HASTENING REACT SSR WITH COMPONENT MEMOIZATION AND TEMPLATIZATION

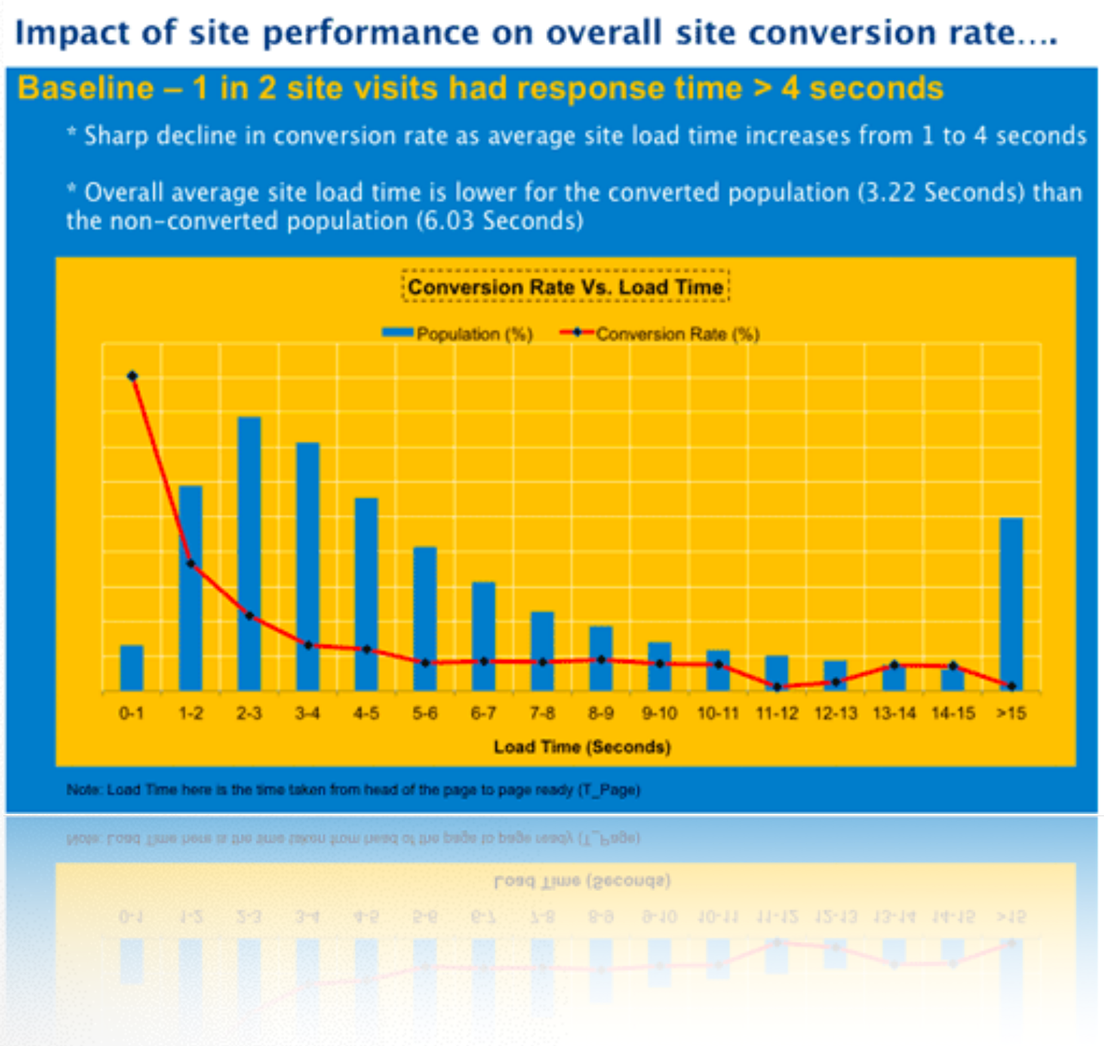
@WalmartLabs

Web App Performance

Increasing User Demand

www.radware.com

Time is Money

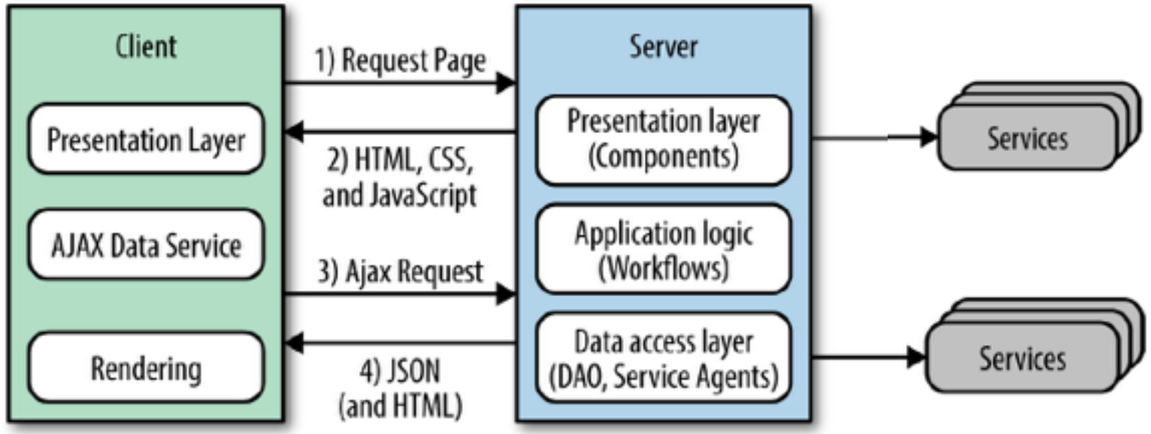


- For every 1 second of improvement, experienced up to a 2% increase in conversions
- For every 100 ms of improvement, grew incremental revenue by up to 1%

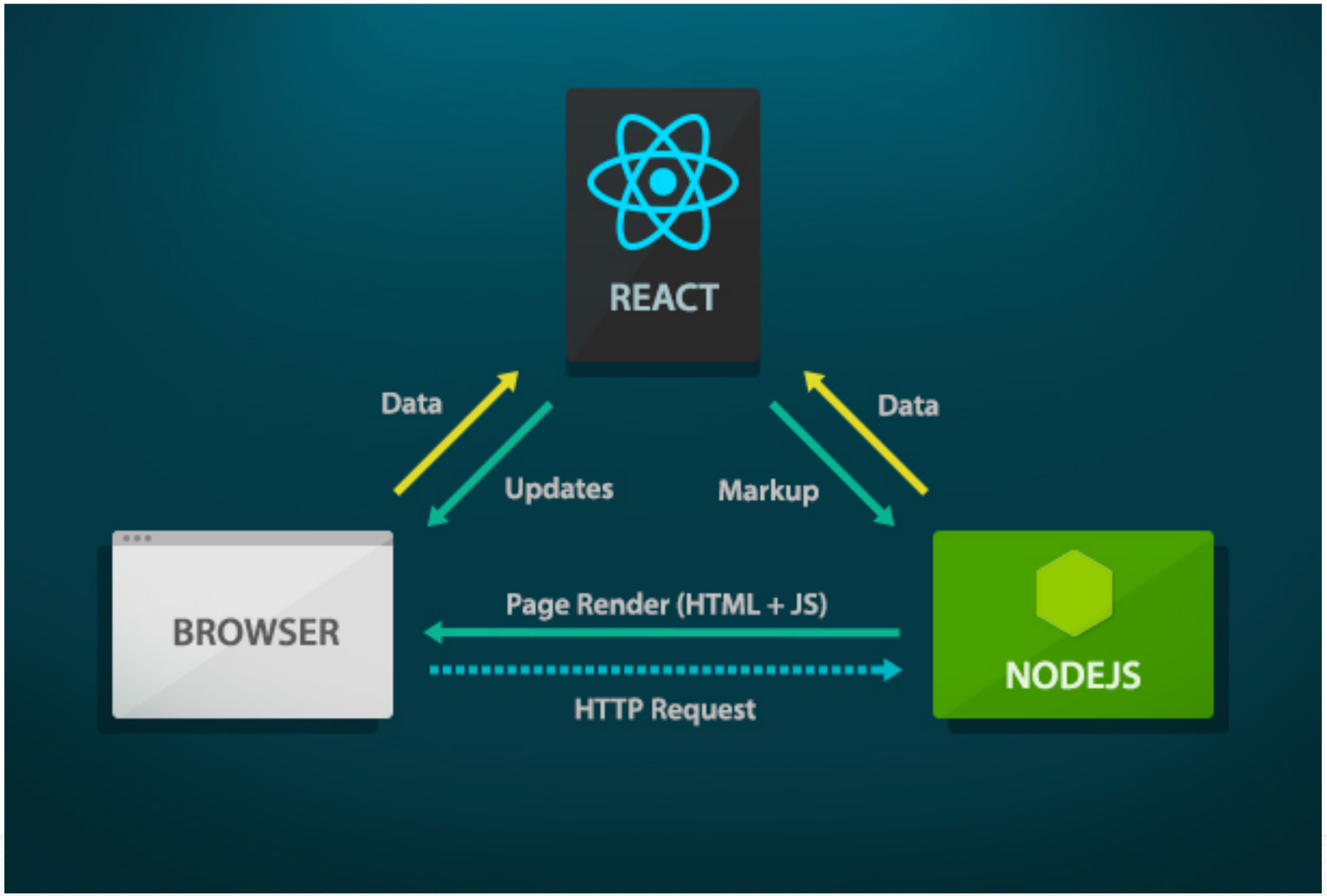
Source: http://www.globaldots.com/how-website-speed-affects-conversion-rates

Server-Side Rendering (SSR)

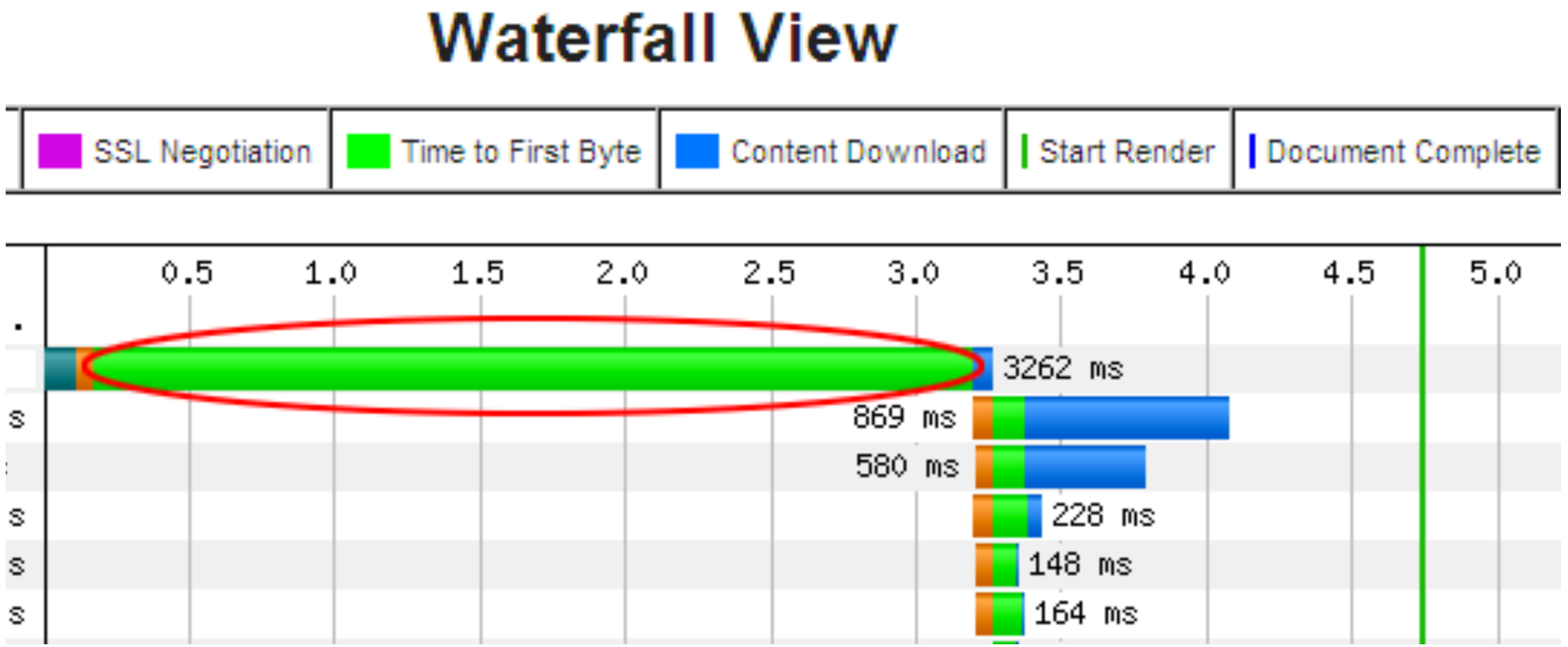
- Better user experience of the initial page load
- Better search engine ranking



SSR with **React**

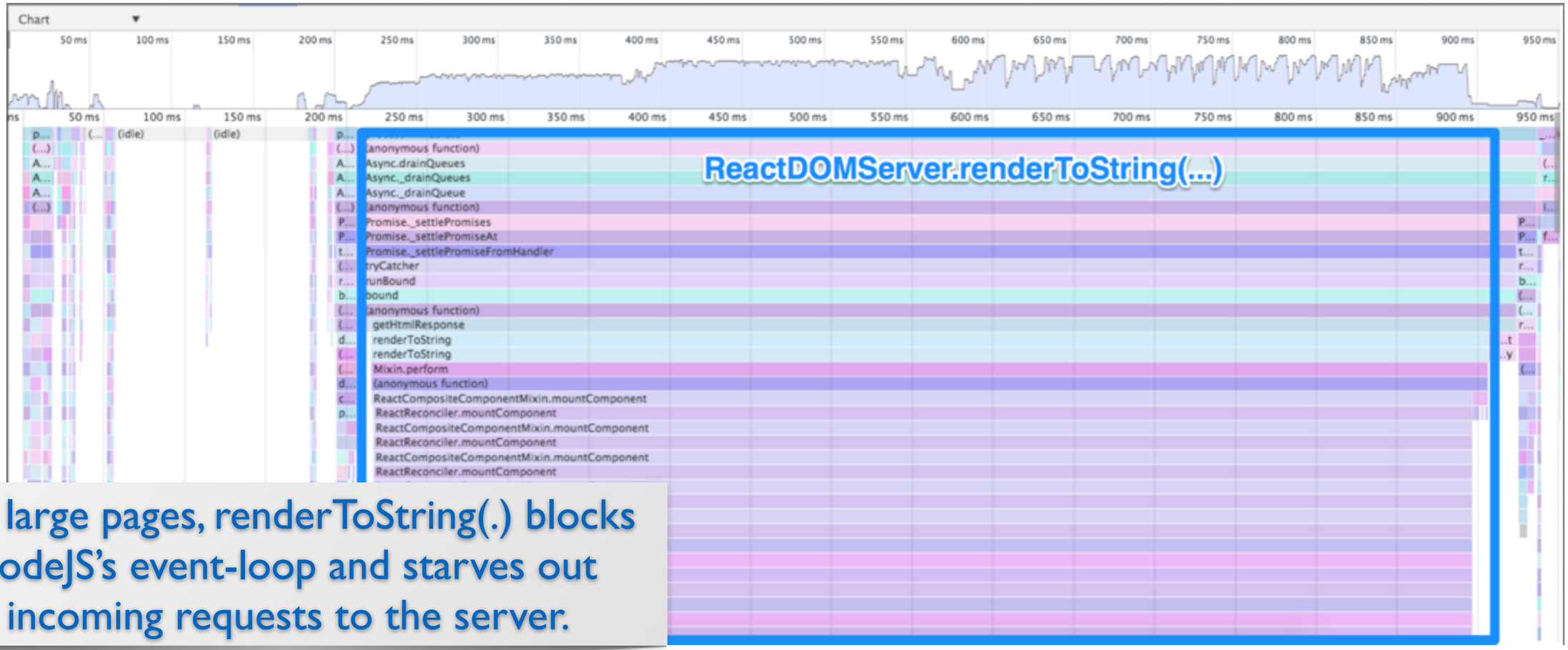


Server-Side **React**



Bad User Experience!

Server-Side **React**



On large pages, renderToString(.) blocks nodeJS's event-loop and starves out incoming requests to the server.

CPU profile for one request

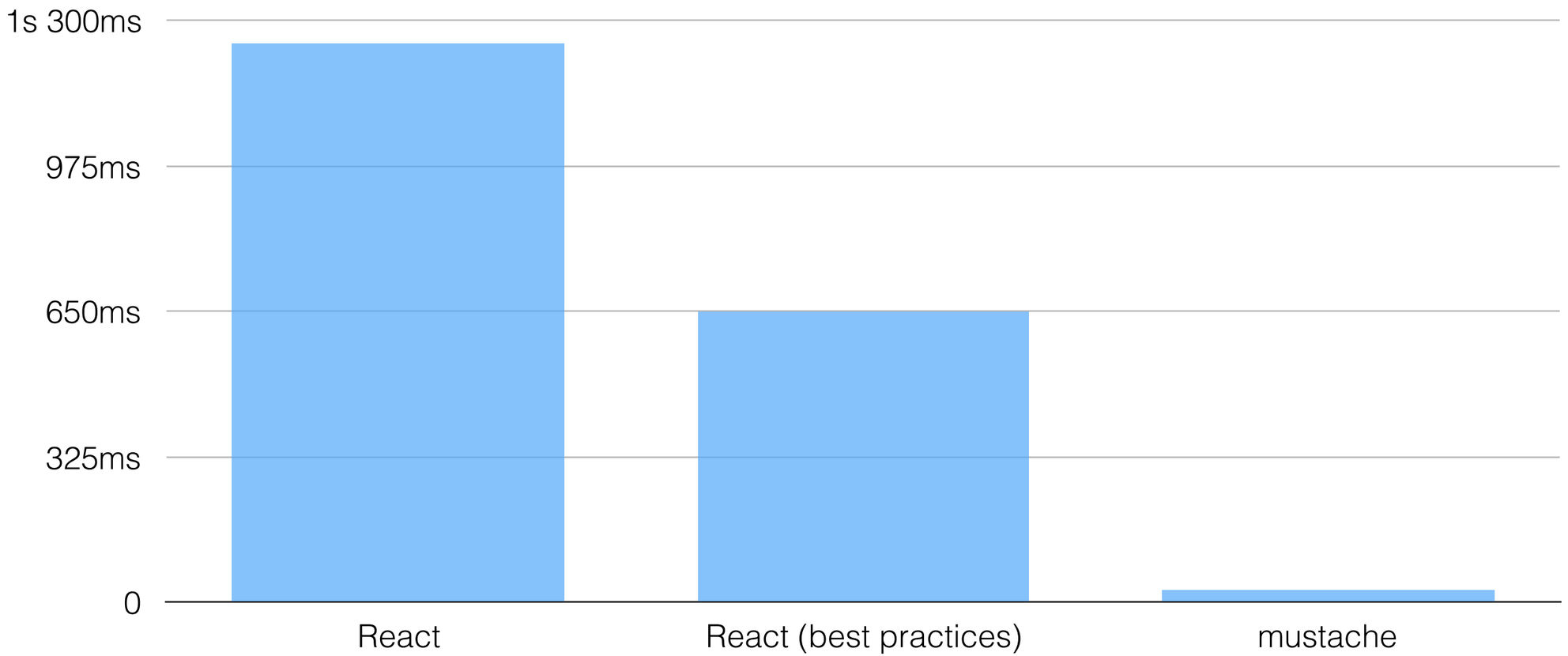
Server-Side **React**

React.js Conf 2015
Q&A with the team

- Facebook: “Funny story about Server Rendering - it wasn’t actually designed that way.” (Sebastian Markbåge)
- Facebook: “we don’t use it that heavily, which is why we haven’t really invested strongly in it.” (Sebastian Markbåge)

https://youtu.be/EPpkboSKvPI?t=7m48s

Server-Side **React**



https://github.com/ndreckshage/react-boston-ssr

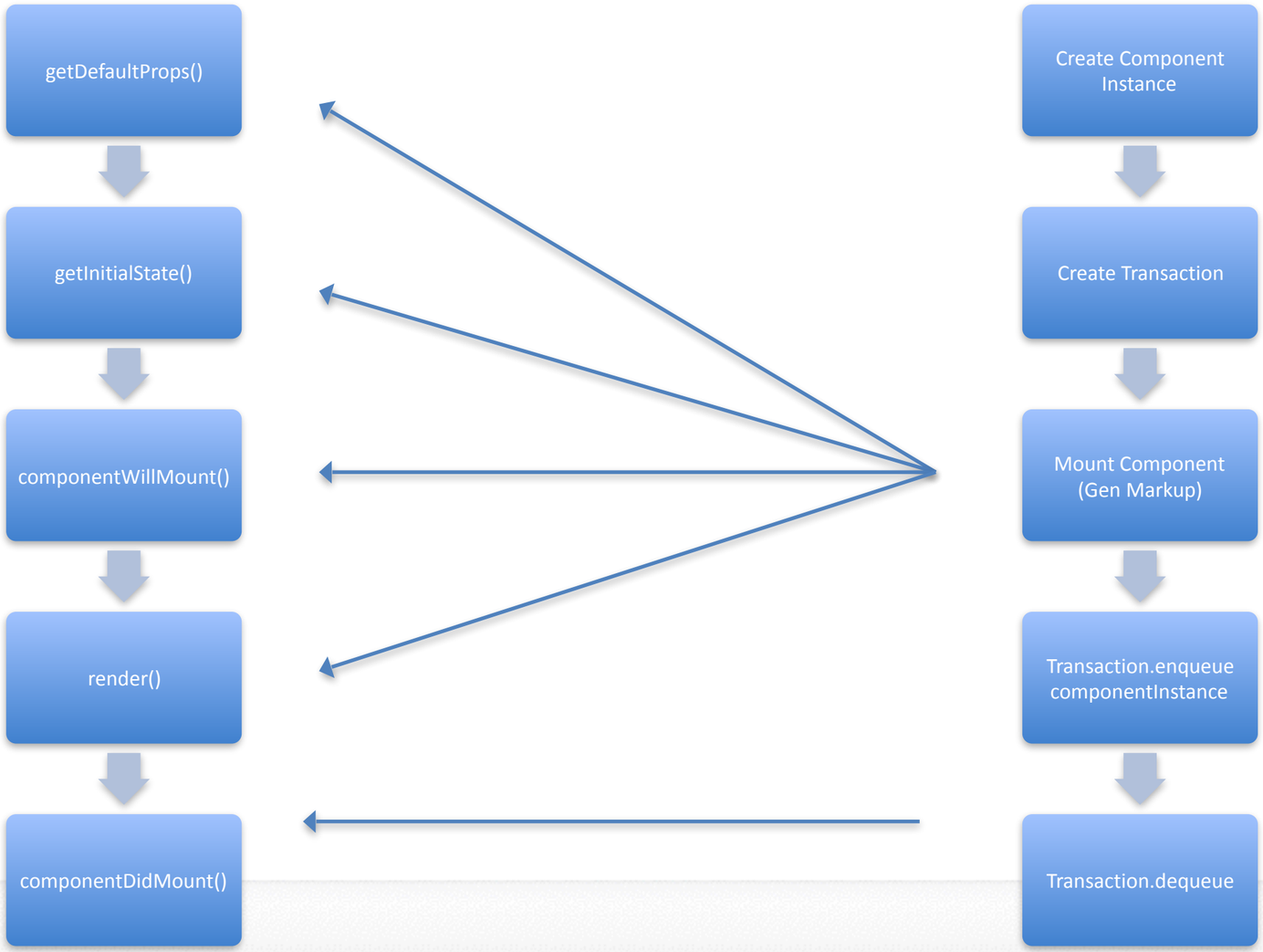
Server-Side **React**

ReactJS SF meetup in January 2016

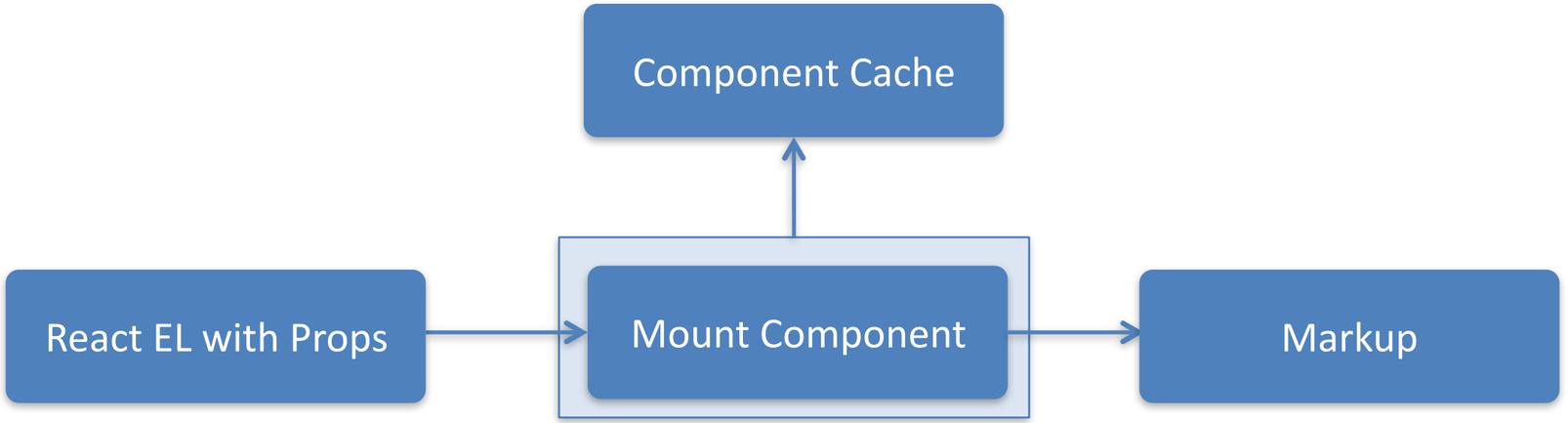
https://github.com/aickin/react-dom-stream

- Looked at *react-dom-stream* and *redfin/react-serve*r to improve TTFB and ATF improvement
- Challenges
  - Forks react and introduces new interfaces, i.e. not “React at heart”
  - Doesn't solve CPU total time

React Rendering **Lifecycle**



Double Clicking into Mount Component



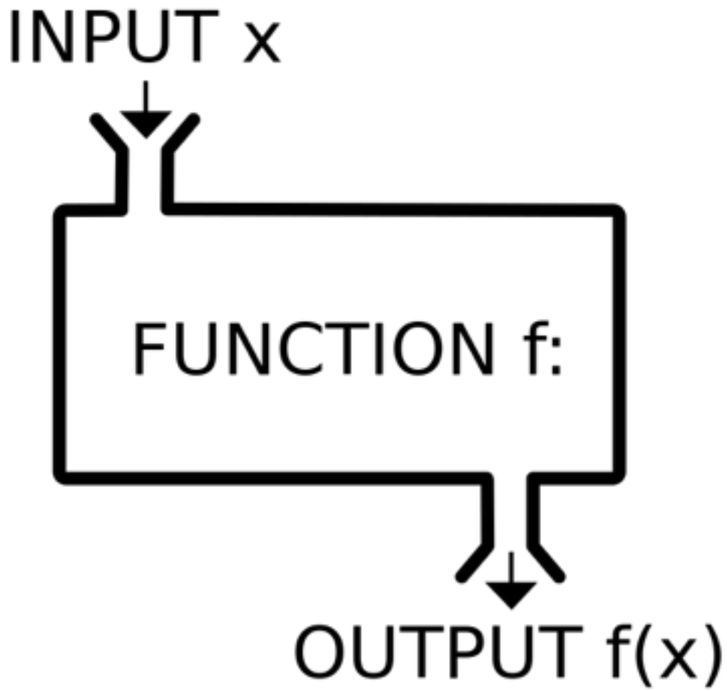
Given a set of properties, the markup generated is always the same

Memoizing Components

The same “text” prop will always return the same helloworld html string

```
class HelloWorldComponent extends React.Component {
  render() {
    return <div>Hello {this.props.text}!</div>;
  }
}
```

When text is “World” the output is:
Hello World!



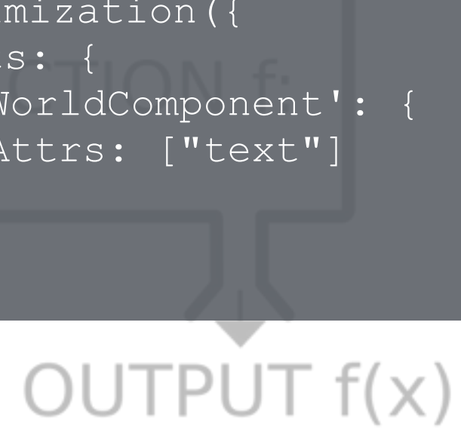
Memoizing Components

The same “text” prop will always return the same helloworld html string

```
class HelloWorldComponent extends React.Component {
  render() {
    return <div>Hello {this.props.text}!</div>;
  }
}
```

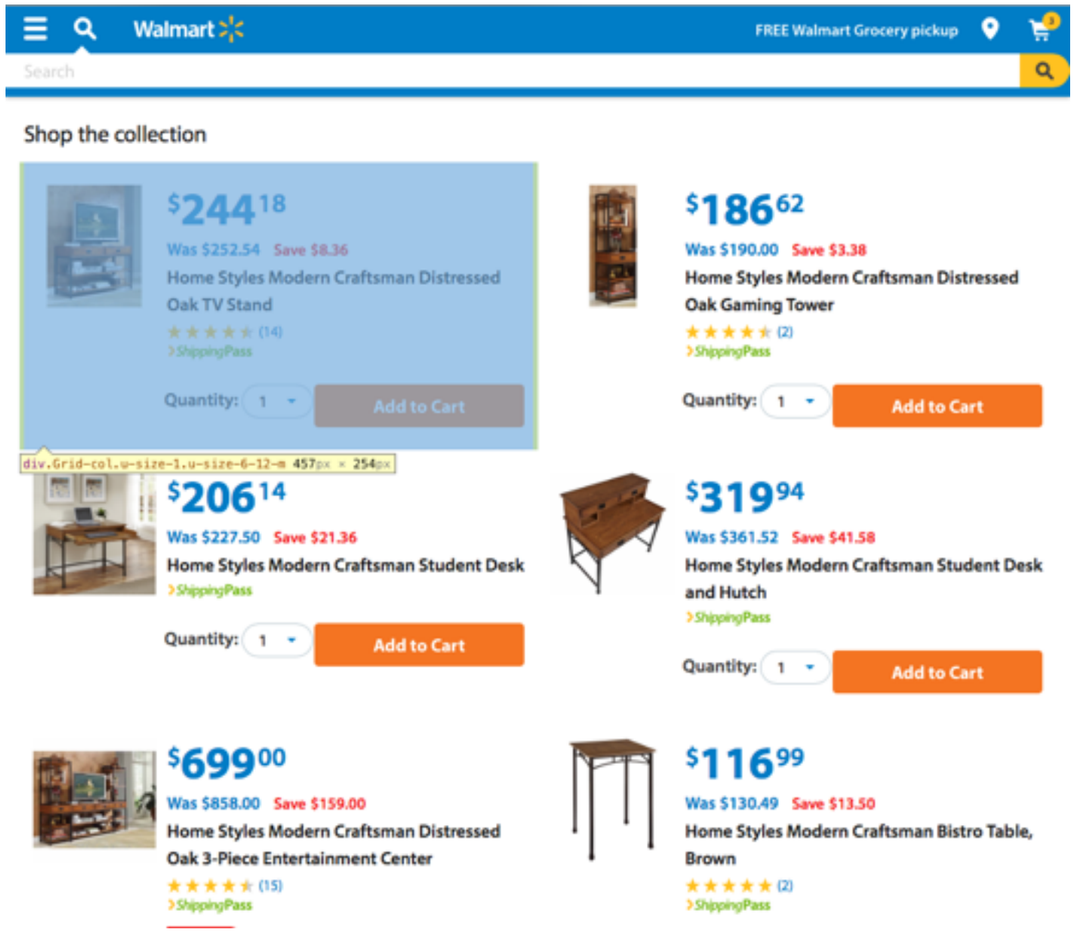
When text is “World” the output is:
Hello World!

```
var componentOptimization =
require("electrode-react-ssr-
optimization");

var componentOptimizationRef =
componentOptimization({
    components: {
      'HelloWorldComponent': {
        keyAttrs: ["text"]
      }
    }
});
```



Memoizing **Demo**

Templatizing Components




```
var React = require('react');

var ProductView = React.createClass({
  render: function() {
    return (
      <div className="product">
        <img src={this.props.product.image}/>
        <div className="product-detail">
          <p className="name">{this.props.product.name}</p>
          <p className="description">{this.props.product.description}
          <p className="price">Price: ${this.props.product.price}</p>
          <button type="button" onClick={this.addToCart}
            Add To Cart
          </button>
        </div>
      </div>
    );
  }
});

module.exports = ProductView;
```

Templatizing Components

- Dynamic props that would be different for each product.
- Switch the corresponding props with template delimiters (i.e. *${ prop_name }*) during react component rendering cycle.
- The template is then compiled, cached, executed and the markup is handed back to React.

```
var React = require('react');

var ProductView = React.createClass({
  render: function() {
    return (
      <div className="product">
        <img src={this.props.product.image}/>
        <div className="product-detail">
          <p className="name">{this.props.product.name}</p>
          <p className="description">{this.props.product.description}
          <p className="price">Price: ${this.props.product.price}</p>
          <button type="button" onClick={this.addToCart}
            Add To Cart
          </button>
        </div>
      </div>
    );
  }
});

module.exports = ProductView;
```

Templatizing **Components**

- Dynamic props that would be different for each product.
- Switch the corresponding props with template delimiters (i.e. *${ prop_name }*) during react component rendering cycle.
- The template is then compiled, cached, executed and the markup is handed back to React.

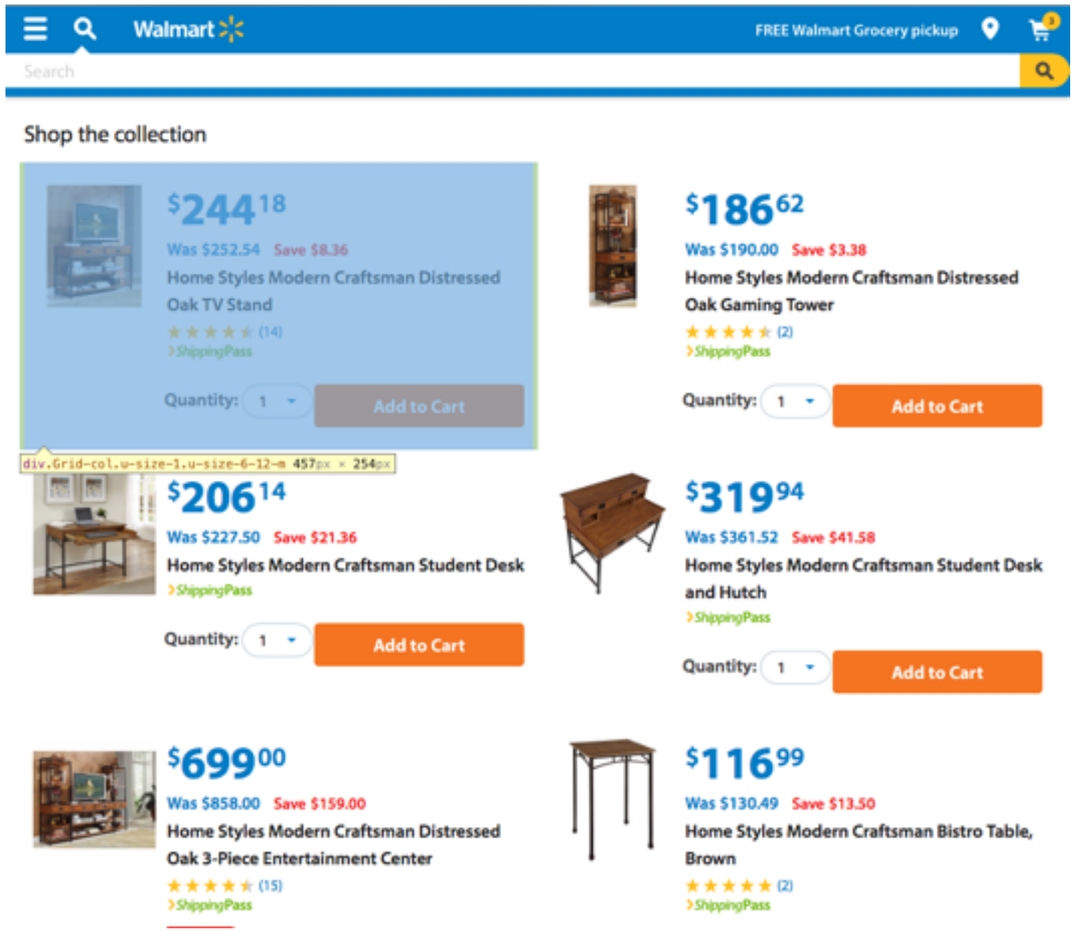
```
<div className="product">
  <img src=${product_image}/>
  <div className="product-detail">
    <p className="name">${product_name}</p>
    <p className="description">${product_description}</p>
    <p className="price">Price: ${selected_price}</p>
    <button type="button" onClick={this.addToCart}>
      Add To Cart
    </button>
  </div>
</div>
```

Templatizing Components

```
var componentOptimization =
require("electrode-react-ssr-
optimization");

var componentOptimizationRef =
componentOptimization({
    components: {
      "ProductView": {
          templateAttrs: ["product.image",
                          "product.name",
                  "product.description",
                      "product.price"]
      },
    }
});
```

```
var React = require('react');

var ProductView = React.createClass({
  render: function() {
    return (
      <div className="product">
        <img src={this.props.product.image}/>
        <div className="product-detail">
          <p className="name">{this.props.product.name}</p>
          <p className="description">{this.props.product.description}
          <p className="price">Price: ${this.props.product.price}</p>
          <button type="button" onClick={this.addToCart}
            Add To Cart
          </button>
        </div>
      </div>
    );
  }
});

module.exports = ProductView;
```

Templatizing Components




```
var React = require('react');

var ProductView = React.createClass({
  render: function() {
    return (
      <div className="product">
        <img src={this.props.product.image}/>
        <div className="product-detail">
          <p className="name">{this.props.product.name}</p>
          <p className="description">{this.props.product.description}
          <p className="price">Price: ${this.props.product.price}</p>
          <button type="button" onClick={this.addToCart}
            {this.props.inventory ? 'Add To Cart' : 'Sold Out'}
          </button>
        </div>
      </div>
    );
  }
});

module.exports = ProductView;
```

Templatizing Components

```
var componentOptimization =
require("electrode-react-ssr-
optimization");

var componentOptimizationRef =
componentOptimization({
    components: {
      "ProductView": {
          templateAttrs: ["product.image",
                          "product.name",
                      "product.description",
                          "product.price"],
          keyAttrs: ["product.inventory"]
      },
    }
});
```

```
var React = require('react');

var ProductView = React.createClass({
  render: function() {
    return (
      <div className="product">
        <img src={this.props.product.image}/>
        <div className="product-detail">
          <p className="name">{this.props.product.name}</p>
          <p className="description">{this.props.product.description}
          <p className="price">Price: ${this.props.product.price}</p>
          <button type="button" onClick={this.addToCart}
            {this.props.inventory ? 'Add To Cart' : 'Sold Out'}
          </button>
        </div>
      </div>
    );
  }
});

module.exports = ProductView;
```

Templatizing Components

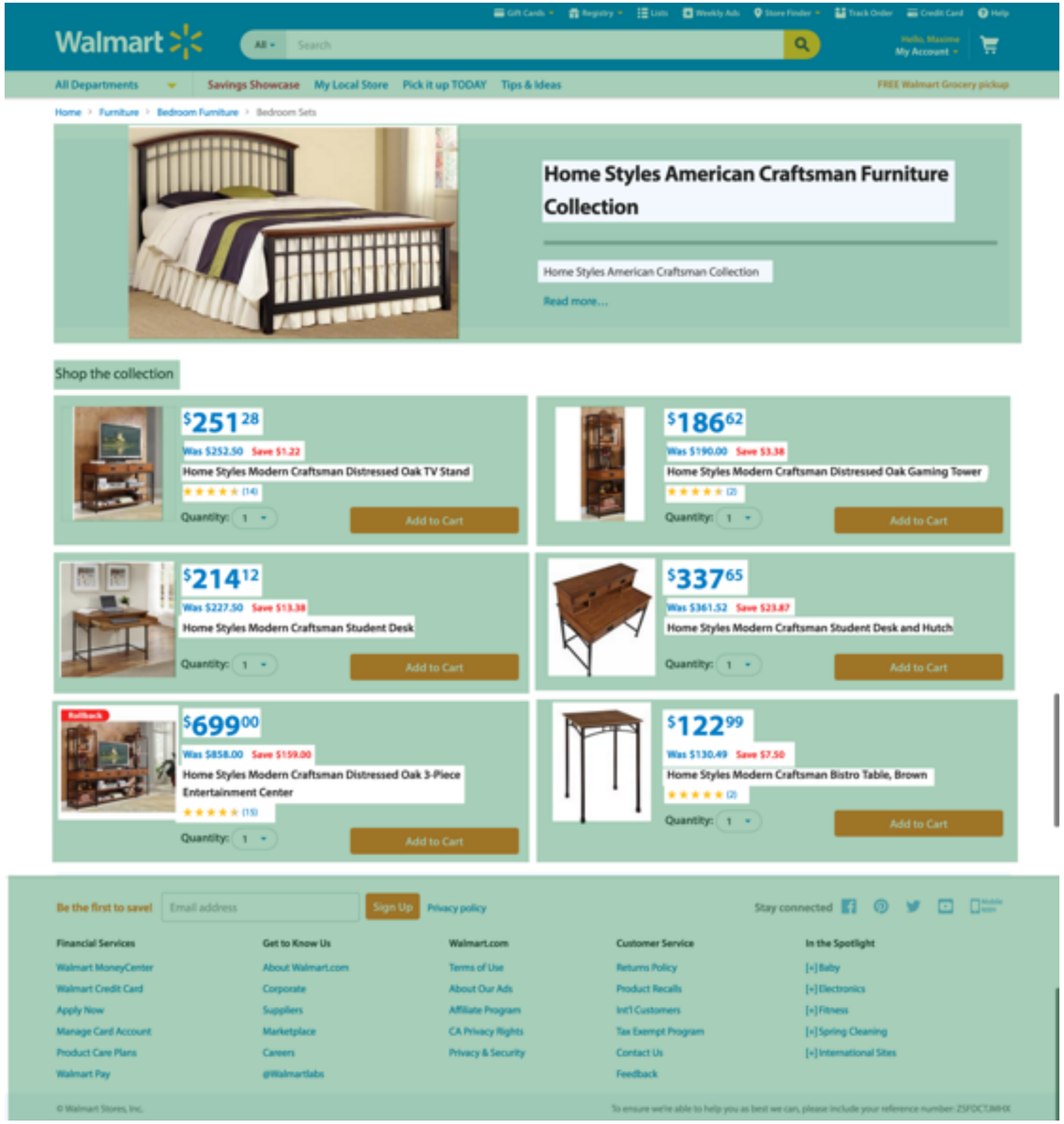
Cached Template 1 - Sold Out

```
<div className="product">
  <img src=${product_image}/>
  <div className="product-detail">
    <p className="name">${product_name}</p>
    <p className="description">${product_description}</p>
    <p className="price">Price: ${selected_price}</p>
    <button type="button" onClick={this.addToCart}>
      Sold Out
    </button>
  </div>
</div>
```

Cached Template 2 - Add To Cart

```
<div className="product">
  <img src=${product_image}/>
  <div className="product-detail">
    <p className="name">${product_name}</p>
    <p className="description">${product_description}</p>
    <p className="price">Price: ${selected_price}</p>
    <button type="button" onClick={this.addToCart}>
      Add To Cart
    </button>
  </div>
</div>
```

Templatizing **Demo**



green blocks indicating markup that was cached on the server

Server-Side **React**

with rendering optimization

CPU profile for one request

Thank You

https://github.com/walmartlabs/electrode-react-ssr-optimization