Vue生命周期

入坑前端一月有余，项目用的vue框架和ES6语法，项目这两天不是很忙，开始填坑，现在知道怎么和后端数据做交互，但对于vue生命周期还不是很清楚，看完vue官网介绍和查阅一些资料之后，总结如下：

1.总结图

2.生命周期钩子

3.Demo实例演示

4.小结：

**1.总结图**

首先给出一个生命周期的总结图：



Vue实例从创建到销毁的过程，就是生命周期。

Vue的生命周期包括：开始创建、初始化数据、编译模板、挂载Dom、渲染→更新→渲染、卸载等一系列过程。

在Vue的整个生命周期中，提供了一系列的事件，可以注册JavaScript方法，达到控制整个过程的目的，在这些JavaScript方法中的this直接指向的是vue的实例。

**2.生命周期钩子**

在Vue的整个生命周期中，实例可以调用一些生命周期钩子，这提供了执行自定义逻辑的机会。Vue提供的生命周期钩子如下：

①　beforeCreate

在实例初始化之后，数据观测(data observer，开始监控Data对象数据变化)和初始化事件(init event，Vue内部初始化事件)之前被调用。

②　created

在实例已经创建完成之后被调用。实例已完成以下的配置：数据观测(data observer)，属性和方法的运算，event事件回调。挂载阶段尚未开始，$el 属性不可见。

③　beforeMount

在挂载开始之前被调用。相关的render函数首次被调用。实例已完成以下的配置：编译模板，把data里面的数据和模板生成html。注意此时还没有挂载html到页面上。

④　mounted

在el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用。实例已完成以下的配置：用上面编译好的html内容替换el属性指向的DOM对象。此时模板中的html渲染到了html页面中，此时一般可以做一些Ajax操作。注意mounted只会执行一次。

注意 mounted 不会承诺所有的子组件也都一起被挂载。如果你希望等到整个视图都渲染完毕，可以用 vm.$nextTick 替换掉 mounted：该钩子在服务器端渲染期间不被调用。



⑤　beforeUpdate

在数据更新之前调用，发生在虚拟DOM重新渲染和打补丁之前。可以在该钩子中进一步地更改状态，不会触发附加的重渲染过程。

⑥　updated

在由于数据更改导致的虚拟DOM重新渲染和打补丁之后调用。调用时，组件DOM已经更新，所以可以执行依赖于DOM的操作。然而在大多数情况下，应该避免在此期间更改状态，因为这可能会导致更新无限循环。该钩子在服务器端渲染期间不被调用。

注意 updated 不会承诺所有的子组件也都一起被重绘。如果你希望等到整个视图都重绘完毕，可以用 vm.$nextTick 替换掉 updated：• 该钩子在服务器端渲染期间不被调用。



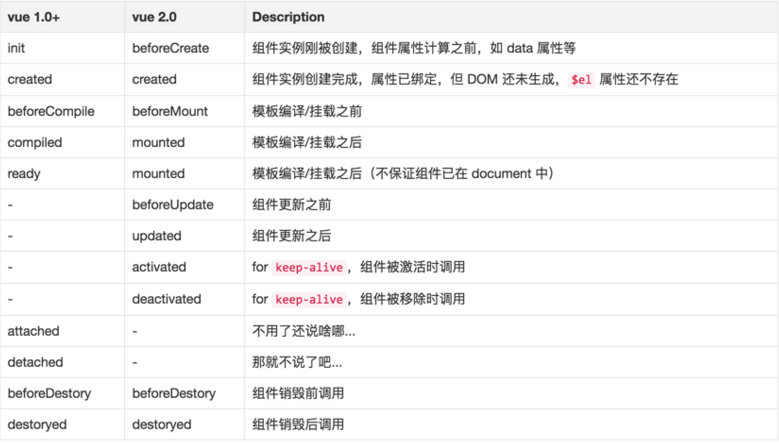
⑦　beforeDestroy

在实例销毁之前调用。实例仍然完全可用。

⑧　destroyed

在实例销毁之后调用。调用后，所有的事件监听器会被移除，所有的子实例也会被销毁。该钩子在服务器端渲染期间不被调用。

生命周期钩子是vue升级中改变最多的一部分了，对照1.0的文档和release note，表总结如下：



**3.Demo实例演示**

我们下面结合一个demo实例看下钩子函数的执行。

Create和mounted相关：

通过chrome里的console，我们可以看到：

1、beforeCreate 此时$el、data 的值都为undefined

2、创建之后，此时可以拿到data的值，但是$el依旧为undefined

3、mount之前，$el的值为“虚拟”的元素节点。

4、mount之后，mounted之前，“虚拟”的dom节点被真实的dom节点替换，并将其插入到dom树中，于是在触发mounted时，可以获取到$el为真实的dom元素()

　　myVue.$el===document.getElementById("app-8") // true

update相关

在console执行：app.message=’qslshan’;

可以看到data里的值被修改后，将会触发update的操作。

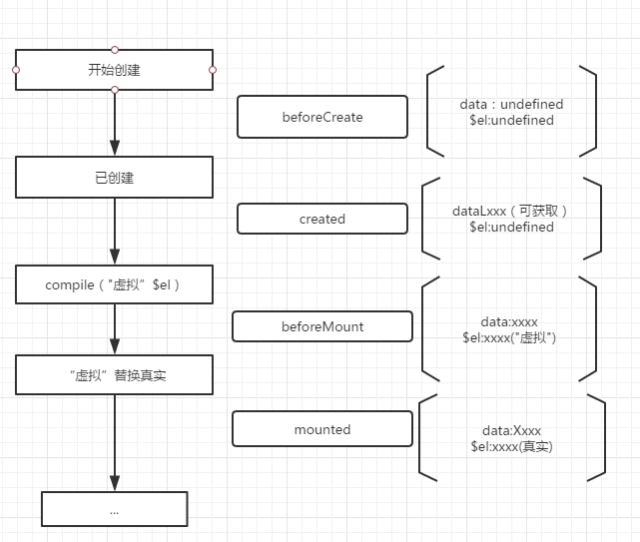


destroy相关

我们在console里执行下命令app.$destroy();

对 vue实例进行销毁。销毁完成后，我们再重新改变message的值，vue不再对此动作进行响应了。但是原先生成的dom元素还存在，可以这么理解，执行了destroy操作，后续就不再受vue控制了。

所以将官方文档的图进行简化后可得到下图：

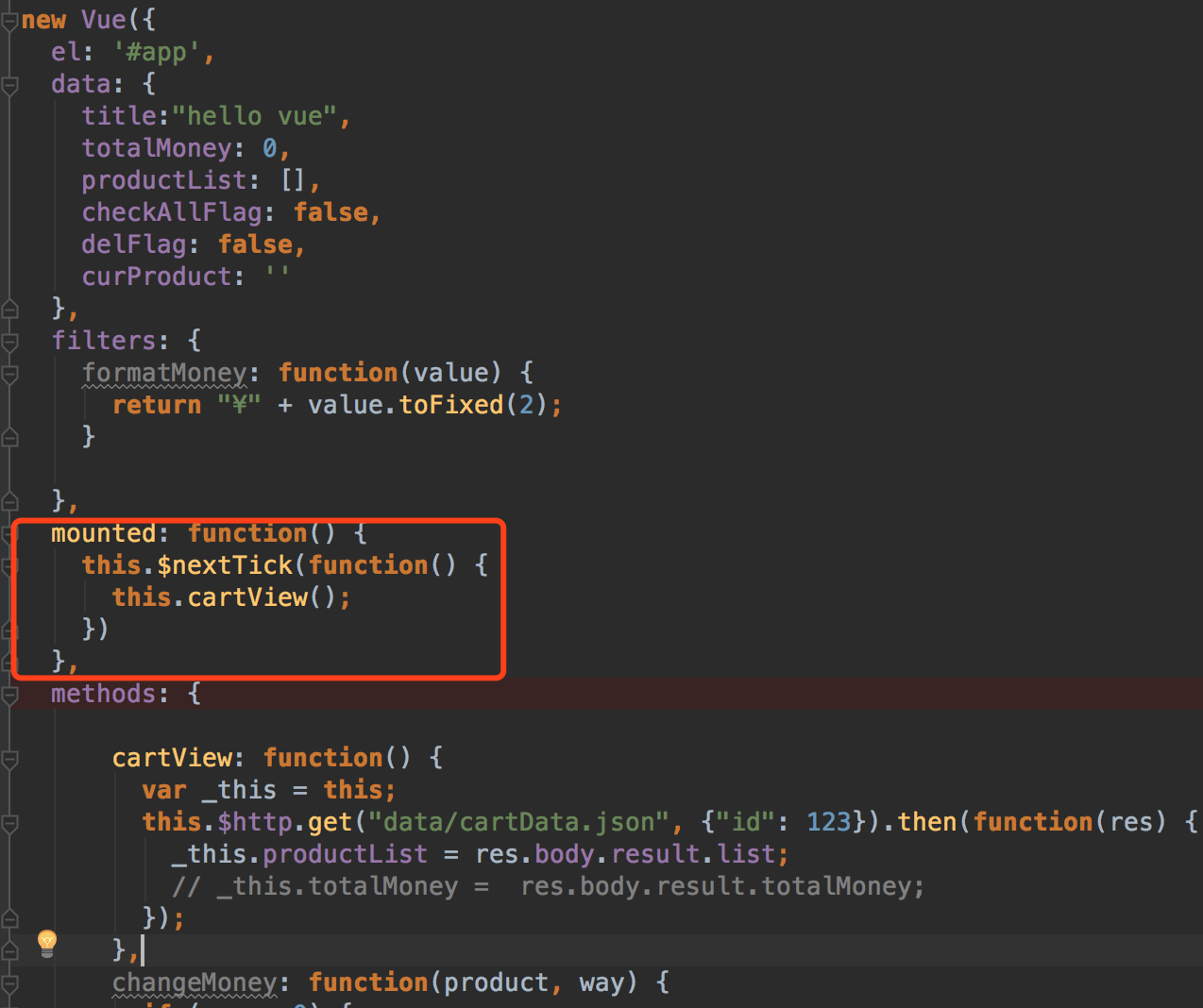


**4.小结：**

我们该怎么用这些钩子函数呢？哈哈，this is a question：

在项目中发现，在各种的钩子中，created使用最频繁，因为在这个阶段已经可以对数据进行异部的读取了（从后台获取数据），也可以读取到相关的属性，因为数据会有延迟，一般不会在mounted这个阶段（当然这个阶段也可以实现方法）才和后台的api进行交互。created钩子每次加载完成后都可以重复执行；而mounted钩子只在页面第一次加载后才调用出来，只要el被加载过，之后的重复加载该页面就不会调用该钩子了。

做的一个实例代码截图如下：因为mounted 不会承诺所有的子组件也都一起被挂载。这里用 vm.$nextTick 替换掉 mounted：



其他几个hook函数的应用场景如下：

beforecreate : 举个栗子：可以在这加个loading事件

created ：在这结束loading，还做一些初始化，实现函数自执行

mounted ： 在这发起后端请求，拿回数据，配合路由钩子做一些事情

beforeDestory： 你确认删除XX吗？ destoryed ：当前组件已被删除，清空相关内容

欢迎一起交流学习！