# 第七章 mixin

第6章提到过，mixin允许我们定义可以在多个组件中共用的方法。让我们更详细地探究这一点。

## 什么是mixin

在React的主页上有这样一段定时器组件的例子：

Var Timer=React.createClass({

getInitialState:function(){

return {secondsElapsed:0};

},

tick:function(){

This.setState({secondsElapsed:this.state.secondsElapsed+1});

},

componentDidMount:function(){

this.interval=setInterval(this.tick,1000);

},

Render:function(){

Return(

<div>Seconds Elapsed:{this.state.secondsElapsed}</div>

);

}

});

这些代码看起来不错，不过我们有多个组件都要使用计时器，执行的都是上面这段代码。这个时候就轮到mixin大显神威了。我们想最终实现一个像下面这样的定时器组件：

var Timer=React.createClass({

mixins:[IntervalMixin(1000)],

getInitialState:function(){

return {secondsElapsed:0};

},

onTick:function(){

this.setState({secondsElapsed:this.state.secondsElapsed+1});

},

Render:function(){

Return(

<div>Seconds Elapsed:{this.state.secondsElapsed}</div>

);  
}

});

Mixin相当简单，它们就是混合进组件类中的对象而已。React在这方面实现得更加深入，它能防止静默函数覆盖，同时还支持多个mixin混合。这些功能在别的系统中可能引起冲突。举个例子：

React.createClass({

Mixins:[{

getInitialState:function(){return {a:1}}

}],

getInitialState:function(){

return {b:2}

}

});

我们在mixin和组件类中同时定义了getInitialState方法，得到的初始state是{a:1,b:2}。如果mixin中的方法和组件类中的方法返回的对象中存在重复的键，React会抛出一个错误来示警这个问题。

以component开头的生命周期方法，如componentDidMount，会按照mixin数组中定义的顺序被调用，并最终调用组件类中定义的componentDidMount，如果它存在的话。

回到我们之前的例子，我们需要实现一个IntervalMixin。有时候单独一个对象足以完成mixin工作，然而其他情况下我们需要用一个函数来返回这个对象。在这个例子中，我们希望能够指定时间间隔。

Var IntervalMixin=function(interval){

Return{

componentDidMount:function(){

this.\_\_interval=setInterval(this.onTick,interval);

},

componentWillUnmount:function(){

clearInterval(this.\_\_interval);

}

};

};

这样实现很不错，但是也有一定的局限性。我们不能拥有多个定时器，用户也不能选择处理定时器的函数，而且我们也不能手动终止这个定时器，除非使用内部属性\_\_interval。要解决这些问题，我们的mixin需要一个公共API。

下面这个例子简单地展示了自2014年1月1日到目前为止的总秒数。这段代码相对长了一些，但却更加灵活、强大。

var IntervalMixin={

setInterval:function(callback,interval){

var token=setInterval(call,interval);

this.\_\_intervals.push(token);

return token;

},

componentDidMount:function(){

this.\_\_intervals=[];

},

componentWillUnmount:function(){

this.\_\_intervals.map(clearInterval);

}

};

Var Since2014=React.createClass({

Mixins:[IntervalMixin],

componentDidMount:function(){

this.setInterval(this.forceUpdate.bind(this),1000);

},

Render:function(){

Var form=Number(new Date(2014,0,1));

Var to=Date.now();

Return(

<div>{Math.round((to-from)/1000)}</div>

);

}

});

关于mixin的例子还有很多，我们无法在这里全部展示，然而下面几种用法值得参考：

A.一个监听事件并修改state的mixin (如flux store mixin)。

B. 一个上传mixin，它负责处理XHR上传请求，同时将状态以及上传的进度同步到state。

C. 渲染层mixin，简化在</body>之前渲染子元素的过程。