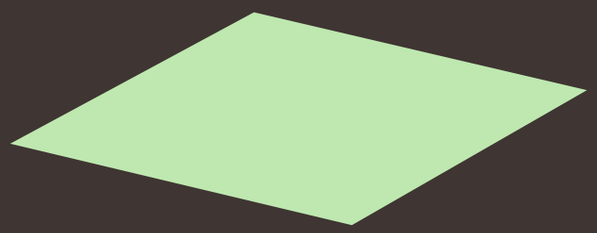
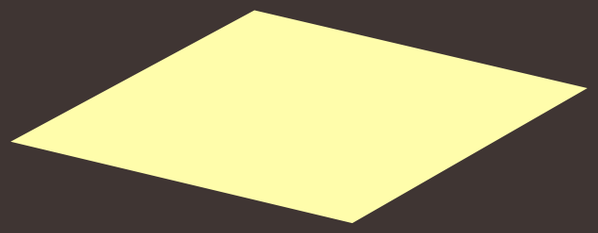
「React.js入門」にお越し頂き
ありがとうございます！

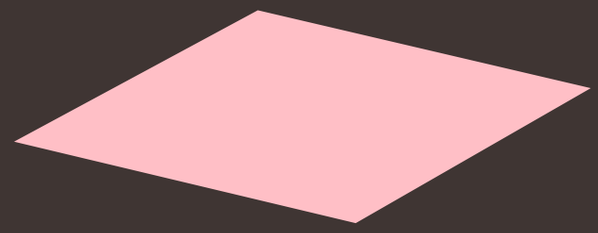
- github.com/OUCC/KC3_React からサンプルコードのダウンロードをお願いします。



JavaScript
初めての方



JavaScript
経験者



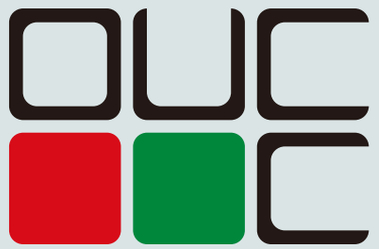
React.js
経験者

- をPC等見やすい位置に貼り付けてください。
黄、赤の方は、ぜひ緑の方の手助けをお願いします！

React.jsでクライアントサイドな Webアプリ入門

2015-9-13 KC3 2015 勉強会

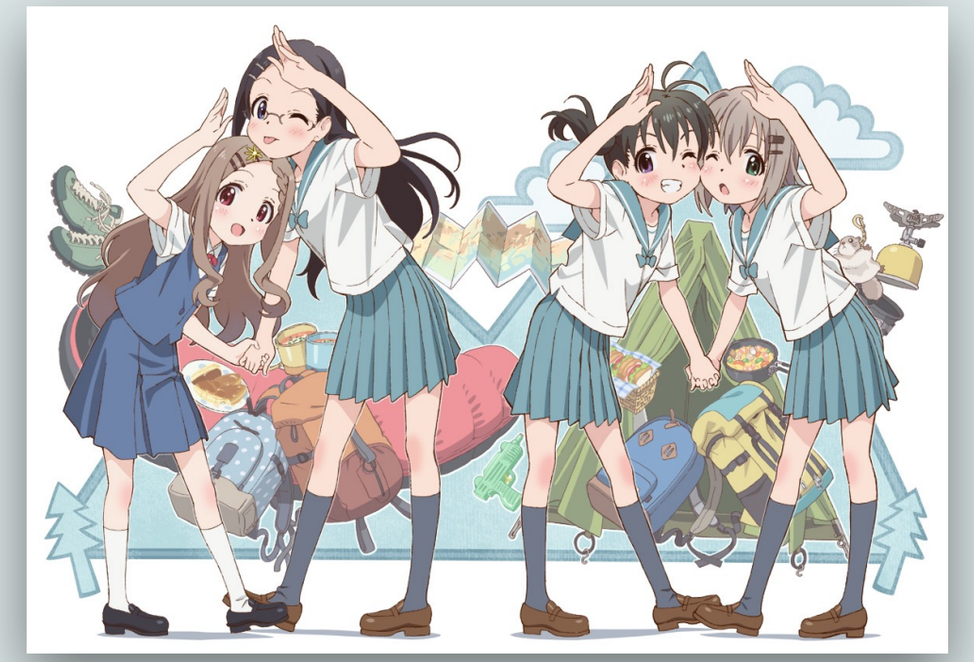
OUCC担当
はるさめ (@spring_raining)

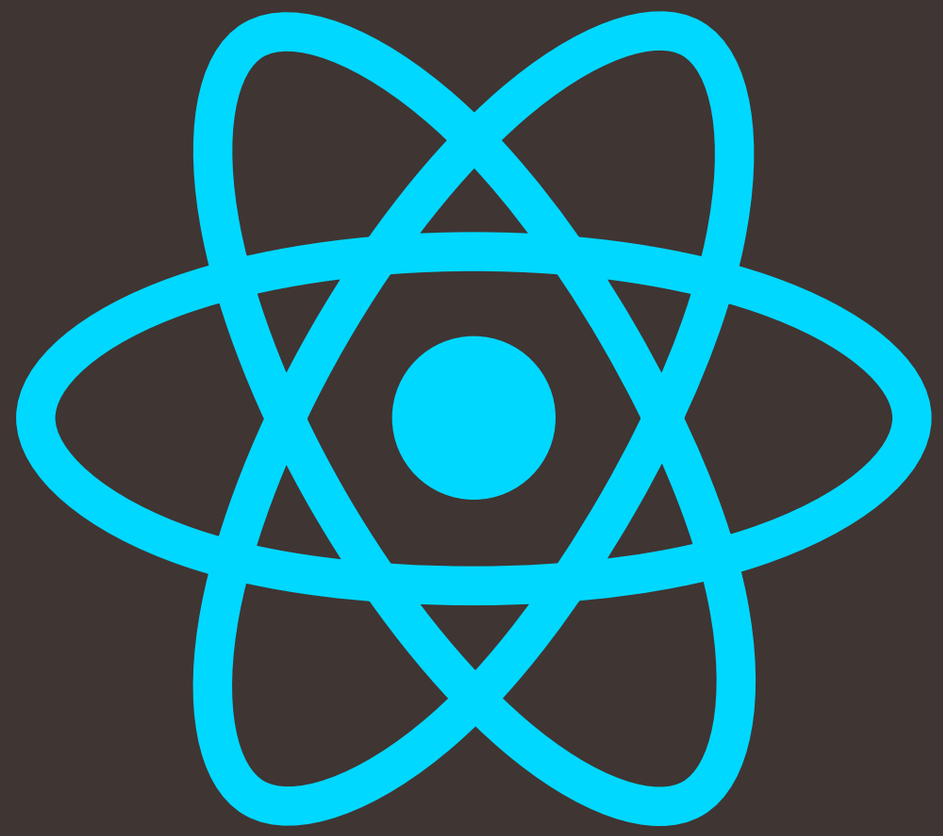


こんにちは

わたしはだれ

- はるさめ @spring_raining
- 大阪大学 3年生(2回目)
- スペースが余ったので以下おすすめのアニメです





https://github.com/facebook/react/wiki/Sites-Using-React

Apps using React Native

Here is a list of apps using **React Native**. Submit a pull request on GitHub to list your app.

Beetroot

By Alex Duckmanton

Discord

By Hammer & Chisel

Discovery VR

By Discovery Communications

DropBot

By Peach Labs

Exponent

By Charlie Cheever & James Ide

F8

By Facebook

Facebook Groups

By Facebook

Facebook Adverts Manager - Android

By Facebook

Facebook Ads Manager - iOS

By Facebook

HSK Level 1 Chinese Flashcards

By HS Schaaf

Leanpub

By Leanpub

Lrn

By Lrn Labs, Inc

Lumpen Radio

By Joshua Habdas

MinTrain

By Peter Cottle

Mr. Dapper

By wei ping woon

Ncredible

By NBC News Digital, LLC

Night Light

By Tian Yuan

ReactTo36

By Jonathan Solichin

RN Playground

By Joshua Sierles

SG Toto 4d

By Steve Ng

Spero for Cancer

By Spero.io

Start - medication

Tabtor Parent

By PrezAs Learning

Tong Xing Wang

By He Yin Tsun

1. 何はともあれJavaScript入門

JavaScript

- プログラミング言語
- (基本)ブラウザ上で動く
- 初めてならここおすすめです↓

  http://dotinstall.com/lessons/basic_javascript_v2

- 他の言語なら分かるよという方↓

  https://developer.mozilla.org/ja/docs/Web/JavaScript/A_re-introduction_to_JavaScript

- というか実は皆さんもう分かってる？

はじめよう

- 任意のブラウザを開いて、Ctrl+Shift+I (⌘+Option+I)
    - 開いたら「Console」を選択
- Node.jsインストールしてる人は**node**コマンドでも

JavaScript 文法とか

```
// ここはコメントです
/* これもコメントです */

var x = 123;
var y = "JavaScript";
console.log(x + y);

console.log(123 + 456);

var array = ["aaa", 'bbb', 123];
console.log(array[0]);

var object = {
    alfa: "apple",
    bravo: 123456,
    charlie: ["yuno", "miyako"]
};

console.log(object.charlie[1]);
```

JavaScript 関数

```
function multi(x, y) {
    var answer = x * y;
    return answer;
}
console.log(multi(2, 3));
console.log(answer);          // エラー
```

```
var gj = function() { return 2013; };

var kobo = {
    MDS: 2014,
    GJ: gj,
    YRYR: function(season) {
        if (season === 1) { return 2011; }
        if (season === 2) { return 2012; }
        if (season === 3) { return 2015; }
    }
};

console.log(kobo.YRYR(3));
```

JavaScript 条件式/ループ

```
var x = "Umaru";
var y = 2;

if (1 + 1 <= y) {
    x = "Ebina";
} else if (y !== "2") {
    x = "Kirie";
} else {
    x = "Sylphynford";
}

var abbr = ["U", "M", "R"];
for (var i = 0; i < abbr.length; i++) {
    console.log(abbr[i]);
}
var n = 0;
while (n < 3) {
    console.log(abbr[n]);
    n += 1;
}
```

forよりもおすすめです

```
[4, 6, 2, 5]
    .filter(function(e) {
        return e > 3;
    }, this)                    // [4, 6, 5]

    .map(function(e) {
        return e * 10;
    }, this)                    // [40, 60, 50]

    .forEach(function(e, i) {
        var print = i + ":" + e;
        console.log(print);
    }, this);
```

2. 何はともあれHTML入門

HTML (Hyper Text Markup Language)

- プログラミングらない言語
- 世のウェブサイトはこれで出来ている
  - CSSは今日はやりません_ _
- 初めてならこことかどうでしょう↓

  http://www.tohoho-web.com/wwwbeg.htm
- というか実は皆さんもう分かってる？

HTMLの文法

```
<タグ　属性=値>内容</タグ>

<div>
    <a href="http://google.com">ぐーぐる</a>
</div>
```

```
<!DOCTYPE html>
<html lang="ja">
<head>
    <meta charset="UTF-8">
    <title>タイトル</title>
</head>
<body>
    ここに内容
</body>
</html>
```

3. Reactいってみよう

Reactとは

- Facebookが開発している、HTML(DOM)の管理を行うJavaScriptライブラリ
    - （MVCでいうところのV）
    - 具体的には、Componentという見た目を管理するオブジェクトを複数作って組み合わせていく
- JavaScriptにHTMLをくっつけたみたいな**JSX**という文法が使える

JSX?

具体的にはこういう感じの

```jsx
render: function() {
  return (
    <div className="commentBox">
      Hello, world! I am a CommentBox.
    </div>
  );
}
```

JSX

```js
render: function () {
  return (
    React.createElement('div', {className: "commentBox"},
      "Hello, world! I am a CommentBox."
    )
  );
}
```

JS

とりあえずコードを見てみよう

サンプルコードを用意しました

/KC3/01hello/index.htmlを開いてみよう

```
<!DOCTYPE html>
<html lang="ja">
  <head>
    <meta charset="UTF-8">
    <title>React.js example</title>
    <link rel="stylesheet" href="../shared/css/base.css" />
    <script src="../../build/react.js"></script>
    <script src="../../build/JSXTransformer.js"></script>
  </head>
  <body>
    <h1>React.js example</h1>
    <div id="container">
      これが見えている場合、あなたのコードは正しく動いていません:(
    </div>
    <script type="text/jsx">
      var Hello = React.createClass({
        render: function() {
          return <p>Hello, {this.props.message}!</p>;
        }
      });
      React.render(
        <Hello message="React" />,
        document.getElementById("container")
      );
    </script>
  </body>
</html>
```

/KC3/01hello/index.htmlを開いてみよう

```
<div id="container">
  これが見えている場合、あなたのコードは正しく動いていません:(
</div>
<script type="text/jsx">
  var Hello = React.createClass({
    render: function() {
      return <p>Hello, {this.props.message}!</p>;
    }
  });
  React.render(
    <Hello message="React" />,
    document.getElementById("container")
  );
</script>
```

`props`

- Componentに値を渡す方法の1つ
- Component内で`props`の値は読み込めるけど、
  書き換えることはできない

➡ **Componentのモジュール化**

いまいち便利さが伝わらない…

/KC3/02text/index.html

```
var Text = React.createClass({
  getInitialState: function() {
    return { text: "" };
  },
  onChangeText: function(e) {
    this.setState({ text: e.target.value });
  },
  render: function() {
    return <div>
      <input type="text" value={this.state.text}
          onChange={this.onChangeText} />
      <p>{this.state.text}</p>
    </div>;
  }
});
React.render(
  <Text />,
  document.getElementById("container")
);
```

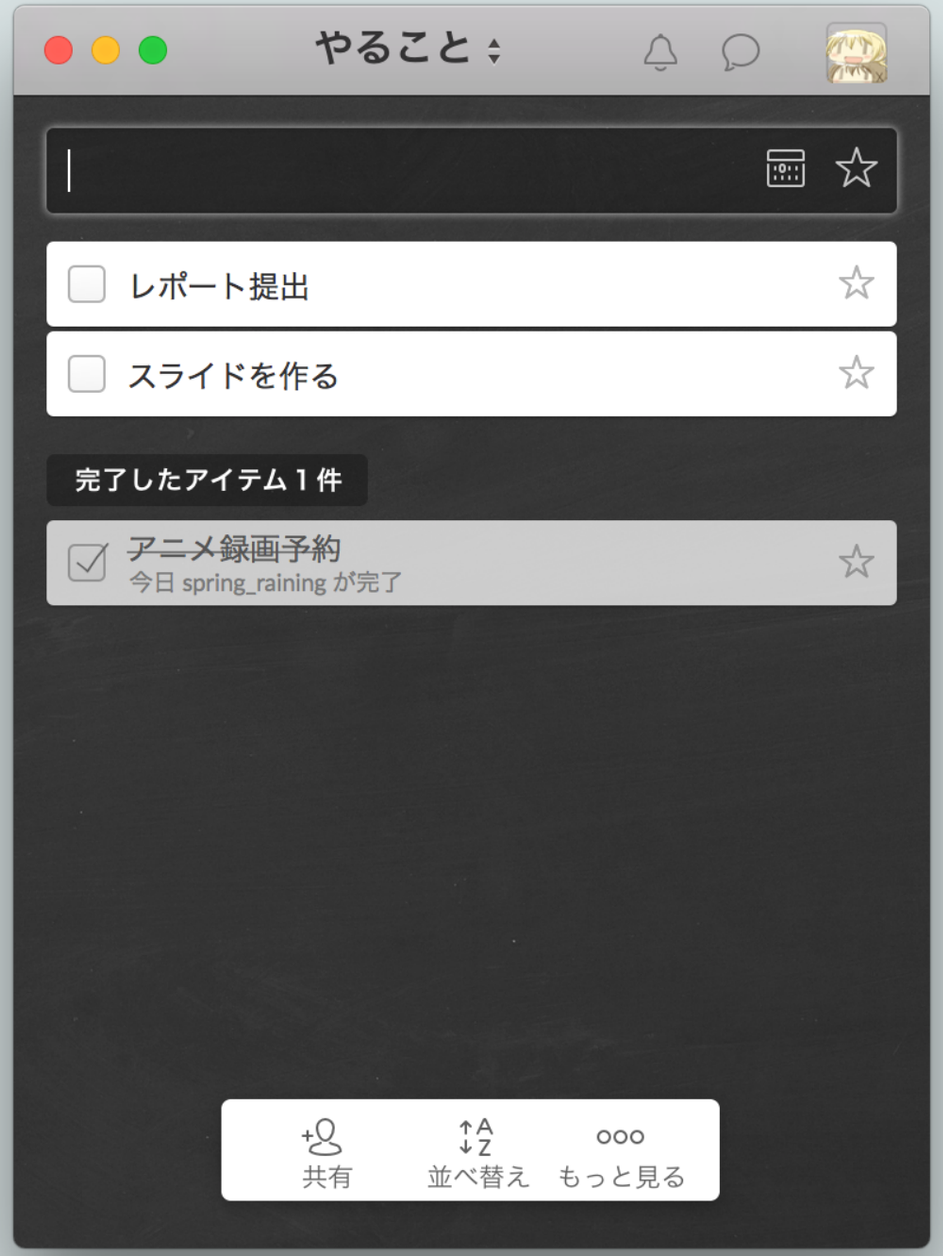
`state`

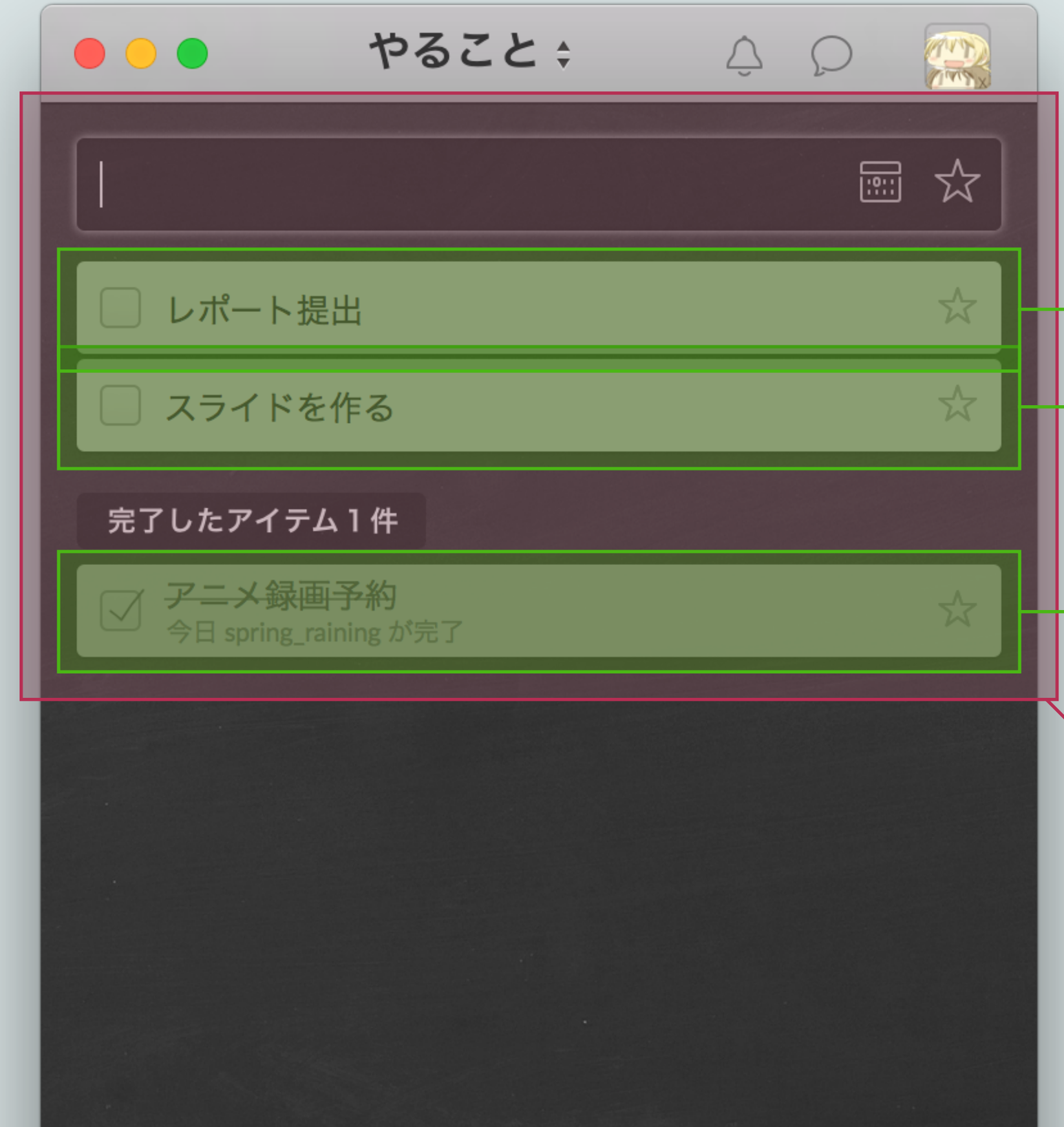
- Componentが持つ書き換え可能な値
- `state`が書き換わると、`state`を利用しているDOMも**自動的に**書き換わる

➡ **“React”ive!!**

早速ですが

- ToDoアプリを作ってみましょう
- やること(ToDo)を追加していき
  終わったものはチェック
- 作例を /KC3/03todo に
  置いています




やること
レポート提出
スライドを作る
完了したアイテム１件
アニメ録画予約
今日 spring_raining が完了
Component
Component
Component
Component

ToDoアプリの方針

- ToDoアプリ本体になるTodo Componentと、やることの1項目を表すTask Componentを作っていく。

- Todo Componentはさっき作ったText Componentをもとに作成します。

Todo Component

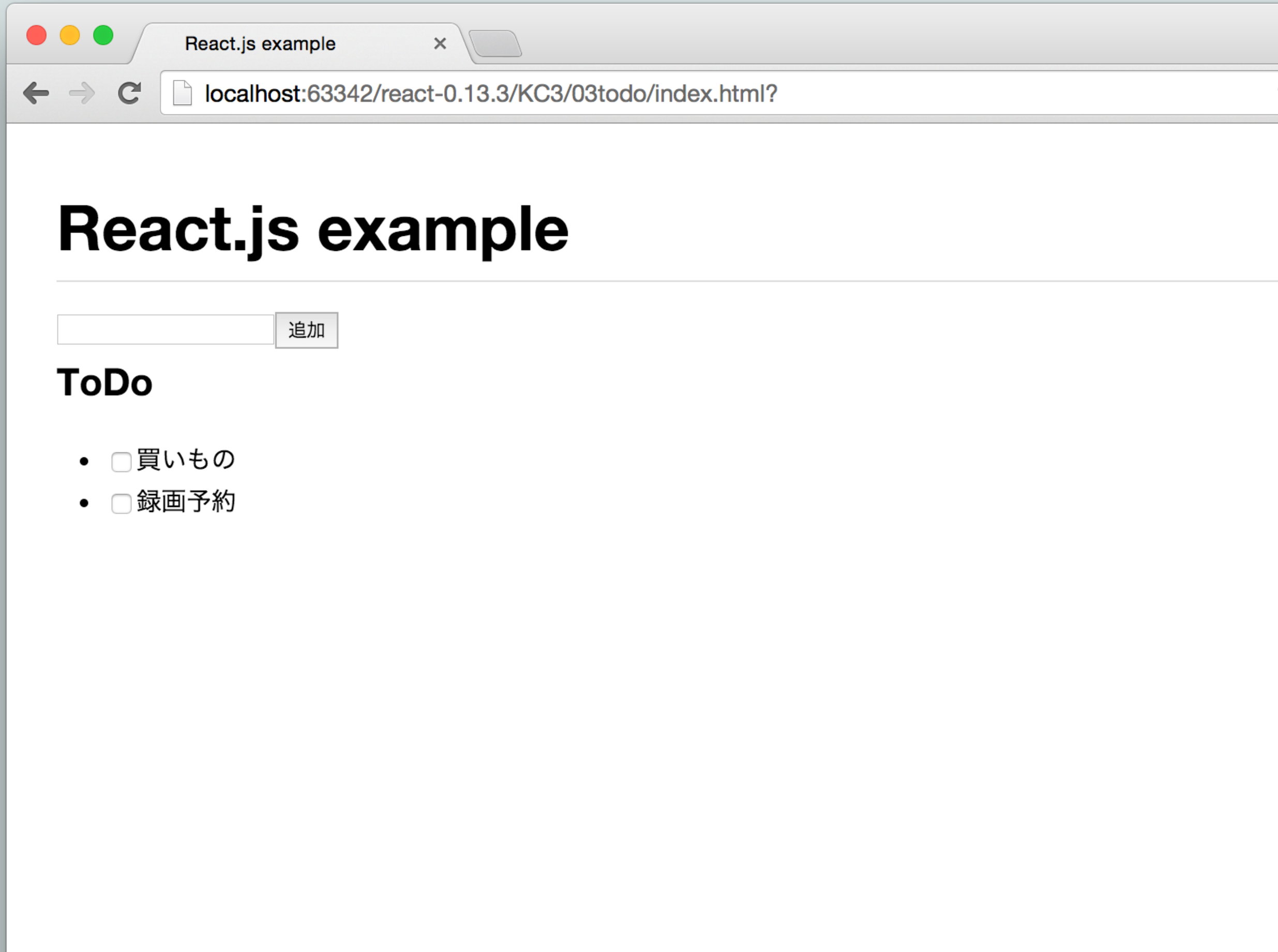
```
var Todo = React.createClass({
  getInitialState: function() {
    return {text: “”, tasks: []};
  },
  _ _ _ _ _ _ _ _
  render: function() {
    var tasks = []
    this.state.tasks.map(function(e, i) {
      tasks.push(
          <Task key={i} id={i} text={e.text}
                complete={e.complete} />);
  }, this);
  return <div>
    <form onSubmit={this.onSubmitText}>
      <input type="text" value={this.state.text}
             onChange={this.onChangeText} />
      <button type="submit">追加</button>
    </form>
    <h3>ToDo</h3>
    <ul>{tasks}</ul>
  </div>;
}
```

Todo Component

```
onSubmitText: function(e) {
  e.preventDefault();
  if (this.state.text !== "") {
    var newTasks = this.state.tasks.concat({
      text: this.state.text,
      complete: false
    });
    this.setState({
      text: "",
      tasks: newTasks
    });
  }
},
```

Task Component

```
var Task = React.createClass({
  render: function() {
    return <li>
      <input type="checkbox" checked={this.props.complete} />
      {this.props.text}
    </li>;
  }
});
```


React.js example
localhost:63342/react-0.13.3/KC3/03todo/index.html?
React.js example
追加
ToDo
買いもの
録画予約

Todo

タスク作成

onSubmitText

Task

チェックボックス
クリック

Todo

タスク作成
onSubmitText

Task

Task Component

```
var Task = React.createClass({
  onChangeCheckbox: function() {
    this.props.onChange(this.props.id);
  },
  render: function() {
    return <li>
      <input type="checkbox" onChange={this.onChangeCheckbox}
             checked={this.props.complete} />
      {this.props.text}
    </li>;
  }
});
```

Todo Component

```
onChange: function(key) {
  var target = this.state.tasks[key];
  var newTasks = this.state.tasks.filter(function(e, i) {
    return i !== key;
  }).concat({
    text: target.text,
    complete: !target.complete
  });
  this.setState({tasks: newTasks});
},
- - - - - -
  tasks.push(
      <Task key={i} id={i} text={e.text}
            complete={e.complete} onChange={this.onChange}/>);
```

Todo Component

```
render: function() {
  var tasks = [];
  var done = [];
  this.state.tasks.map(function(e, i) {
    if (e.complete) {
      done.push(
          <Task key={i} id={i} text={e.text}
                complete={true} onChange={this.onChange}/>);
    } else {
      tasks.push(
          <Task key={i} id={i} text={e.text}
                complete={false} onChange={this.onChange}/>);
    }
  }, this);
  return <div>
         :
    <h3>ToDo</h3>
    <ul>{tasks}</ul>
    <h3>Complete</h3>
    <ul>{done}</ul>
  </div>;
```

React.js example

localhost:63342/react-0.13.3/KC3/03todo/index.html?

React.js example

追加

ToDo

- 蒼樹うめ展
- KC3

Complete

- 夏コミ

4. JSタスクランナーのいる日常

index.htmlだけだと…

- まあこうなると思います

```
    this.pos = pos;
  }
  return Rule;
})();

var Stylesheet = (function(parent) {
  function Stylesheet(pos, styles) {
    parent.call(this, pos);
    this.styles = styles;
  }
  Stylesheet.prototype.toString = function() {
    return "StyleSheet\n"
      + this.styles.map(function(e) {
        return prettify(e);
      }).join("\n");
  };
  return Stylesheet;
})(Rule);

var RuleList = (function(parent) {
  function RuleList(pos, rules) {
    parent.call(this, pos);
    this.rules = rules;
  }
  RuleList.prototype.toString = function() {
    return "RuleList\n"
      + this.rules.map(function(e) {
        return prettify(e);
      }).join("\n");
  };
  return RuleList;
})(Rule);

var AtRule = (function(parent) {
  function AtRule(pos, atKeywordToken, componentValues, tail)
    parent.call(this, pos);
    this.atKeywordToken = atKeywordToken;
    this.componentValues = componentValues;
    this.tail = tail;
  }
  AtRule.prototype.toString = function() {
    return "AtRule\n"
      + prettify(this.atKeywordToken) + "\n"
      + this.componentValues.map(function(e) {
        return prettify(e);
      }).join("\n") + "\n"
      + prettify(this.tail);
  };
  return AtRule;
})(Rule);

var QualifiedRule = (function(parent) {
  function QualifiedRule(pos, componentValues, braceBlock) {
    parent.call(this, pos);
    this.componentValues = componentValues;
    this.braceBlock = braceBlock;
  }
  QualifiedRule.prototype.toString = function() {
    return "QualifiedRule\n"
      + this.componentValues.map(function(e) {
        return prettify(e);
      }).join("\n") + "\n"
      + prettify(this.braceBlock);
  };
  return QualifiedRule;
})(Rule);

var DeclarationList = (function(parent) {
  function DeclarationList(pos, declarations) {
    parent.call(this, pos);
    this.declarations = declarations;
  }
  DeclarationList.prototype.toString = function() {
    return "DeclarationList\n"
      + this.declarations.map(function(e) {
        return prettify(e);
      }).join("\n");
  };
  return DeclarationList;
})(Rule);
```

ファイルを分割しよう

どうやって？

- JSファイルを複数作って`<script/>`で読み込む
  - `<script/><script/><script/>` …
- 別ファイルの変数を相互に参照したいときには？
  - がんばる
  - Reactiveとは…

Browserify使おう

- `require()`が使えるようになります
- Node.jsをインストールして
  `$ npm install -g browserify`
- (このあたりの分野は群雄割拠なので、各種ライブラリの思想、将来性についてよく調べて、お好きなものを使いましょう)
  - 検索キーワード: RequireJS, webpack, ES6, Babel.js

/KC3/04comp/

```
/KC3/04comp/task.jsx
var React = require("react");
var Task = React.createClass({
    :
module.exports = Task;
```

```
/KC3/04comp/task.jsx
var Task = require("./task.jsx");
var Todo = React.createClass({
    :
React.render(
  <Todo />,
  document.getElementById("container")
);
```

「Browserify」使ってみた。

- `$ npm init`
  適当にEnter押してると、「package.json」というファイルが作成されます

- `$ npm install react browserify reactify`
  「react」「browserify」「reactify」をインストール

- `$ browserify -t reactify todo.jsx > bundle.js`

- bundle.jsが作られる！

さらなる高みへ

- Gulpを使いましょう
- JSタスクランナーの一種
- `$ npm install -g gulp`
- (この辺りの分野は群雄割拠なので(ry

Gulpの流れ

- gulpfile.jsの`gulp.task()`でタスクを定義
- `$ gulp [command]`でタスク実行！
- サンプルコードではdefaultタスクとwatchタスクを用意しています
- browserifyしたり、minifyしたり、watchさせたり

Enjoy Reactive Coding!