Vue实例中的生命周期钩子

1. Vue周期前沿

vue实例有有一个完整的生命周期，也就是从开始创建、初始化数据、编译模版、挂载DOM----->渲染、更新---->渲染、卸载等一系列过程，我们称之为vue生命周期。

二、生命周期钩子使用：

A、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

B、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

C、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

D、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

E、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。

F、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。

G、beforeDestroy：在实例销毁之前调用。实例仍然完全可用；

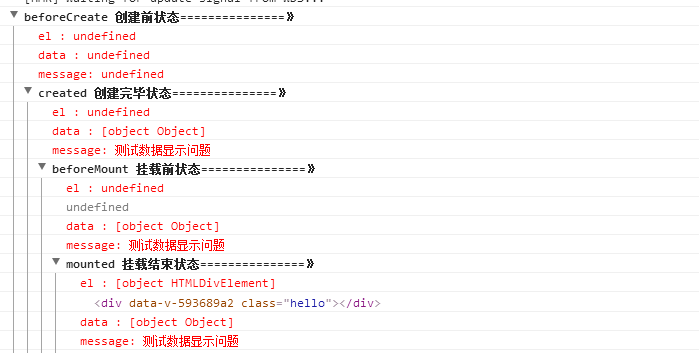
H、destroyed：在实例销毁之后调用。调用后，所有的事件监听器会被移除，所有的子实例也会被销毁。该钩子在服务器端渲染期间不被调用

下面代码可以直接执行查看结果：

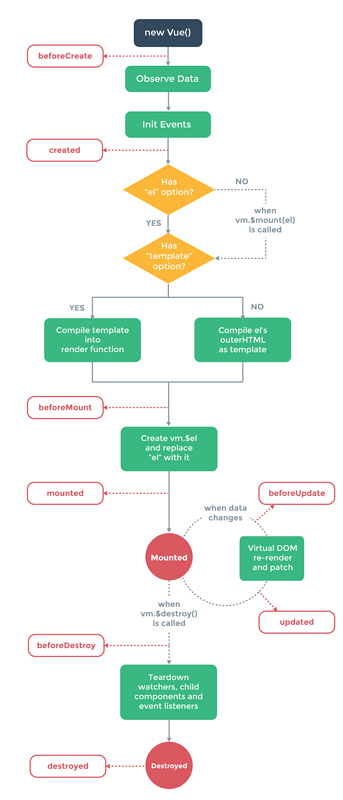




接下来通过vue的渲染出来的结果是：



三、生命周期简介图章



四、vue实例初始化的选项配置对象介绍：

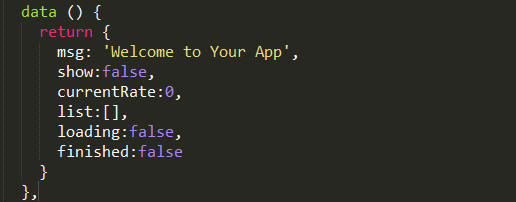
4.1、vue实例的data对象：自动监视data里面的数据变化，自动更新数据到html标签上去，本质原理就是：vue中自动将data里面的数据进行递归换成getter和setter，然后就可以自动更新HTML标签了。

data对象的类型：

类型是Object或者Function；

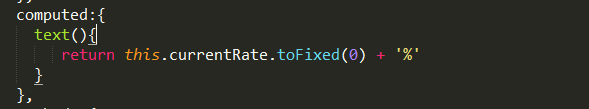
如果是组件对象，data必须是Function类型

例如：



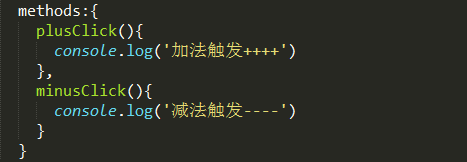
4.2、vue实例的computed属性：vue中的计算属性(computed)的属性会自动混入vue的实例中，所有getter和setter的this上下文自动地绑定为vue实例，而且计算属性中定义的函数里面可以直接使用指向了vue实例的this。

{ 键：函数} { [key: string]: Function | { get: Function, set: Function } }



4.3、vue实例的methods对象：methods将被混入到vue实例中，可以通过实例访问methods里面的这些方法，或者在指令表达式中使用，方法中的this自动绑定为vue实例。

类型: { [key: string]: Function }

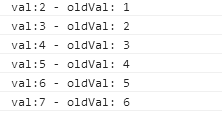


4.4、vue实例中watch对象：键是需要观察的表达式，值是回调函数，值也可以是方法名，或者包含选项的对象，vue实例将会在实例化时调用$watch()，遍历watch对象的每一个属性

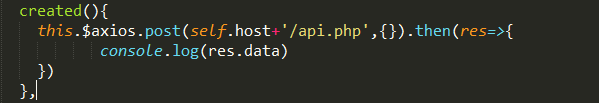
{ [key: string]: string | Function | Object }



运行之后的结果打印：

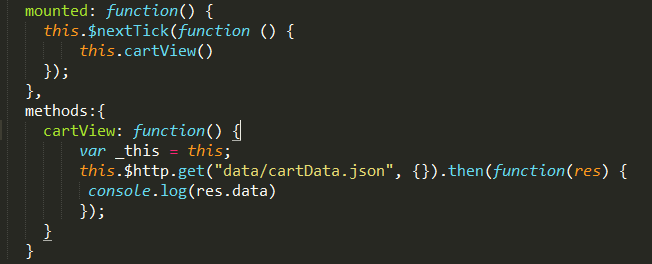


4.5、vue实例的created方法：vue实例被生成后调用这个函数，一般可以在created函数中调用ajax获取页面初始化所需的数据；



4.6、vue实例中的mounted方法：在这发起后端请求，拿回数据，配合路由钩子进行一些DOM节点操作；

类型： Function



综上几点vue的生命钩子，说明vue的实例封装的还是挺有艺术的，很符合开发者的思维规范，生命周期清晰，使用方便。在vue的整个生命周期中，它提供了一系列的时间，可以让我们注册JS方法，可以让我们打到控制整个过程的目的。从开始创建、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。这就是vue的生命周期；