React 概览

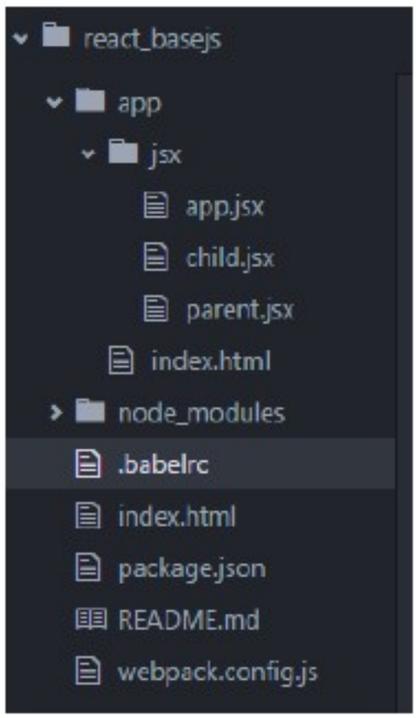
- React 的核心思想是：封装组件，各个组件维护自己的状态和 UI，当状态变更，自动重新渲染整个组件。
- 基于这种方式的一个直观感受就是我们不再需要不厌其烦地来回查找某个 DOM 元素，然后操作 DOM 去更改 UI。
- React 大体包含下面这些概念：
  - 组件
  - JSX
  - Virtual DOM
  - Data Flow

大纲

- 开发环境配置
- JSX
- 组件生命周期
- 事件和属性
- DOM
- 组合组件
- 组件间通信
- Mixins
- 表单
- 打包发布
- 学习资源

开发环境配置

- npm init 创建项目
- 安装 Webpack : npm install –g webpack
- Webpack 使用 webpack.config.js 的配置文件
- package.json 配置文件
- .babelrc babel 配置文件
- js 目录结构：



JSX

- JSX 简单来说就是为了把 HTML 模板直接嵌入到 JS 代码里面，这样就做到了模板和组件关联，但是 JS 不支持这种包含 HTML 的语法，所以需要通过工具将 JSX 编译输出成 JS 代码才能使用。

```
const Child = React.createClass({
    render: function(){
        return (
            <div>
                and this is the <b>{this.props.name}</b>.
            </div>
        )
    }
});
```

组件

- React 组件 两个核心概念
  - props 就是组件的属性，由外部通过 JSX属性传入设置完成。
  - state 是组件的当前状态，可以把组件简单看成一个“状态机”，根据状态 state 呈现不同的UI 展示。一旦状态（数据）更改，组件就会自动调用 render 重新渲染UI; 可以通过 this.setState方法解法

```
const HelloWord = React.createClass({
    render: function(){
        return (
            <div>
                <h1>{this.props.name} <span>欢迎学习React.</span></h1>
            </div>
        )
    }
});
ReactDOM.render(
  <HelloWord name="胡天鑫"/>,
  document.getElementById('app')
);
```

组件生命周期

- 一个组件类必须由调用 React.createClass 创建，并且提供一个 render 方法以及其他可选的生命周期函数、组件相关的事件或方法定义。组件分为三个阶段：实例化、运行时及销会&清理期

首次使用一个组件时，以下这些方法依次被调用:（实例化）

- getDefaultProps
- getInitialState
- componentWillMount
- render
- componentDidMount

当状态改变，以下这些方法依次被调用: (运行时)

- componentWillReceiveProps
- shouldComponentUpdate
- componentWillUpdate
- render
- componentDidUpdate

销毁，当该组件使用完成，componentWillUnmount 方法将会被调用。（销毁）

组件生命周期 之实例化

- getDefaultProps
  - 对于组件，这个方法只会被调用一次，对于父组件指定的props属性，默认设置为该组件的 props 值。
- getInitialState
  - 这个方法只会调用有次，用于初始化组件的state。
- componentWillMount
  - 只会调用一次，在render（渲染）之前调用，这也是在render 方法调用前修改组件 state 的最后一次机会
- render
  - render 方法是必须的。这里会创建虚拟DOM。
- componentDidMount
  - 在 render 方法成功调用并且真实的DOM 已渲染后，可以在componentDidMount 内部通过this.getDOMNode() 来获取相应DOM 节点。

组件生命周期 之运行时

- componentWillReceiveProps
  - 组件在接收到新的props 时候调用，在初始化渲染时不会调用；一般是通过父组件来更改 props
- shouldComponentUpdate
  - 在接收到新的props 或 state 时在渲染之前调用，如果该方法返回false , 则 render() 将不会执行。
- componentWillUpdate
  - 在接收到新的 props 或 state 之前将被调用。
- componentDidUpdate
  - 在渲染成功后将被调用，和componentDidMount 方法类似。

组件生命周期 之销毁&清理期

- componentWillUnmount
  - 在组件从DOM中移除的时候被调用；比如在componentDidMount 执行了setInterval() 方法 ， 在移除组件前可以通过 componentWillUnmount 方法来clearInterval() 方法来结束任务

事件和属性 之事件

- 剪贴板事件
  - onCopy onCout onPaste
- 键盘事件
  - onKeyDown onKeyPress onKeyUp
- 焦点事件
  - onFocus onBlur
- 表单事件
  - onChange onInput onSubmit
- 鼠标事件
  - onClick onContextMenu onDoubleClick onDrag onDragEnd onDragEnter onDragExit
  - onDragLeave onDragOver onDragStart onDrop onMouseDown onMouseEnter onMouseLeave
  - onMouseMove onMouseOut onMouseOver onMouseUp
- 触控事件
  - onTouchCancel onTouchEnd onTouchMove onTouchStart
- UI 事件
  - onScroll
- 滚轮事件
  - onWheel

事件和属性之 属性

- HTML 的属性和标签 在React中大部分都被支持
- class 在 React 中 使用 className 替换

DOM 操作

- findDOMNode()
  - 当组件加载到页面之后，可以通过 ReactDOM.findDOMNode () 方法拿到组件对应的 DOM 元素
- Refs
  - 通过在DOM元素上面设置一个 ref 属性指定一个名称，然后通过 this.refs.name 来访问对应的 DOM 元素。

组合组件

- 使用组件的目的就是通过构建模块化的组件，相互组合组件最后组装成一个复杂的应用。

```
var Avatar = React.createClass({
  render: function() {
    return (
      <div>
        <PagePic pagename={this.props.pagename} />
        <PageLink pagename={this.props.pagename} />
      </div>
    );
  }
});

var PagePic = React.createClass({
  render: function() {
    return (
      <img src={'https://graph.facebook.com/' + this.props.pagename + '/picture'} />
    );
  }
});

var PageLink = React.createClass({
  render: function() {
    return (
      <a href={'https://www.facebook.com/' + this.props.pagename}>
        {this.props.pagename}
      </a>
    );
  }
});
```

组件间通信

- 父子组件间通信
  - 这种情况下很简单，就是通过 props 属性传递，在父组件设置子组件的属性，所以子组件就可以通过 props 或者事件绑定 访问到父组件的方法，这样就搭建起了父子组件间通信的桥梁。

```
var GroceryList = React.createClass({
    handleClick: function(i) {
        console.log('You clicked: ' + this.props.items[i]);
    },

    render: function() {
        return (
            <div>
                {this.props.items.map(function(item, i) {
                    return (
                        <div onClick={this.handleClick.bind(this, i)} key={i}>{item}</div>
                    );
                }, this)}
            </div>
        );
    }
});

ReactDOM.render(
    <GroceryList items={['Apple', 'Banana', 'Cranberry']} />,
      document.getElementById('app')
);
```

Mixins

- Mixins
  - 虽然组件的原则就是模块化，彼此之间相互独立，但是有时候不同的组件之间可能会共用一些功能，共享一部分代码。所以 React 提供了 mixins 这种方式来处理这种问题。
  - Mixin 就是用来定义一些方法，使用这个 mixin 的组件能够自由的使用这些方法（就像在组件中定义的一样），所以 mixin 相当于组件的一个扩展，在 mixin 中也能定义“生命周期”方法。

```
var SetIntervalMixin = {
    componentWillMount: function() {
        this.intervals = [];
    },
    setInterval: function() {
        this.intervals.push(setInterval.apply(null, arguments));
    },
    componentWillUnmount: function() {
        this.intervals.map(clearInterval);
    }
};

var TickTock = React.createClass({
    mixins: [SetIntervalMixin], // Use the mixin
    getInitialState: function() {
        return {seconds: 0};
    },
    componentDidMount: function() {
        this.setInterval(this.tick, 1000); // Call a method on the mixin
    },
    tick: function() {
        this.setState({seconds: this.state.seconds + 1});
    },
    render: function() {
        return (
            <p>
                React has been running for {this.state.seconds} seconds.
            </p>
        );
    }
});

ReactDOM.render(
    <TickTock />,
```

表单

- 状态属性
- 表单元素有这么几种属于状态的属性
  - value，对应 <input> 和 <textarea> 所有
  - checked，对应类型为 checkbox 和 radio 的 <input> 所有
  - selected，对应 <option> 所有
- 受控组件
  - 对于设置了上面提到的对应“状态属性“值的表单元素就是受控表单组件，比如：

```
render: function() {
    return <input type="text" value="hello"/>;
}
```

- 非受控组件
  - 和受控组件相对，如果表单元素没有设置自己的“状态属性”，或者属性值设置为 null，这时候就是非受控组件。

打包发布

- 使用 webpack –p 进行打包

资源

- React 官网 https://facebook.github.io/react/index.html
- React 中文社区： http://react-china.org/
- React 中文文档
  - 中文文档地址 http://reactjs.cn
  - GitHub地址 https://github.com/reactjs-cn/react-docs
  - 阮一峰的 "React 入门实例教程" http://www.ruanyifeng.com/blog/2015/03/react.html
  - 极客学院整理的文档 http://wiki.jikexueyuan.com/project/react/
- React 入门教程 http://hulufei.gitbooks.io/react-tutorial/
- React 视频 http://yun.baidu.com/s/1pJJqYWN
- React UI
  - Material-UI http://www.material-ui.com/
  - React ui https://github.com/Lobos/react-ui
  - ANT DESIGN UI http://ant.design/
  - React-Bootstrap http://react-bootstrap.github.io/
  - **Amaze UI React http://amazeui.org/react/**