React入门

课堂目标

1. 学习react基础语法
2. 熟悉官方create-react-app脚手架
3. 掌握JSX语法
4. 掌握setState
5. 掌握react生命周期
6. 掌握props传递参数
7. 掌握React组件通信

##

资源

1. react
2. create-react-app

起步

上手

1. `npm install -g create-react-app` 安装官方脚手架
2. create-react-app react01 初始化
3. react的api比较少，基本学一次，就再也不用看文档了，核心就是js的功力
4. demo体验

文件结构

```
├── README.md                    文档
├── package-lock.json
├── package.json                 npm 依赖
├── public                       静态资源
|   ├── favicon.ico
|   ├── index.html
|   └── manifest.json
└── src                          源码
    ├── App.css
    ├── App.js
    ├── App.test.js              测试
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── serviceWorker.js         pwd的支持
```

###

React和ReactDom

删除src下面所有代码，新建index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';

ReactDOM.render(<App />, document.querySelector('#root'));
```

新建App.js

```
import React from 'react'

class KaikebaCart extends React.Component{
  render(){
    return <div>
      <button>雷猴啊</button>
    </div>
  }
}

export default KaikebaCart
```

React设计之初，就是使用JSX来描述UI，所以解耦和dom操作，react只做逻辑层，reactDom去渲染实际的dom，如果换到移动端，就用别的渲染库

JSX

上面的代码会有一些困惑的地方，首先就是JSX的语法

```
ReactDOM.render(<App />, document.querySelector('#root'));
```

看起来是js和html的混合体，被称之为JSX，实际核心的逻辑完全是js实现的

State和setState

在APP组件里，我们可以通过{}在jsx中渲染变量

```
import React from 'react'

class KaikebaCart extends React.Component{
  render(){
    const name = '开课吧'
    return <div>
      <button>{name}</button>
    </div>
  }
}

export default KaikebaCart
```

如果数据需要修改，并且同时页面响应变化，我们需要放在state中，并且使用setState来修改数据

```
import React from 'react'

class KaikebaCart extends React.Component{
  constructor(props){
    super(props)
    this.state={
      name:'开课吧'
    }
```

```
    setTimeout(()=>{
      this.setState({
        name:'react真不错'
      })
    },2000)
  }
  render(){
    return <div>
      <button>{this.state.name}</button>
    </div>
  }
}

export default KaikebaCart
```

我们不能使用`this.state.name='react'`而是要用this.stateState,设置一个新的state，对数据进行覆盖

Props属性传递

```
ReactDOM.render(<App title="开课吧真不错" />, document.querySelector('#root'));

...


<h2>{this.props.title}</h2>
```

JSX实质

jsx实质就是React.createElement的调用

```
class App extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}, I am {2 + 2} years old
      </div>
    )
  }
}

ReactDOM.render(
  <App name="React" />,
  mountNode
)
```

REACT 编辑器 显示JSX

```
class App extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name,
      ", I am ",
      2 + 2,
      " years old"
    )
  }
}

ReactDOM.render(React.createElement(App, { name: "React" }),
mountNode)
```

条件渲染和循环

React的api不多，条件渲染和循环，都是普通的js语法

```
import React from 'react'
```

```
class KaikebaCart extends React.Component{
  constructor(props){
    super(props)
    this.state={
      name:'开课吧',
      showTitle:true,
      goods: [
        { text: '百万年薪架构师', price: 100, id:1 },
        { text: 'web全栈架构师', price: 80 ,id:2},
        { text: 'Python爬虫', price: 60 ,id:3}
      ]
    }
    setTimeout(()=>{
      this.setState({
        showTitle:false
      })
    },2000)
  }
  render(){
    return <div>
      {this.state.showTitle && <h2>{this.props.title}</h2>}
      <ul>

      {this.state.goods.map(good=>{
        return <li key={good.id}>
          <span>{good.text}</span>
          <span>{good.price}</span>元

        </li>
      })}

      </ul>
      <button>{this.state.name}</button>
    </div>
  }
}

export default KaikebaCart
```

class VS 函数组件

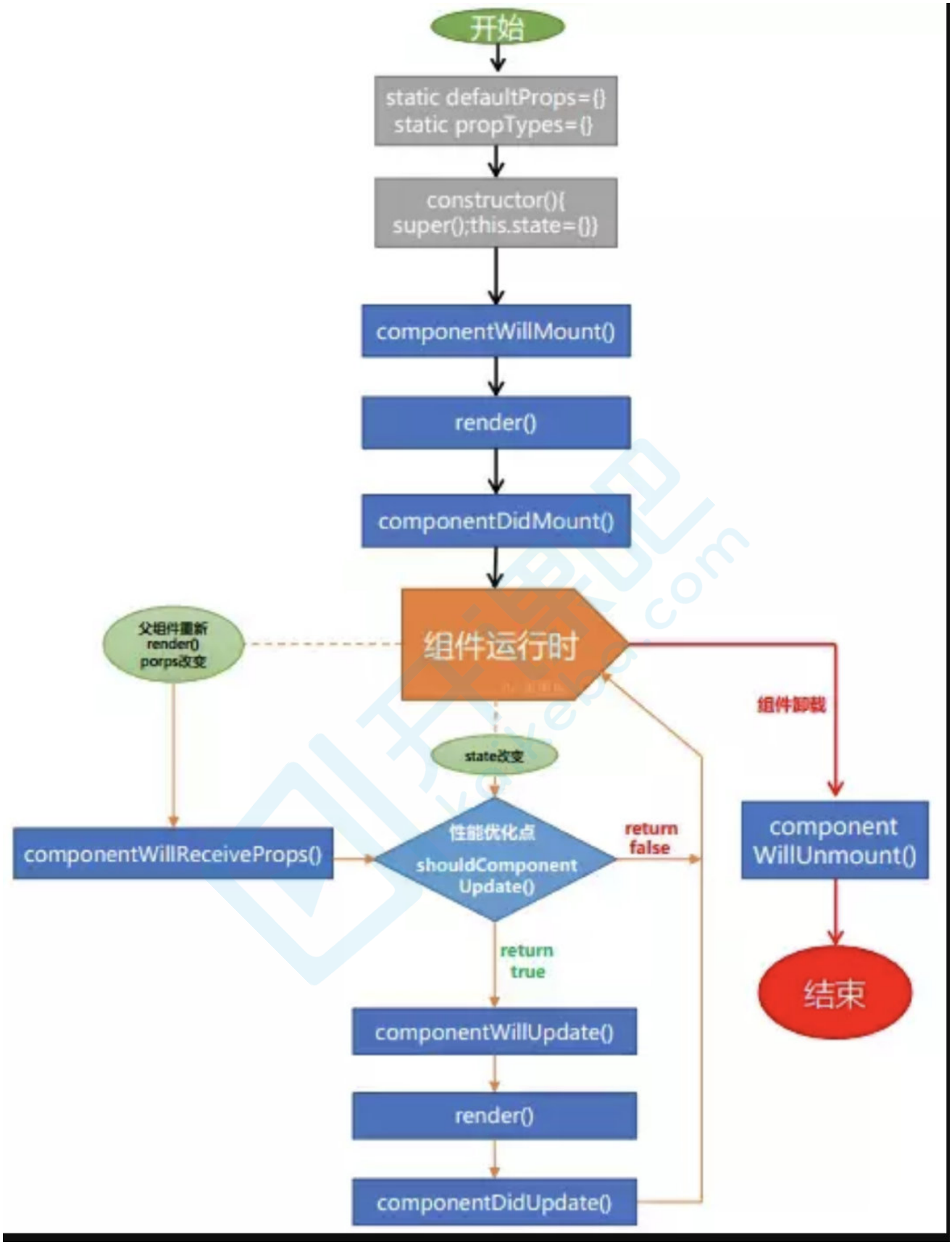
###

事件监听

组件通信

大名鼎鼎的虚拟DOM

生命周期


开始
static defaultProps={}
static propTypes={}
constructor(){
super();this.state={}}
componentWillMount()
render()
componentDidMount()
组件运行时
父组件重新
render()
porps改变
state改变
componentWillReceiveProps()
性能优化点
shouldComponent
Update()
return
false
return
true
componentWillUpdate()
render()
componentDidUpdate()
组件卸载
component
WillUnmount()
结束

###

回顾