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

预习资源

1. react
2. create-react-app

开发环境

1. vscode下载
2. node.js下载

起步

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                  根组件
    ├── App.test.js             测试
    ├── index.css               全局样式
    ├── index.js                入口
    ├── logo.svg
    └── serviceWorker.js        pwa支持
```

知识点

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, { Component } from "react";

export default class App extends Component{
  render(){
    return <div>
      <button>雷猴啊</button>
    </div>
  }
}

export default KaikebaCart
```

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

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
```

Props属性传递

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

...

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

条件渲染和循环

index.js

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

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

app.js

```
export default class App extends Component {
  constructor(props){
    super(props);
    this.state={
      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}
      ],
    }
  }
  render() {
    return (
      <div>
          {this.state.showTitle && <h2>{this.props.title}</h2>}
          <ul>
      {this.state.goods.map((good,i)=>{
        return <li key={good.id}>
          <span>{good.text}</span>
          <span>{good.price}</span>元
          <button onClick={()=>this.handleClick(i)}>添加购物车</button>
        </li>
      })}

      </ul>
      </div>
    )
  }
}
```

class VS 函数组件

如果一个组件只根据props渲染页面，没有内部的state，我们完全可以用函数组件的形式来实现(hooks的到来 会改变这个现状)

```
function Title({title}){
  return <h2>{title}</h2>
}
<Title title={this.props.title}></Title>
```

事件监听

React中使用onClick类似的写法来监听事件，注意this绑定问题 react里严格遵循单项数据流，没有数据双向绑定，所以输入框要设置value和onChange

```
handleChange(e){
    this.setState({
      name:e.target.value
    })
  }

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

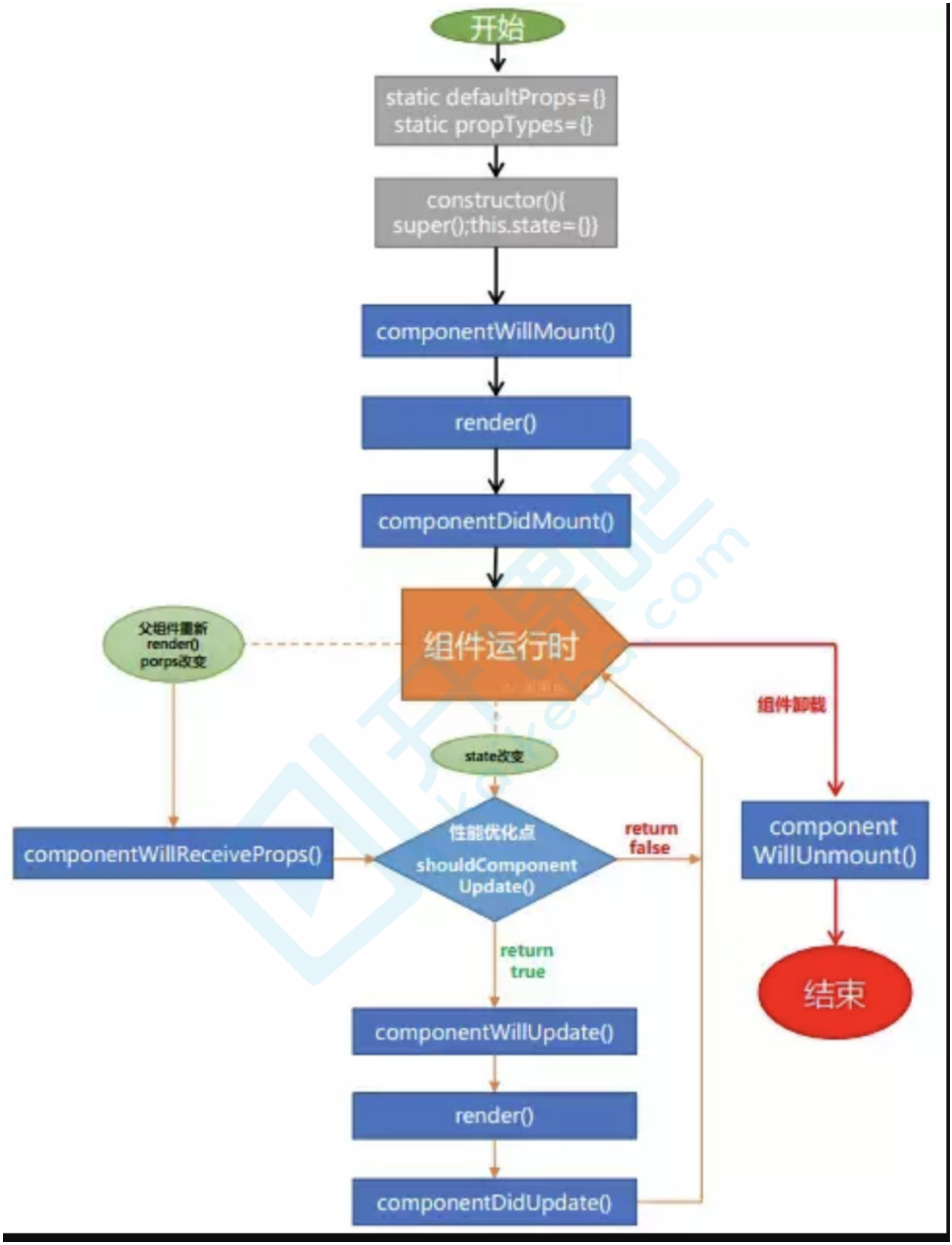
组件通信

```
function Welcome(props) {
  return (
    <div>
       hello,{props.name}
    </div>
  )
}

export default class App extends Component {
  render() {
    return (
      <div>
          <Welcome1 name="Jerry"></Welcome1>
      </div>
    )
  }
}
```

虚拟DOM

react生命周期



react后续展望

组件设计