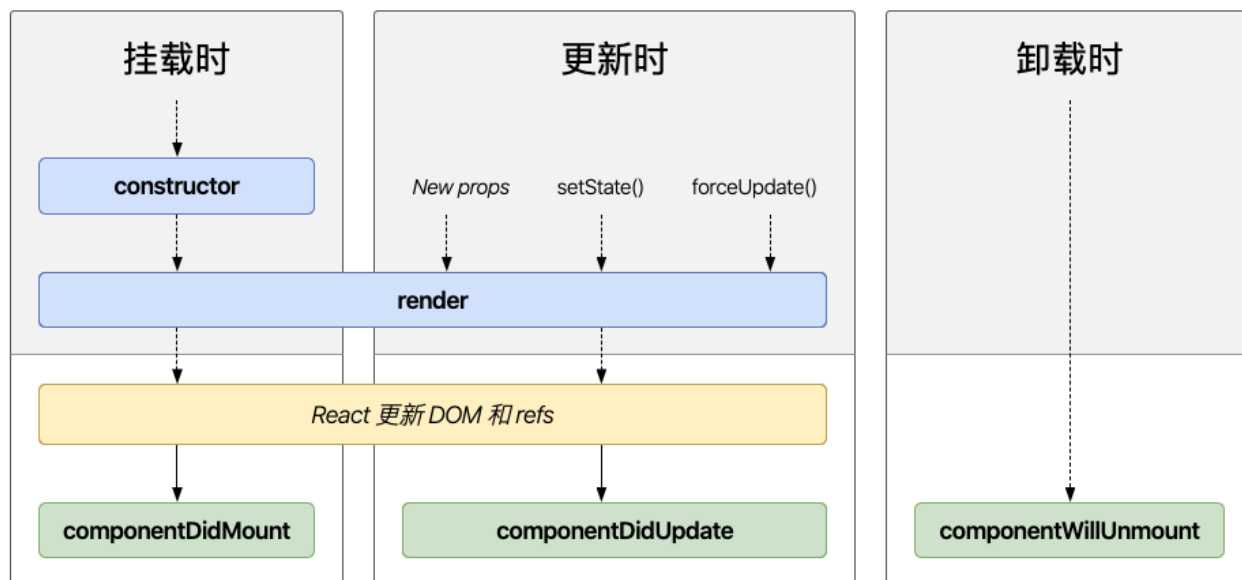


基本知识

向军大叔每晚八点在 [抖音 \(opens new window\)](#)和 [bilibili \(opens new window\)](#)直播

每个组件都包含“生命周期方法”，你可以重写这些方法，以便于在运行过程中特定的阶段执行这些方法。

以下是常用[生命周期图谱\(opens new window\)](#)



#接口开发

#AXIOS

为了讲解生命周期我们需要异步请求操作，下面来安装使用[Axios \(opens new window\)](#)库，[Axios \(opens new window\)](#)是一个基于 promise 的 HTTP 库,可以用在浏览器和 node.js 中使用。

安装扩展

```
1 npm i axios
2
```

#MOCK

为了测试我们使用 [淘宝 RAP2 \(opens new window\)](#)来做接口测试

新建接口配置



接口参数配置

响应内容

预览

	名称	必选	类型	生成规则 ②	初始值	简介
	code	 <input type="checkbox"/>	Number		200	状态码
	message	 <input type="checkbox"/>	String		用户列表获取成功	响应字符串
▼	data	 <input type="checkbox"/>	Array	20		列表数据
	id	 <input type="checkbox"/>	Number		@id	用户编号
	name	 <input type="checkbox"/>	String		@cname	用户名
	age	 <input type="checkbox"/>	Number	20-60		houdunren.com@向军大叔

现在我们可以使用 <http://rap2api.taobao.org/app/mock/243692/user> 来进行测试了

#接口服务

现在来添加接口请求服务功能，创建以下目录结构

```
1  src/services
2  └─ apis.js  #接口址
3  └─ index.js #服务入口，用于导出所有接口
4
```

apis.js 用于配置接口地址

```
1  export default {
2    user: {
3      list: "/user",
4      create: "user/create"
5    }
6  }
7
```

index.js 用于初始化 AXIOS，并定义接口动作

```
1  import axios from "axios"
2  import apis from "./apis"
3  const http = axios.create({
4    baseURL: "
http://rap2api.taobao.org/app/mock/243692
    baseURL: "
5  })
```

```

6 export default {
7   user: {
8     list: () => http.get(apis.user.list),
9     create: () => http.get(apis.user.crate)
10  }
11 }
12

```

#完善组件

现在我们在组件中使用 componentDidMount 生命周期方法来获取数据

- 删除原来的状态数据 this.users 清空
- componentDidMount 中发送异步请求，并更新组件状态

```

1 export default class List extends Component {
2   constructor() {
3     super()
4     this.state = {
5       users: []
6     }
7   }
8   componentDidMount() {
9     Api.user.list().then(response => {
10       this.setState({
11         users: response.data.data
12       })
13     })
14   }
15   ...
16 }
17

```

最终请求结果如图

用户名@年龄

后盾人@向军大叔

用户列表

编号	姓名	年龄	操作
230000197209268548	熊涛	28	删除
440000199202037315	戴勇	35	删除
310000199912175748	熊秀兰	25	删除
510000199404263730	江娟	39	删除
990000198312278163	贺军	45	删除
340000198111120713	程敏	56	删除

#实例操作

#shouldComponentUpdate

当 props 或 state 发生变化时，`shouldComponentUpdate()` 会在渲染执行之前被调用。

- 参数包括 nextProps 下次的 props 值与 nextState 下次的状态值
- 函数返回 false 将不更新组件，返回

false 并不会阻止子组件在 state 更改时重新渲染

修改 User/user.js 组件的 render 方法打印输出用户

- 会发现每次添加用户时都会渲染所有组件
- 因为我们是添加用户，老的用户数据没有必要渲染

```
1  render() {
2      console.log(this.props.name)
3      ...
4  }
5
```

像我们的组件老数据就不需要渲染了，使用 `shouldComponentUpdate` 生命周期方法返回 `FALSE` 可阻止更新

```
1  shouldComponentUpdate(nextProps, nextState) {  
2    return false  
3  }  
4
```

大家可以再控制台查看效果

#PureComponent

React.PureComponent 与 React.Component 很相似。两者的区别在于 React.Component 并未实现 shouldComponentUpdate(), 而 React.PureComponent 中以浅层对比 prop 和 state 的方式来实现了该函数。

- React.PureComponent 仅作对象的浅层比较
- React.PureComponent 将跳过所有子组件树的 props 更新, 所以子组件也都是“纯”的组件。

把之前定义的 shouldComponentUpdate 生命周期函数删除, 将类继承 PureComponent 即可实现相同的效果

```
1  import React, { Component, PureComponent } from "react"  
2  export default class User extends PureComponent {  
3    ...  
4  }
```