积云-React实战-基础篇

一、为什么要学习React?

1）前言

选择任何一门语言学习都是有时间和金钱的成本的，那么React值不值得学习呢?

2）学习的必要性?

1) 使用组件化开发方式，符合现代Web开发的趋势；企业前后端项目分离，唯有React是首选；

2) 技术成熟，社区完善，配件齐全，适用于大型Web项目（生态系统健全）

3) 由Facebook专门的团队维护，技术支持可靠;

4) ReactNative - Learn once, write anywhere: Build mobile apps with React

5) 使用方式简单，性能非常高，支持服务端渲染;

6) React使用前端技术非常全面，有利于整体提高技术水平；此外，有利于求职和晋升，有利于参与潜力大的项目。

二、React快速入门

1）概念介绍

React 起源于 Facebook 的内部项目，因为该公司对市场上所有 JavaScript MVC 框架，都不满意，就决定自己写一套，用来架设他们自己的 Instagram 的网站。

做出来以后，发现这套东西很好用，在2013年5月开源了。目前已经成为前端的三大主流框架。

2）React是什么?

React是用于构建用户界面的JavaScript 库，围绕React的技术栈主要包括：React, redux, react-redux, react-router, ...

官网

React官网：https://reactjs.org/

React中文：https://react.docschina.org/

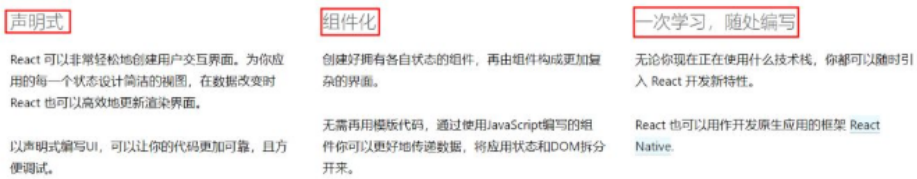
3）React具备的特点

1）使用JSX语法创建组件，实现组件化开发，为函数式的 UI 编程方式打开了大门;

2）性能高：通过 diff算法 和 虚拟DOM 实现视图的高效渲染和更新;

3）图示

JSX --> HTML
JSX --> native ios或android中的组件（XML）
JSX --> VR/AR
JSX --> 区块链/物联网
JSX --TO--> EveryThing



4）React核心

1）虚拟DOM

1）概念

React将DOM抽象为虚拟DOM，虚拟DOM其实就是用一个对象来描述DOM，通过对比前后两个对象的差异，最终只把变化的部分重新渲染，提高渲染的效率

2）为什么用虚拟DOM?

当DOM发生更改时需要遍历DOM对象的属性，而原生DOM可遍历属性多达200多个个，而且大部分属性与渲染无关，导致更新页面代价太大。

3）虚拟DOM的处理方式?

1） 用 JS对象结构表示 DOM 树的结构，然后用这个树构建一个真正的 DOM 树，插到文档当中。

2）当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异。

3） 把记录的差异应用到步骤1所构建的真正的DOM树上，视图就更新了。

2）Diff算法

1）概念

最小化页面重绘

当我们使用React在render() 函数创建了一棵React元素树，在下一个state或者props更新的时候，render() 函数将会创建一棵新的React元素树。

React将对比这两棵树的不同之处，计算出如何高效的更新UI（只更新变化的地方），此处所采用的算法就是diff算法。

2）后续讲解

三、React初体验

1）小demo：你好，积云学院

1）类库引入

2）代码编写

3）配置react-developer-tools开发调试工具插件

4）图示

2）JSX语法全面入门

Demo1

往容器中插入一个span标签, class为: "it-like", 内容为: "积云学院"。 两种实现方式: a) 通过典型js方式; b) JSX方式?

总结

1. JSX只是高级语法糖, 最终执行时还是会被转成原生js, 通过babel等方式;

2. 更加语义化, 更加直观, 代码可读性更高;

3. 性能相对原生方式更加好一些!

Demo2

JSX常见的界面操作方式?

主体

a. 多重标签嵌套

```
ReactDOM.render(
    <div>
        <h2>多层标签嵌套</h2>
        <img src="img/logo.jpg" alt="" width="200"/>
        <p>嵌套多个标签时，需要使用一个大标签(比如:div)将多层标签包裹起
    </div>,
    document.getElementById("app")
);
```

b. js中的变量, 表达式要写在{}内

```
ReactDOM.render(
    <div>
        <p>100+20*2/5 = {100 + 20 * 2 / 5}</p>
    </div>,
    document.getElementById("app")
)
```

c. 内联样式通过对象方式引入

```
const myStyle = {
    backgroundColor: 'purple',
    color: 'yellow',
    fontSize: 30
};
ReactDOM.render(
    <div>
        <h2>来点颜色</h2>
        <div style={myStyle}>客官，颜色来了</div>
    </div>,
    document.getElementById("app")
);
```

d. 数组遍历

```
// 1. 数据
const dataArr = [
    {name: '周杰伦', age: 38},
    {name: '谢霆锋', age: 37},
    {name: '刘德华', age: 48},
    {name: '前端学院', age: 2},
];
// 2. 创建虚拟DOM
const vDom = (<ul>
    {dataArr.map((data, index) => <li>{index + 1} - 姓名：{data.name}，年龄：{data.age}岁</li>)}
</ul>);
// 3. 渲染虚拟DOM
ReactDOM.render(vDom, document.getElementById('app'));
```

总结

1. JSX中添加属性时，使用 className 代替 class，像label中的for属性，使用htmlFor代替；

2. JSX创建DOM的时候，所有的节点，必须有唯一的根元素进行包裹；

3. JSX语法中，标签必须成对出现，如果是单标签，则必须自闭合；

4. 在 JSX 中可以直接使用 JS代码，直接在 JSX 中通过 {} 中间写 JS代码即可；

5. 在 JSX 中 {} 里面只能使用表达式，不能出现语句；

6. 在 JSX 中注释语法：{/* 中间是注释的内容 */}

四、React中的组件/模块, 组件化/模块化

1）基本概念

1）组件

一个应用/版块/页面中用于实现某个局部的功能(包括html, js, css等)

把这些局部功能组装到一起就形成了完整的一个大的功能

主要目的在于：复用代码, 提高项目运行效率。

2）组件化

如果一个应用是用多组件的方式进行综合开发的, 那么这个应用就是一个组件化应用。

3）模块

多个组件形成模块, 或者是一个提供特定功能的js文件，主要特点在于耦合性低, 可移植性高, 执行效率好。

4）模块化

如果一个应用都是用模块的形式来构建的，那么这个应用就是模块化应用。

PS: 对照现实生活，本地 和 本地化

2）React中组件创建方式

1）构造函数创建组件

使用构造函数来创建组件时，如果要接收外界传递的数据，需要在构造函数的参数列表中使用props来接收；

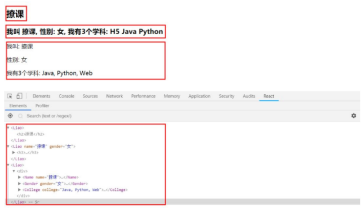
必须要向外return一个合法的JSX创建的虚拟DOM；

Demo: 简单组件 / 带参数组件 / 复合组件

核心代码

```
// 1. 简单使用
function Liao() {
    return <h2>撩课</h2>
}
// 2. 可以传递参数
function Liao1(props) {
    return <h3>我叫：{props.name}，性别：{props.gender}， 我有3个学科：{props.college}</h3>;
}
// 3. 复合组件
function Name(props) {  return <p>我叫：{props.name}</p>; }
function Gender(props) { return <p>性别：{props.gender}</p>; }
function College(props) { return <p>我有3个学科：{props.college}</p>;}
function Person(){
    return (
        <div>
            <Name name="撩课" />
            <Gender gender="女" />
            <College college="Java, Python, Web" />
        </div>
    );
}
```

运行效果



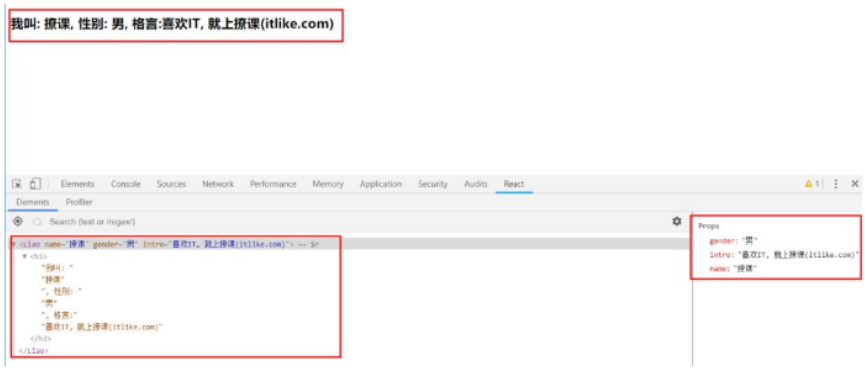
2）class关键字创建组件

核心代码

```
// ES6类方式创建
class Liao extends React.Component {
    render() {
        return <h3>我叫：{this.props.name}，性别：{this.props.gender}，格言:{this.props.intro}</h3>
    }
}

ReactDOM.render(
    <Liao name="撩课" gender="男" intro="喜欢IT，就上撩课(itlike.com)"/>,
    document.getElementById("app")
);
```

运行结果



3）区别

1）构造函数创建的组件叫：无状态组件；

2）class关键字创建的组件叫：有状态组件；

3）有状态组件与无状态组件的本质区别在于是否有

state（状态）属性。

五、React中的state(状态)

1）什么是state?

React 把组件看成是一个状态机（State Machines），通过状态 (State) 去操作状态机。

在开发中, 通过与用户的交互，实现不同状态，然后渲染 UI，让用户界面和数据保持一致。

在React 中，只需更新组件的 state，然后根据新的 state 重新渲染用户界（不要操作 DOM）。

2）state的使用方式?

Demo1：更改界面内容

- 核心代码
- 运行结果
- 点击后效果

六、props和state混合使用

Demo: 定义一个组件描述一条狗

1）需求

1) 分析props和state的使用场景?

2) props的使用细节?

3) 单组件中, 两者的区别?

2）核心代码

a）定义props和state

```
// 1. 创建组件类
class Dog extends React.Component{
    constructor(props){
        super(props);
        // 2. 设置state
        this.state = {
            age: 1, // 年龄
            dogFriends: [] // 狗友
        }
    }
    // 3. 设置props属性的默认值
    static defaultProps ={};
    // 4. 设置props属性的类型
    static propTypes ={
        name: PropTypes.string.isRequired, // 姓名
        gender: PropTypes.string.isRequired, // 性别
    };
```

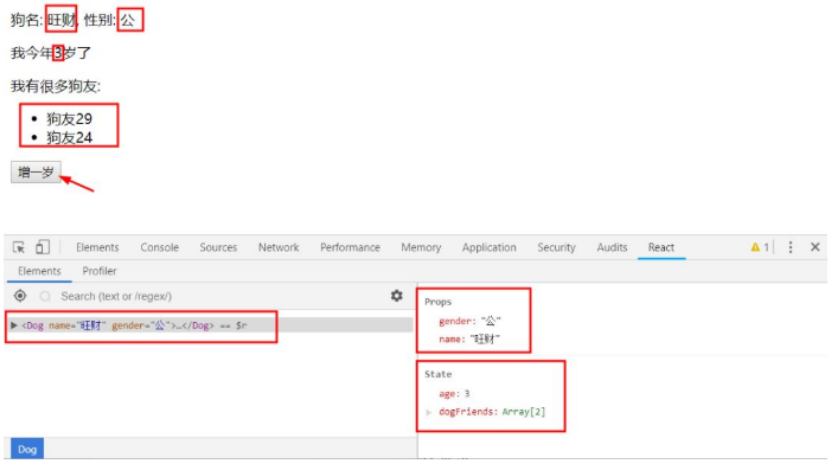
b) 渲染页面

```
render(){
    const {name, gender} = this.props;
    const {age, dogFriends} = this.state;
    return (
        <div>
            <p>狗名：{name}，性别：{gender}</p>
            <p>我今年{age}岁了</p>
            <p>我有很多狗友：</p>
            <ul>
                {dogFriends.map((friend, index)=> <li key={index}>{friend}</li>)}
            </ul>
            <button onClick={this.addYear.bind(this)}>增一岁</button>
        </div>
    )
}
```

c) 业务处理

```
addYear(){
    // 增加狗友
    let tempArr = this.state.dogFriends;
    tempArr.push('狗友' + Math.floor(Math.random() * 100));
    this.setState({
        age: this.state.age += 1,
        dogFriends: tempArr
    })
}
```

d) 运行结果



3）案例总结

1）在单组件中, props一般用于接收外部传入的数据; 而state则用于记录组件内部数据，而且是需要经常改变的数据。

2）state是组件内部的状态（数据），不能够直接修改，必须要通过setState来改变值的状态，从而达到更新组件内部数据的作用。

3）props更多是组件间传递数据的一种方式，props同样也可以传递state。由于React的数据流是自上而下的，所以是从父组件向子组件进行传递；另外组件内部的this.props属性是只读的不可修改。

七、ref的使用

1）定义

Refs 提供了一种方式，用于访问在 render 方法中创建的 DOM 节点或 React 元素。

2）使用场景

a) 处理焦点、文本选择或媒体控制。

b) 触发强制动画。

c) 集成第三方 DOM 库。

3）注意

官方提示, 如果可以通过声明式实现，则尽量避免使用 refs。话外音：React无法控制局面的时候，就需要直接操作Refs了。

4）案例使用

让输入框获得焦点?

核心代码

```
class CustomTextInput extends React.Component {
    constructor(props) {
        super(props);
        // 绑定ref
        this.myInput = React.createRef();
        this.myBtn = React.createRef();
        this.focusTextInput = this.focusTextInput.bind(this);
    }
    render() {
        return (
            <div>
                <input type="text" ref={this.myInput} />
                <input type="button" ref={this.myBtn} value="获取焦点" onClick={this.focusTextInput}/>
            </div>
        );
    }
    focusTextInput() {
        this.myInput.current.focus(); // 获得焦点
        console.log(this.myBtn);
    }
}
```

八、多组件应用

1) 案例1

Demo: 完成对学生信息的展示/添加/删除?

步骤一

1）静态组件拆解, 核心代码

2）核心代码

```
// 1. 创建组件
// 父组件
class App extends React.Component {
    render() {
        return (
            <div>
                <Add />
                <List />
            </div>
        )
    }
}
// 子组件1 -- 添加版块
class Add extends React.Component {...}
// 子组件2 -- 展示版块
class List extends React.Component {...}
```

3）静态组件运行效果

步骤二

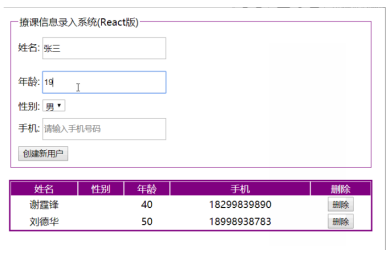
1）动态效果实现(添加/删除/显示), 核心代码如下

2）添加和删除部分

```
/** 插入新的学生记录 ...*/
addToArr(student){
    // 1. 插入
    const {studentArr} = this.state;
    studentArr.unshift(student);
    // 2. 更新数据
    this.setState({
        studentArr
    });
}

/** 删除一条学生记录 ...*/
delFromArr(index){
    // 1. 删除
    const {studentArr} = this.state;
    studentArr.splice(index, 1);
    // 2. 更新数据
    this.setState({
        studentArr
    });
}
```

3）案例运行效果



总结

1. 多层组件中, 数据该放在何处?

   如果只用于一个组件, 则定义在该组件内容; 如果是运用于多个组件, 则定义在它们的父组件中。

2. 多层组件中, 数据传递问题?

   父组件通过props传递数据给子组件, 子组件不能直接修改父组件的数据, 必须是父组件定义修改数据的函

数, 传递给子组件然后由子组件调用。

静态代码

样式

结构

2) 案例2

Demo：用户名和密码获取?

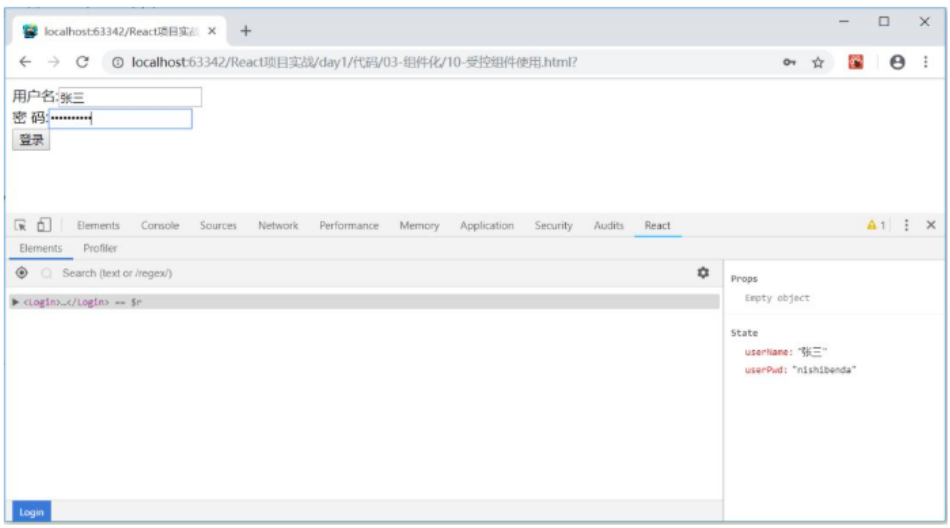
核心代码

```
render(){
    return(
        <form action="" onSubmit={(e)=>this.submitLogin(e)}>
            <div>
                <span>用户名:</span>
                <input type="text" value={this.state.userName} onChange={(event)=>this.userChange(event)}/>
            </div>
            <div>
                <span>密 码:</span>
                <input type="password" value={this.state.userPwd} onChange={(event)=>this.pwdChange(event)}/>
            </div>
            <input type="submit" value="登录" />
        </form>
    )
}
```

```
userChange(event){
    console.log(event);
    // 1. 读取输入的用户名
    const userName = event.target.value;
    // 2. 更新状态
    this.setState({userName})
}
```

```
pwdChange(event){
    // 1. 读取输入的用户名
    const userPwd = event.target.value;
    // 2. 更新状态
    this.setState({userPwd})
}
submitLogin(event){
    const {userName, userPwd} = this.state;
    alert(`用户名:${userName}, 密码:${userPwd}`);
    return;
}
```

界面效果



3）案例总结

开发中尽可能用受控组件, 因为ref方式官方不希望过渡使用。