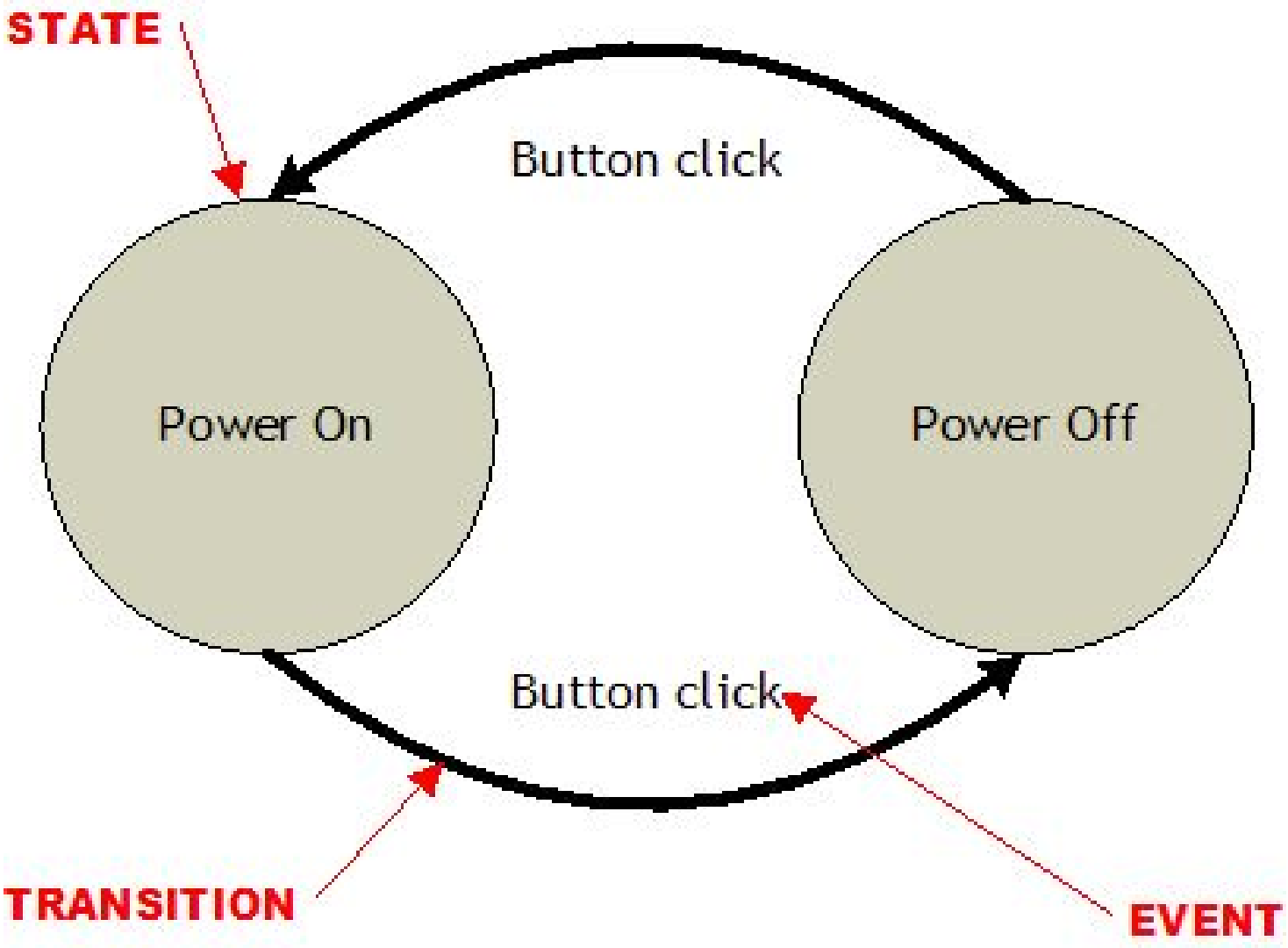
UI

State Machine

О состояниях и переходах между ними


STATE
Button click
Power On
Power Off
Button click
TRANSITION
EVENT

Реальная жизнь

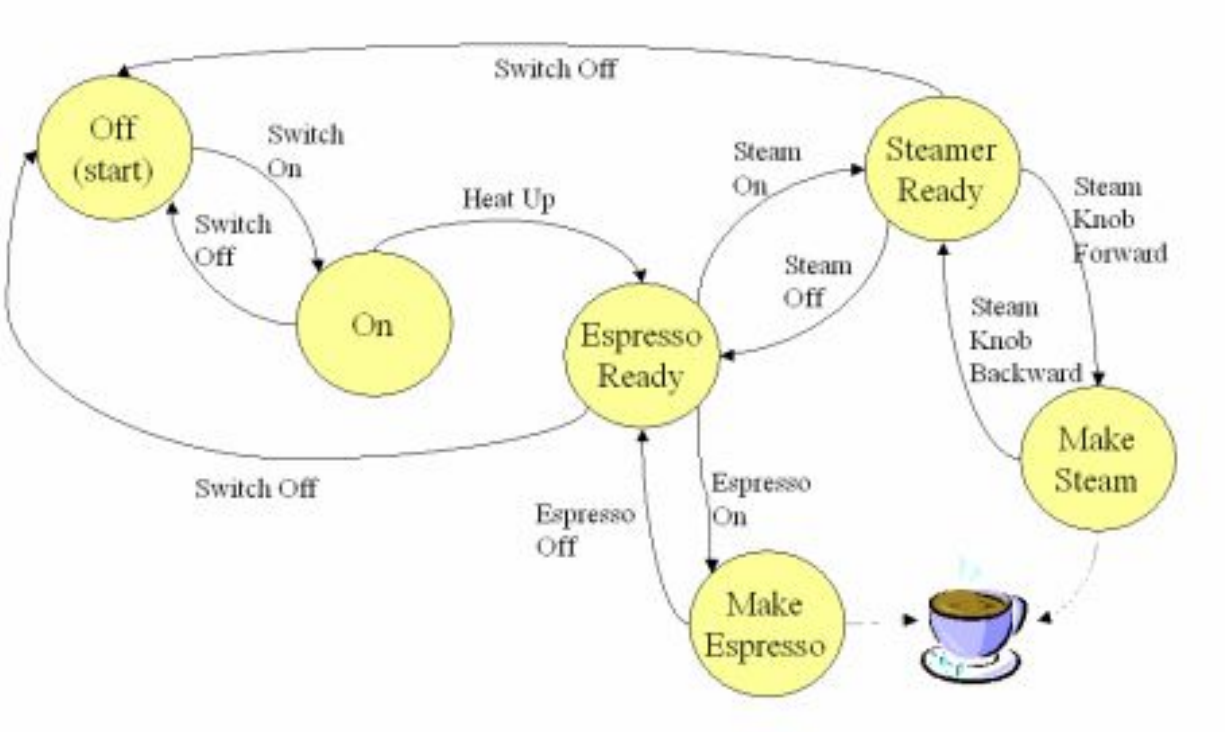
Вединговые аппараты

Бытовая электроника

Машины

Человек

Все что может меняться со временем

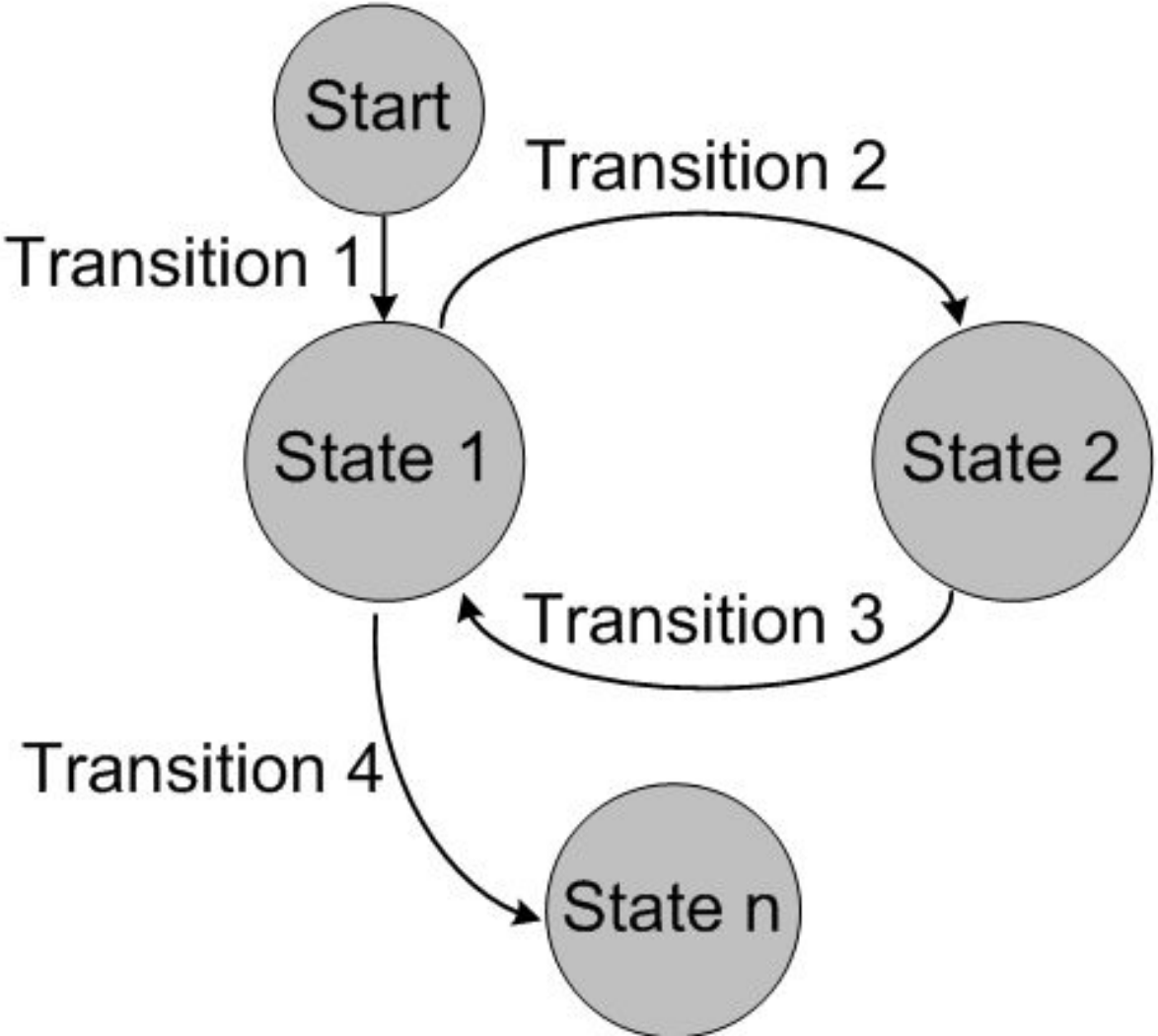


Программирование

Все что может меняться со временем

Состояние счета (оплачен/просрочен)

Страница (опубликована/скрыта)



При чем тут UI?

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

slider: artwork and screenshot close ups of tools in action.

News Headline

News Headline

News Headline

feature: something visual - upcoming release or competition or event... etc

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Sed a lorem ut est tincidunt consectetuer. Nulla facilisi.

Praesent aliquet, neque pretium congue mattis, ipsum augue dignissim ante, ac pretium nisl lectus.

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

Download

Click on your operating system to download the inkscape version that's right for you

Tutorials

Support

Examples

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

License and Code

GPL 3

Launchpad

Wiki

Get involved!

feature: something visual - upcoming release or competition or event... etc

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

Showcase 1

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

News Headline 1

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

Get Support!

Inkscape on Freenode

Inkscape on Jabber

Inkscape Forums

User Mailing List

feature: something visual - upcoming release or competition or event... etc

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

Get Involved!

Join the Dev team

Share your art

Help others

Bug testing

feature: something visual - upcoming release or competition or event... etc

Inkscape
draw freely

Download
Tutorials
Bug Tracker
License & code

Showcase 2

2э = 2п

э - экран, п - переход

5э = ?

Вспоминаем комбинаторику ;)

$$20 = 5*(5-1)$$

Для 10 экранов это 90

```
// Switch this view into `"editing"` mode, displaying the input field.
edit: function() {
  $(this.el).addClass("editing");
  this.input.focus();
},

// Close the `"editing"` mode, saving changes to the todo.
close: function() {
  this.model.save({content: this.input.val()});
  $(this.el).removeClass("editing");
},
```

Backbone View - Ручное управление DOM

А что с обычными сайтами? (клиент-серверная модель)

X Экранов = X Переходов

В реальности переходы остались, но на уровне данных

А если попробовать реализовать на клиенте?

Принцип работы: полная перерисовка всегда

Встречайте Act

https://github.com/mokevnin/act

В этом месте живой пример для тех кто слушает доклад в живую

Facts

- it works!
- ~ 60 sloc
- innerHTML
- State is Structure
- Render is Pure
- One-Way Data Flow

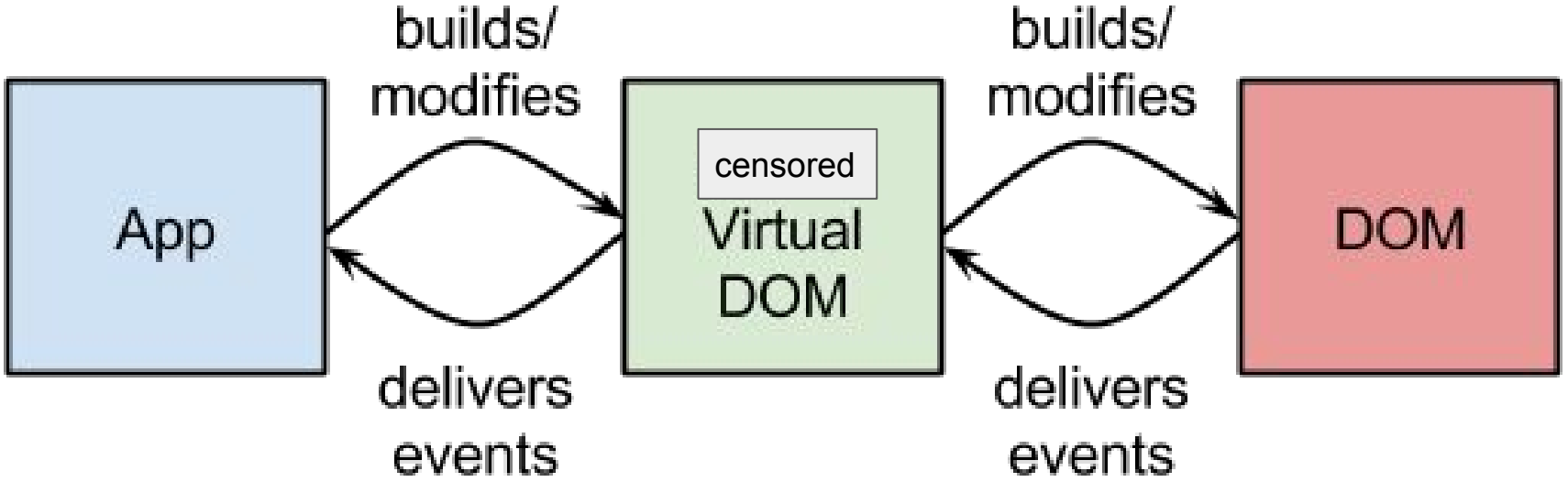
```
class ToDo extends Act.Component {
  getInitialState() {
    return {currentText: "", tasks: this.ops.tasks || []};
  }

  handleAddTask(e) {
    e.preventDefault();

    const tasks = this.state.tasks.slice();
    tasks.unshift(this.state.currentText);
    this.setState({currentText: "", tasks: tasks});
  }

  render() {
    const locals = {
      tasks: this.state.tasks,
      currentText: this.state.currentText
    };

    return templateFn(locals);
  }
```

В чем подвох?

- innerHTML - медленно
- HTML - не расширяемо. Приложение - один компонент.
- HTML - много проблем события/фокусы.

Что делать?

Virtual Dom


App
builds/
modifies
delivers
events
censored
Virtual
DOM
builds/
modifies
delivers
events
DOM

Virtual Dom

- Diff алгоритм может быть сведен к O(n) вместо O(n^3)
- JS адски быстрый, генерировать {} каждый раз легко
- В реальном DOM делаются только минимально необходимые изменения
- Никто не может делать изменения напрямую в реальный DOM, контролируемый виртуальным

На практике метод render должен возвращать структуру {} в терминах VDOM

Рабочий вариант, но глубина убивает понимание

```
function render(data) {
    return new VNode('div', {
        className: "greeting"
    }, [
        new VText("Hello " + String(data.name))
    ]);
}
```

А если эмулировать html?

Неплохо. Но это ведь js код...

```
class Toolbar extends Component<{}, {}, {}> {
  render() {
    return (
      <div className="toolbar">
        <div className="btn-group" role="group">
          <div className={this.getStatusClasses()}>
            <span className={this.getStatusInnerClasses()} />
          </div>
        </div>
      </div>
    );
  }
```

Пока не придумали ничего лучше, но на практике это работает нормально

Что получили?

- Своя система событий (исправляющая проблемы браузерной)
- Имена аттрибутов соответствуют спецификациям dom
- Возможность создавать новые компоненты. Композируемость
- Серверная генерация содержимого
- А если рендерить не в html? И это тоже


React

Hello, World!

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello {this.props.name}</div>;
  }
});

ReactDOM.render(<HelloMessage name="John" />, mountNode);
```

Reactjs

- Just View
- One-Way Data Flow
- Virtual Dom / Independent rendering
- Flux/React Native/Babel/Immutable.js

Reactjs

- Линейный рост сложности. Практически отсутсвует случайная сложность.
- Предсказуемое поведение. Простая отладка.
- Легко сопровождать. Есть один понятный и простой способ создавать архитектуру приложения. Сложно писать по другому.
- Раньше были проблемы с анимациями, сейчас не знаю. https://github.com/chenglou/react-motion

Спасибо! twitter.com/mokevnin