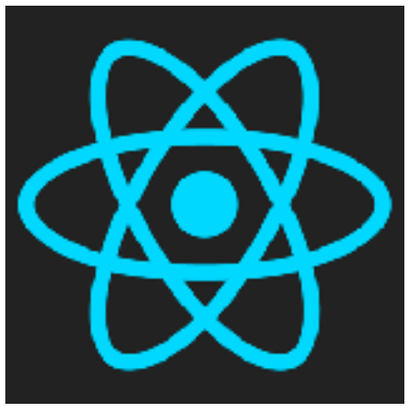
Node.js 组件化 UI 渲染

背景



组件化开发模式成为主流，模版库打入冷宫

模版

```
{{extend ("./parent")}}

{{#block ("head")}}
    <link type="text/css" href="test.css" rev="stylesheet" rel="stylesheet"/>
{{/block}}

{{#block ("body")}}
    <h2>{{title}}</h2>
{{/block}}
```

分离技术

组件化

```
import {createElement, Component, render} from 'rax';
import {Text} from 'rax-components';

class Hello extends Component {
  render() {
    return <Text style={styles.title}>Hello {this.props.name}</Text>;
  }
}

const styles = {
  title: {
    color: '#ff4400',
    fontSize: 48,
    fontWeight: 'bold',
  }
};

render(<Hello name="world" />);
```

分离关注点

单页面的问题

- SEO 弱
- 首屏性能问题

服务器端渲染

- React服务器端渲染：https://facebook.github.io/react/docs/react-dom-server.html

- Rax服务器渲染：https://github.com/alibaba/rax/tree/master/packages/rax-server-renderer

- Vue服务器端渲染：https://vuejs.org/v2/guide/ssr.html

React服务器端渲染

```
import App from ‘./App’;
import React from ‘react’;
import ReactDOMServer from ‘react-dom-server’;

ReactDOMServer.renderToString(React.createElement(App));
```

性能问题

React#renderToString x 169 ops/sec ±2.00% (74 runs sampled)

只有 **169 ops/sec**

初步优化

- 异步化

```
render(<MyComponent {...props} />)
  .toPromise()
  .then(htmlString => console.log(htmlString));
```

- Stream流式输出

```
app.get('/example', function(req, res){
  render(<MyComponent prop="stuff" />)
    .toStream()
    .pipe(res);
});
```

https://github.com/FormidableLabs/rapscallion

进一步：组件级别的缓存

```
const Child = ({ val }) => (
  <div>
    ComponentA
  </div>
);
```

```
const Parent = ({ toVal }) => (
  <div cacheKey={ `Parent:${toVal}` }>
    {
      _.range(toVal).map(val => (
        <Child cacheKey={ `Child:${val}` } key={val} />
      ))
    }
  </div>
);
```

客户端编译为VDOM

```
function render(input, out) {
  var data = input;

  out.be("DIV", {
      style: marko_styleAttr(styles.container)
    }, null, 4);

  out.n(marko_createElement("H2", null, 1, 0, marko_cor
    .t("MarkoList"));

  out.be("DIV", {
      style: marko_styleAttr(styles.list)
    }, null, 4);

marko_forEachProp(data.items, function(idx, item) {
  out.e("A", {
      style: marko_styleAttr(styles.item),
      href: item.url
    }, 3)
    .e("IMG", {
        src: item.img,
        style: marko_styleAttr(styles.itemImg)
      }, 0)
    .e("P", {
        style: marko_styleAttr(styles.itemTitle)
      }, 1, 4)
      .t(item.title)
    .e("P", {
        style: marko_styleAttr(styles.itemPrice)
      }, 1, 4)
```

服务器端编译为模版

```
function render(input, out) {
  var data = input;

  out.w("<div" +
    marko_styleAttr(styles.container) +
    "><h2>MarkoList</h2><div" +
    marko_styleAttr(styles.list) +
    ">");

  marko_forEachProp(data.items, function(idx, item) {
    out.w("<a" +
      marko_styleAttr(styles.item) +
      marko_attr("href", item.url) +
      "><img" +
      marko_attr("src", item.img) +
      marko_styleAttr(styles.itemImg) +
      "><p" +
      marko_styleAttr(styles.itemTitle) +
      ">" +
      marko_escapeXml(item.title) +
      "</p><p" +
      marko_styleAttr(styles.itemPrice) +
      "><span>price: " +
      marko_escapeXml(item.price) +
      "</span></p></a>");
  });

  out.w("</div></div>");
}
```

换个思路　http://markojs.com/

数据对比

React#renderToString x **169** ops/sec ±2.00% (74 runs sampled)

Marko#renderToString x **6,157** ops/sec ±2.08% (80 runs sampled)

Thanks