React.js 小书

<-- 返回首页

挂载阶段的组件生命周期（一）

- 作者：胡子大哈
- 原文链接： http://huziketang.com/books/react/lesson18

我们在讲解 JSX 的章节中提到，下面的代码：

```
ReactDOM.render(
 <Header />,
  document.getElementById('root')
)
```

会编译成：

```
ReactDOM.render(
  React.createElement(Header, null),
  document.getElementById('root')
)
```

其实我们把 `Header` 组件传给了 `React.createElement` 函数，又把函数返回结果传给了 `ReactDOM.render`。我们可以简单猜想一下它们会干什么事情：

```
// React.createElement 中实例化一个 Header
const header = new Header(props, children)
// React.createElement 中调用 header.render 方法渲染组件的内容
const headerJsxObject = header.render()

// ReactDOM 用渲染后的 JavaScript 对象来来构建真正的 DOM 元素
const headerDOM = createDOMFromObject(headerJsxObject)
// ReactDOM 把 DOM 元素塞到页面上
document.getElementById('root').appendChild(headerDOM)
```

上面过程其实很简单，看代码就能理解。

我们把 **React.js 将组件渲染，并且构造 DOM 元素然后塞入页面的过程称为组件的挂载**（这个定义请好好记住）。其实 React.js 内部对待每个组件都有这么一个过程，也就是初始化组件 -> 挂载到页面上的过程。所以你可以理解一个组件的方法调用是这么一个过程：

```
-> constructor()
-> render()
// 然后构造 DOM 元素插入页面
```

这当然是很好理解的。React.js 为了让我们能够更好的掌控组件的挂载过程，往上面插入了两个方法：

```
-> constructor()
-> componentWillMount()
-> render()
// 然后构造 DOM 元素插入页面
-> componentDidMount()
```

`componentWillMount` 和 `componentDidMount` 都是可以像 `render` 方法一样自定义在组件的内部。挂载的时候，React.js 会在组件的 `render` 之前调用 `componentWillMount`，在 DOM 元素塞入页面以后调用 `componentDidMount`。

我们给 `Header` 组件加上这两个方法，并且打一些 Log：

```
class Header extends Component {
  constructor () {
    super()
    console.log('construct')
  }

  componentWillMount () {
    console.log('component will mount')
  }

  componentDidMount () {
    console.log('component did mount')
  }

  render () {
    console.log('render')
    return (
      <div>
        <h1 className='title'>React 小书</h1>
      </div>
    )
  }
}
```

在控制台你可以看到依次输出：

```
construct
component will mount
render
component did mount
```

可以看到，React.js 确实按照我们上面所说的那样调用了定义的两个方法 `componentWillMount` 和 `componentDidMount`。

机灵的同学可以想到，一个组件可以插入页面，当然也可以从页面中删除。

```
-> constructor()
-> componentWillMount()
-> render()
// 然后构造 DOM 元素插入页面
-> componentDidMount()
// ...
// 从页面中删除
```

React.js 也控制了这个组件的删除过程。在组件删除之前 React.js 会调用组件定义的 `componentWillUnmount`：

```
-> constructor()
-> componentWillMount()
-> render()
// 然后构造 DOM 元素插入页面
-> componentDidMount()
// ...
// 即将从页面中删除
-> componentWillUnmount()
// 从页面中删除
```

看看什么情况下会把组件从页面中删除，继续使用上面例子的代码，我们再定义一个 `Index` 组件：

```
class Index extends Component {
  constructor() {
    super()
    this.state = {
      isShowHeader: true
    }
  }

  handleShowOrHide () {
    this.setState({
      isShowHeader: !this.state.isShowHeader
    })
  }

  render () {
    return (
      <div>
        {this.state.isShowHeader ? <Header /> : null}
        <button onClick={this.handleShowOrHide.bind(this)}>
          显示或者隐藏标题
        </button>
```

```
      </div>
    )
  }
}

ReactDOM.render(
  <Index />,
  document.getElementById('root')
)
```

`Index` 组件使用了 `Header` 组件，并且有一个按钮，可以控制 `Header` 的显示或者隐藏。下面这行代码：

```
...a
{this.state.isShowHeader ? <Header /> : null}
...
```

相当于 `state.isShowHeader` 为 `true` 的时候把 `Header` 插入页面，`false` 的时候把 `Header` 从页面上删除。这时候我们给 `Header` 添加 `componentWillUnmount` 方法：

```
...
  componentWillUnmount() {
    console.log('component will unmount')
  }
...
```

这时候点击页面上的按钮，你会看到页面的标题隐藏了，并且控制台打印出来下图的最后一行，说明 `componentWillUnmount` 确实被 React.js 所调用了：

```
construct
component will mount
render
component did mount
component will unmount
```

你可以多次点击按钮，随着按钮的显示和隐藏，上面的内容会按顺序重复地打印出来，可以体会一下这几个方法的调用过程和顺序。

总结

React.js 将组件渲染，并且构造 DOM 元素然后塞入页面的过程称为组件的挂载。这一节我们学习了 React.js 控制组件在页面上挂载和删除过程里面几个方法：

- `componentWillMount`：组件挂载开始之前，也就是在组件调用 `render` 方法之前调用。
- `componentDidMount`：组件挂载完成以后，也就是 DOM 元素已经插入页面后调用。
- `componentWillUnmount`：组件对应的 DOM 元素从页面中删除之前调用。

但这一节并没有讲这几个方法到底在实际项目当中有什么作用，下一节我们通过例子来讲解一下这几个方法的用途。

> 因为第三方评论工具有问题，对本章节有任何疑问的朋友可以移步到 React.js 小书的论坛 发帖，我会回答大家的疑问。

下一节：挂载阶段的组件生命周期（二）

上一节：前端应用状态管理 — 状态提升