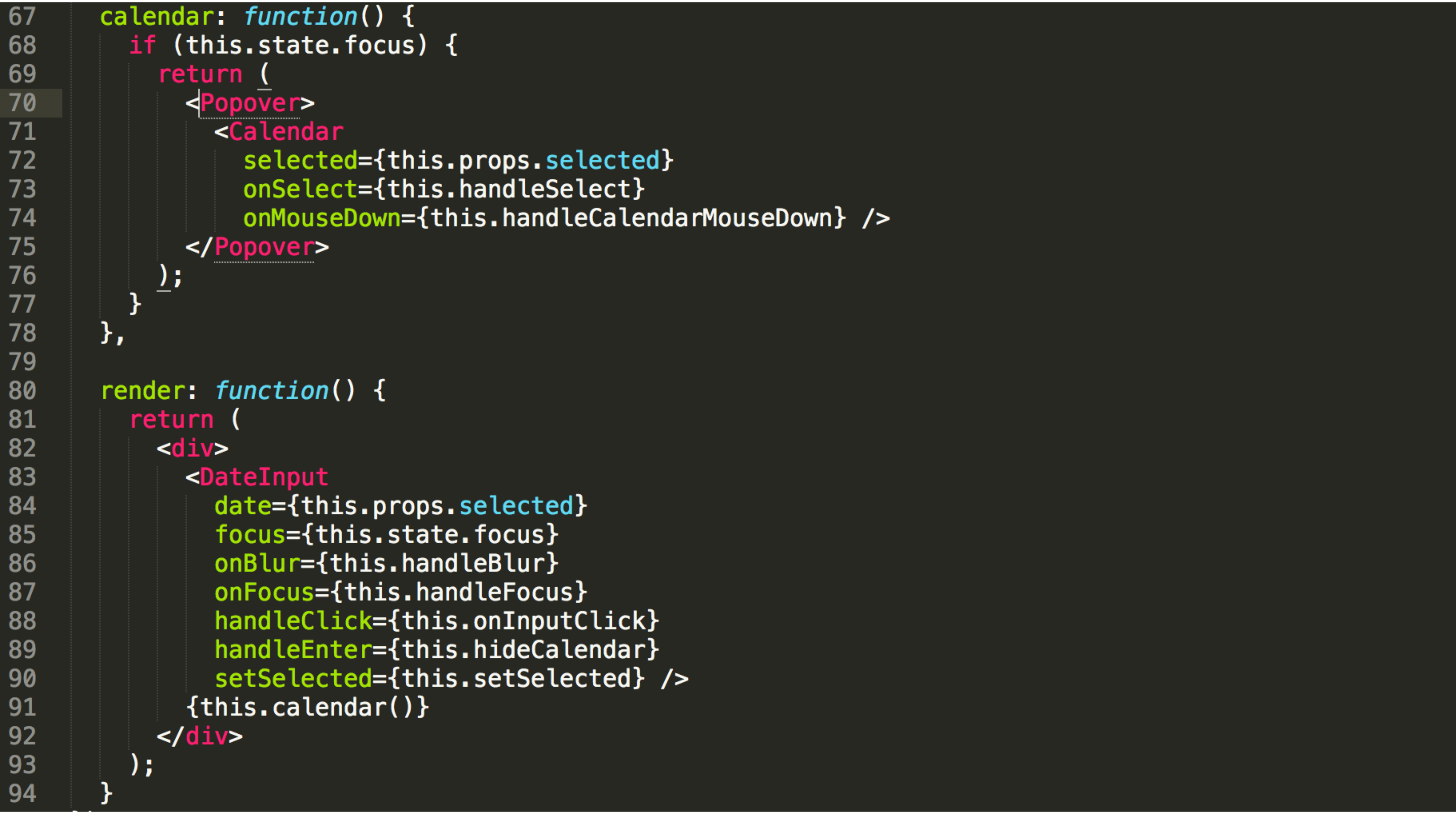
```
SetIntervalMixin = {
  componentWillMount: function() {
    this.intervals = [];
  },
  setInterval: function() {
    this.intervals.push(setInterval.apply(window, arguments));
  },
  componentWillUnmount: function() {
    this.intervals.map(clearInterval.bind(window));
  }
};
```

Composability



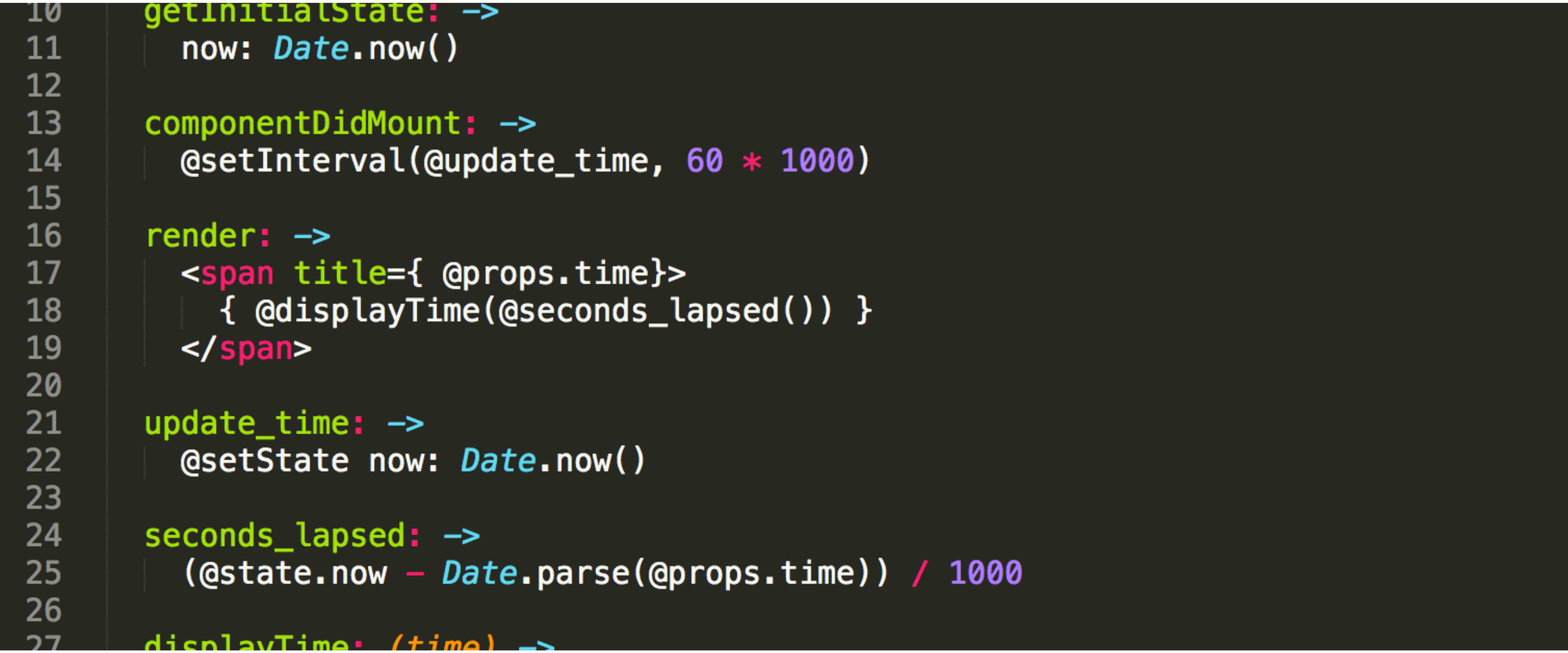
```
  calendar: function() {
    if (this.state.focus) {
      return (
        Popover(null,
          Calendar({
            selected: this.props.selected,
            onSelect: this.handleSelect,
            onMouseDown: this.handleCalendarMouseDown
          })
        );
      );
    }
  },

  render: function() {
    return (
      React.Dom.div(null,
        DateInput({
          date: this.props.selected,
          focus: this.state.focus,
          onBlur: this.handleBlur,
          onFocus: this.handleFocus,
          handleClick: this.onInputClick,
          handleEnter: this.hideCalendar,
          setSelected: this.setSelected
        },
        this.calendar())
      );
    );
  }
```

JSX



```
  calendar: function() {
    if (this.state.focus) {
      return (
        <Popover>
          <Calendar
            selected={this.props.selected}
            onSelect={this.handleSelect}
            onMouseDown={this.handleCalendarMouseDown} />
        </Popover>
      );
    }
  },

  render: function() {
    return (
      <div>
        <DateInput
          date={this.props.selected}
          focus={this.state.focus}
          onBlur={this.handleBlur}
          onFocus={this.handleFocus}
          handleClick={this.onInputClick}
          handleEnter={this.hideCalendar}
          setSelected={this.setSelected} />
        {this.calendar()}
      </div>
    );
  }
```

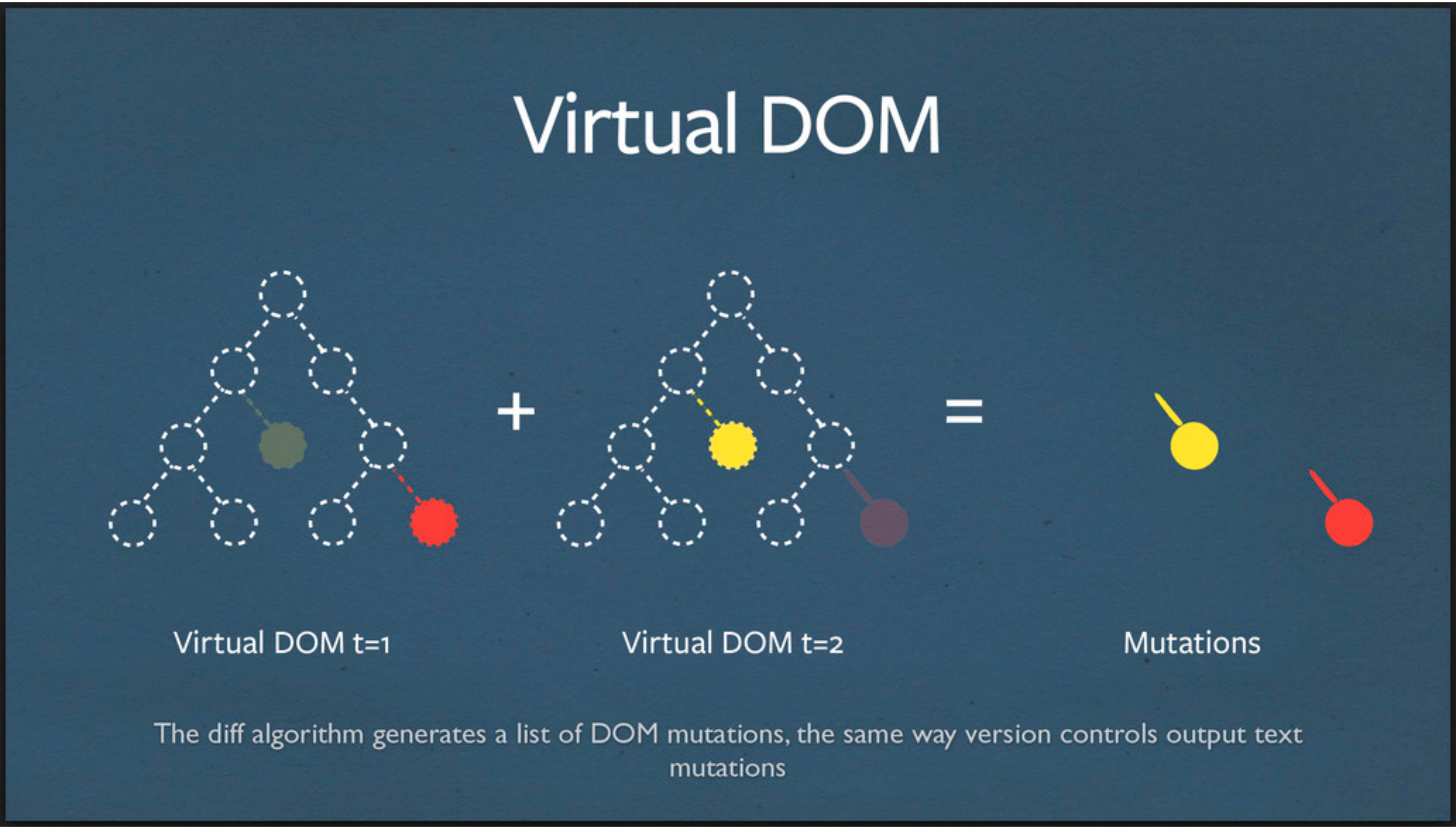
CJSX



```
getInitialState: ->
  now: Date.now()

componentDidMount: ->
  @setInterval(@update_time, 60 * 1000)

render: ->
  <span title={ @props.time}>
    { @displayTime(@seconds_lapsed()) }
  </span>

update_time: ->
  @setState now: Date.now()

seconds_lapsed: ->
  (@state.now - Date.parse(@props.time)) / 1000

displayTime: (time) ->
```

Rendering

- Rendering everything is conceptually simple
- **`render()`** is called every time state/props change
- Naive implementation would be really slow

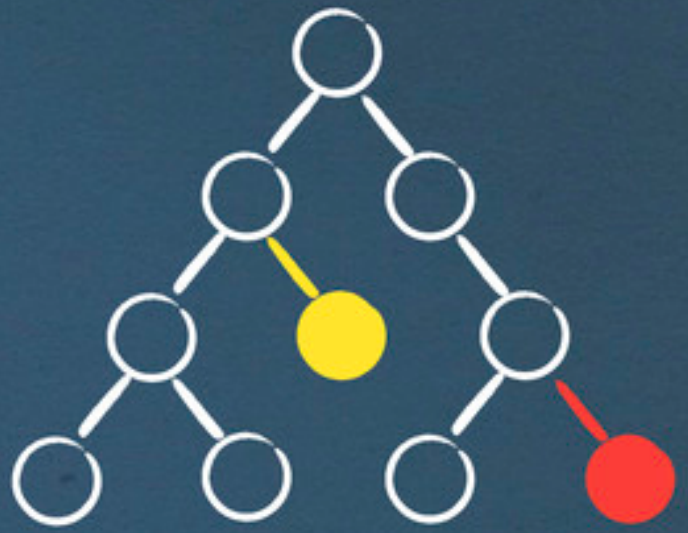


Virtual DOM

The diff algorithm generates a list of DOM mutations, the same way version controls output text mutations

Credits: Christopher “vjeux” Chedeau

Virtual DOM



That we can apply to the real DOM. Then we let the browser do all its optimized pipeline. We reduced the number of expensive, but needed, DOM mutations to the bare minimum

Credits: Christopher “vjeux” Chedeau

Best practices

- Minimize `state`
- Isolate state to a logical place in a component
- Do not mutate the DOM directly.
- Think of components as state machines
- Pass `props` in one direction, use callbacks

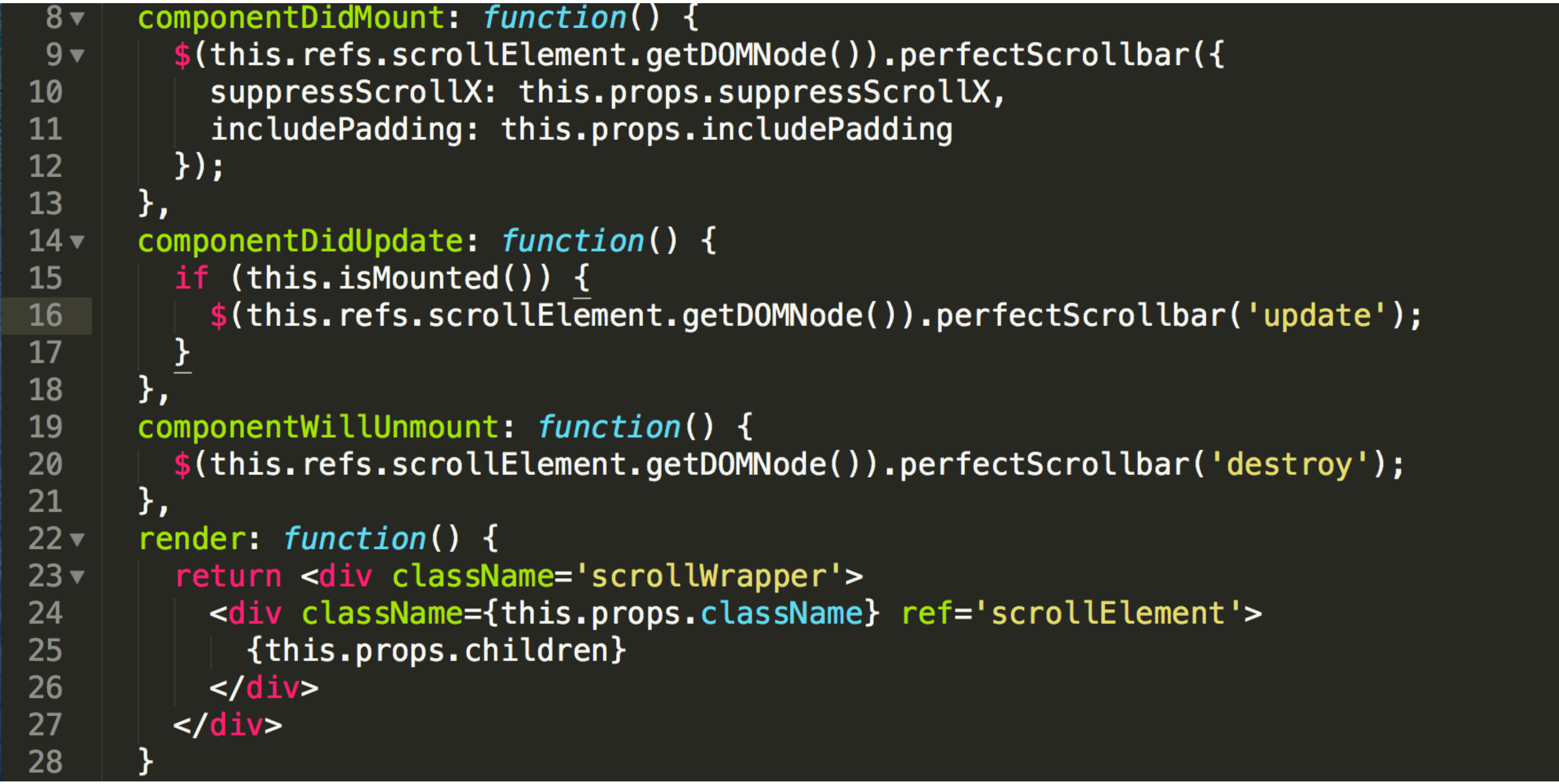
Integration

- Most components should only contain display logic
- Use a root component that does orchestration
- Use another place to do that (`Backbone.Router`)

Existing plugins

- Some lifecycle methods:
  - `componentDidMount()`
  - `componentDidUpdate()`
  - `componentWillUnmount()`
- `getDomNode()`
- `refs`

perfectScrollbar



```
componentDidMount: function() {
  $(this.refs.scrollElement.getDOMNode()).perfectScrollbar({
    suppressScrollX: this.props.suppressScrollX,
    includePadding: this.props.includePadding
  });
},
componentDidUpdate: function() {
  if (this.isMounted()) {
    $(this.refs.scrollElement.getDOMNode()).perfectScrollbar('update');
  }
},
componentWillUnmount: function() {
  $(this.refs.scrollElement.getDOMNode()).perfectScrollbar('destroy');
},
render: function() {
  return <div className='scrollWrapper'>
    <div className={this.props.className} ref='scrollElement'>
      {this.props.children}
    </div>
  </div>
}
```

"I also like things about React, which is in some ways less ambitious because React just says, "We're the view. You can use us with Ember or Angular.""

"And the view, there are various ways to skin that cat and I like the React way of using virtual DOM diffing and immutability."

–Brendan Eich

http://javascriptjabber.com/124-jsj-the-origin-of-javascript-with-brendan-eich/

`logout`