ReactJS生命周期详解

# 实例化

首次实例化

* getDefaultProps
* getInitialState
* componentWillMount
* render
* componentDidMount

实例化完成后的更新

* getInitialState
* componentWillMount
* render
* componentDidMount

# 存在期

组件已存在时的状态改变

* componentWillReceiveProps
* shouldComponentUpdate
* componentWillUpdate
* render
* componentDidUpdate

# 销毁&清理期

* componentWillUnmount

# 说明

生命周期共提供了10个不同的API。

#### 1.getDefaultProps

作用于组件类，只调用一次，返回对象用于设置默认的props，对于引用值，会在实例中共享。

#### 2.getInitialState

作用于组件的实例，在实例创建时调用一次，用于初始化每个实例的state，此时可以访问this.props。

#### 3.componentWillMount

在完成首次渲染之前调用，此时仍可以修改组件的state。

#### 4.render

必选的方法，创建虚拟DOM，该方法具有特殊的规则：

* 只能通过this.props和this.state访问数据
* 可以返回null、false或任何React组件
* 只能出现一个顶级组件（不能返回数组）
* 不能改变组件的状态
* 不能修改DOM的输出

#### 5.componentDidMount

真实的DOM被渲染出来后调用，在该方法中可通过this.getDOMNode()访问到真实的DOM元素。此时已可以使用其他类库来操作这个DOM。

在服务端中，该方法不会被调用。

#### 6.componentWillReceiveProps

组件接收到新的props时调用，并将其作为参数nextProps使用，此时可以更改组件props及state。

componentWillReceiveProps: function(nextProps) {

if (nextProps.bool) {

this.setState({

bool: true

});

}

}

#### 7.shouldComponentUpdate

组件是否应当渲染新的props或state，返回false表示跳过后续的生命周期方法，通常不需要使用以避免出现bug。在出现应用的瓶颈时，可通过该方法进行适当的优化。

在首次渲染期间或者调用了forceUpdate方法后，该方法不会被调用

#### 8.componentWillUpdate

接收到新的props或者state后，进行渲染之前调用，此时不允许更新props或state。

#### 9.componentDidUpdate

完成渲染新的props或者state后调用，此时可以访问到新的DOM元素。

#### 10.componentWillUnmount

组件被移除之前被调用，可以用于做一些清理工作，在componentDidMount方法中添加的所有任务都需要在该方法中撤销，比如创建的定时器或添加的事件监听器。