React Cheatsheet

入门

使用React.js jsfiddle开始Hack。(或者使用非官方的jsbin)

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

ReactDOM.render(<Component name="John" />, document.body);
```

嵌套

使用嵌套组件分离关注点。查看多组件

```
var UserAvatar = React.createClass({ ... });
var UserProfile = React.createClass({ ... });

var Info = React.createClass({
  render() {
    return (
      <div>
        <UserAvatar src={this.props.avatar} />
        <UserProfile username={this.props.username} />
      </div>
    );
  }
});
```

状态和属性

使用props(this.props)访问父类传递过来的参数。使用state(this.state)管理动态数据。

```
<MyComponent fullscreen={true} />

//属性
this.props.fullscreen //=> true

//状态
this.setState({username: 'rstacruz'});
this.replaceState({ ... });
this.state.username   //=> 'rstacruz'

render: function() {
  return (
    <div className={this.props.fullscreen ? 'full' : ''}>
      Welcome, {this.state.username}
    </div>
  );
}
```

设置默认值

提前渲染this.state.comments和this.props.name。

```
React.createClass({
  getInitialState: function () {
    return { comments: []};
  }

  getDefaultProps: function () {
    return { name: "Hello"};
  }
});
```

组件API

这些是组件实例可以访问的方法。查看组件API。

```
ReactDOM.findDOMNode(c) // 0.14+
React.findDOMNode(c)    // 0.13
c.getDOMNode()          // 0.12 以下

c.forceUpdate()
c.isMounted()

c.state
c.props

c.setState({ ... })
c.replaceState({ ... })

c.setProps({ ... })     //废弃
c.replaceProps({ ... }) //废弃

c.refs
```

组件规范

你可以重写方法和属性。查看组件规范

```
render()

getInitialState()

getDefaultProps()

mixins: [ ... ]         混入... [更多](http://ricostacruz.com/cheatsheets/react.html#mixins)

propTypes: { ... }      校验... [更多](http://ricostacruz.com/cheatsheets/react.html#property-validation)

statics: { ... }        静态方法

displayName: "..."      JSX自动填充
```

生命周期

挂载

在初始化渲染之前。在didMount上加一些DOM之类的东西(events, timer之类)。查看参考。

```
componentWillMount()     在渲染之前(还没有DOM)

componentDidMount()      在渲染之后
```

更新

当父类改变属性和调用.setState()时调用。这些不会在初始化渲染的时候调用。查看参考。

```
componentWillReceiveProps (newProps={})                    这里使用setState()

shouldComponentUpdate (newProps={}, newState={})           如果返回false，不会
调用render()

componentWillUpdate (newProps={} newState={})              这里不能使用setSta
te()

componentDidUpdate(preProps={}, preState={})               这里在DOM上操作
```

卸载

这里清理DOM上的一些东西(可能在didMount上已经完成)。查看参考。

```
componentWillUnmount()                                     在DOM移除之前调用
```

示例: 加载数据

查看初始化Ajax数据

```
React.createClass({
  componentWillMount: function () {
    $.get(this.props.url, function (data) {
      this.setState(data);
    }.bind(this));
  }

  render: function () {
    return <CommentList data={this.state.data} />
  }
});
```

DOM节点

参考

允许访问DOM节点。 查看参考。

```
<input ref="myInput" />

this.refs.myInput
ReactDOM.findDOMNode(this.refs.myInput).focus()
ReactDOM.findDOMNode(this.refs.myInput).value
```

DOM事件

增加类似onChange属性。查看事件。

```
<input type="text"
       value={this.state.value}
       onChange={this.handleChange} />

handleChange: function (event) {
  this.setState({value: event.target.value});
}
```

双向绑定

使用LinkedStateMixin轻松实现双向绑定。

```
Email: <input type="email" valueLink={this.linkState('email')} />

React.createClass({
  mixins: [React.addons.LinkedStateMixin]
});

this.state.email
```

属性校验

基础类型

原始类型: .string, .number, .func, 和 .bool。查看propTypes。

```
React.createClass({
  propTypes: {
    email:    React.PropTypes.string,
    seats:    React.PropTypes.number,
    settings: React.PropTypes.object,
    callback: React.PropTypes.func,
    isClosed: React.PropTypes.bool,
    any:      React.PropTypes.any
  }
});
```

必填类型

添加.isRequired。

```
propTypes: {
  requiredFunc: React.PropTypes.func.isRequired,
  requiredAny:  React.PropTypes.any.isRequired
}
```

React元素

使用.element, .node。

```
propTypes: {
  element: React.PropTypes.element,   //react元素
  node:    React.PropTypes.node       //num, string, element
                                      // ...或者它们的数组
}
```

枚举

使用 .oneOf, .oneOfType。

```
propTypes: {
  enum:   React.PropTypes.oneOf(['M', 'F']),  //枚举
  union:  React.PropTypes.oneOfType([         //任何一个
            React.PropTypes.string,
            React.PropTypes.number
          ])
}
```

数组和对象

使用 .array[Of], .object[Of], .instanceOf, .shape。

```
propTypes: {
  array:     React.PropTypes.array,
  arrayOf:   React.PropTypes.arrayOf(React.PropTypes.number),
  object:    React.PropTypes.object,
  objectOf:  React.PropTypes.objectOf(React.PropTypes.number),

  message: React.PropTypes.instanceOf(Message),

  object2: React.PropTypes.shape({
    color: React.PropTypes.string,
    size:  React.PropTypes.number
  })
}
```

自定义验证

提供你自己的函数

```
propTypes: {
  customProp: function (props, propName, componentName) {
    if (!/matchname/.test(props[propName])) {
      return new Error('Validation failed!');
    }
  }
}
```

其他特性

类集合

使用classname操作DOM class。之前都知道使用React.addons.classSet。查看Class set。

```
var cx = require('classname');

render: function () {
  var classes = cx ({
    'message': true,
    'message-important': this.props.isImportant,
    'message-read': this.props.isRead
  });

  return <div className={classes}> Great Scott! </div>;
}
```

扩展属性

查看Transferring props。

```
<VideoPlayer src="video.mp4" />

var VideoPlayer = React.createClass({
  render: function () {
    /* 将 src="..." 属性传递到它的子组件 */
    return <VideoEmbed {...this.props} controls='false' />;
  }
});
```

Mixins

在addons中查看内建mixins。

```
var SetIntervalMixin = {
  componentWillMount: function () {
    ...
  }
}

var TickTock = React.createClass({
  mixins: [SetIntervalMixin]
});
```

顶级API

```
React.createClass({ ... })

React.isValidElement(c)

ReactDOM.findDOMNode(c)     // 0.14+
ReactDOM.render(<Component />, domnode, [callback])  //0.14+
ReactDOM.unmountComponentAtNode(domnode) //0.14+

ReactDOMServer.renderToString(<Component />) //0.14+
ReactDOMServer.renderToStaticMarkup(<Component />) //0.14+
```

JSX模式

Style简写

查看内联样式。

```
var style = {
  backgroundImage: 'url(xxx.jpg)',
  height: 10
};

return <div style={style}></div>;
```

InnerHTML

查看dangerouslySetInnerHTML。

```
function markdownify() {
  return "<p> ... </p>";
}

<div dangerouslySetInnerHTML={{__html: markdownify()}} />
```

列表

```
var TodoList = React.createClass({
  render: function () {
    function item(itemText) {
      return <li>{itemText}</li>;
    }

    return <ul>{this.props.items.map(item)}</ul>;
  }
});
```

可以查看

动画