第十二周

React & Redux & Router

主讲：曾亮

JS
Full Stack

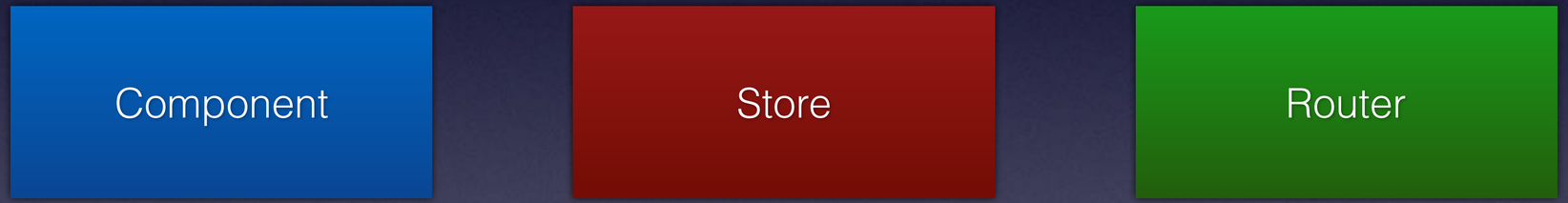
课程诞生原因

- 直播中的 UIBase， 启发组件之思。
- 更完整的工具库需求
- 应对复杂前端程序

如何应对复杂前端应用

- 组件化
- 可重用
- 脱离底层 DOM
- 路由页面分割
- 数据与组件分离

React 技术模块



开发环境

- 安装需要的库
- 配置 webpack
- 配置 package.json

安装需要的库

- npm i babel-loader babel-preset-env babel-preset-react webpack webpack-cli webpack-dev-server babel-core react react-dom react-router-dom redux react-redux

配置 webpack

- https://github.com/liangzeng/qa/blob/master/react-webpack/webpack.config.js

配置 package.json

```
"scripts": {
  "dev": "webpack-dev-server --open"
}
```

Component

- 创建
- 外界只读数据 props
- 组件内在自身状态数据 state
- 事件
- 渲染

第一个 React 组件开发

- const React = require(“react”); // 导入 react
- const ReactDom = require(“react-dom”); // 导入 react-dom
- class List extends React.Component // 继承
- render(){ … } // 编写渲染函数
- ReactDom.render(<List /> , document.body); // 组件渲染到网页

拆分组件 化整为零

- Item 组件
- List 组件

外部只读数据 props

- 不要更改 props！
- super(props)　//　this.props
- 根据 props 渲染
- 看例子 🌰

静态组件 动态组件

- 静态组件：只用 props
- 动态组件：有自己的状态 state

有状态组件的渲染

- this.state 可以读写
- this.setState({ … }) 更改状态，会触发 render 重绘。
- 看栗子  (setTimeout 渲染)

组件 render 渲染函数

- render函数返回值，作为当前渲染结果。
- 当调用 setState() 函数时，会触发 render 函数。
- 不可有过多逻辑。
- { 表达式 }

事件处理

- onXXX = { this.handle } // 由自身方法 handle 处理
- this.handle = this.handle.bind(this); // 绑定当前组件
- handle(event){} // event 和 jquery event所具有的属性基本相同
- 看栗子 

自定义事件

- 调用者 <List onItemSelected={ handle } />
- List 内部触发 this.props.onItemSelected(…)

key 标签属性

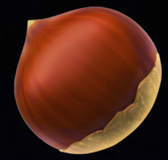
- 列表结构需要为 item 提供 key 属性
- 此key属性值，在当前组不重复
- 不应用数组 index 作为 key
- 可以用 uuid 库实现唯一 key

className & style

- className 代替 class
- style 的值是个对象 { backgroundColor: “谁色啦？ ” }

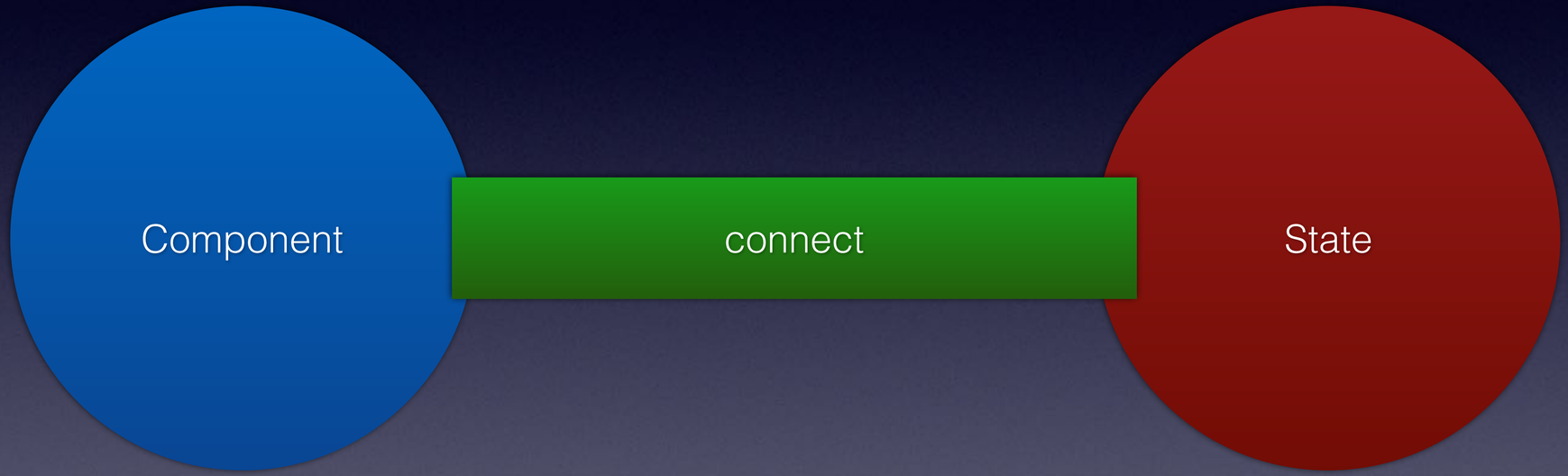
表单元素的 value

- 表单元素的值通过 value 属性指定。

小栗子  s...

- tab 切换 
- 左右互斥 

UI 组件与数据分而治之



Store

- const store = createStore(updater); // 内部有一个 state 状态数据

updater

```
function updater(state,event){

    ……

    return newState;

}
```

Actions

```
function actions(dispatch) {
    return {
        action1(){},
            action2(){
                dispatch({ type: “add” });

            }

            ……
    }
}
```

get_sub_state

state : {list, articles, users}

```
function get_sub_state(state){
    return {
        list : state.list
    }
}
```

connect

- const NewList = connect(get_sub_state , actions)(List);

把这些合在一起 话糙理不糙

```
ReactDOM.render(
  <Provider store={store}>
    <NewList  />
  </Provider>,
document.body
)
```

理论墨迹废话看完了，看实际演示 🌰

路由

- Router
- Link
- Route
- 不难，看 

通关作业

- Todo 项目实战