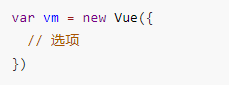
# Vue实例

## 创建Vue实例

每个 Vue 应用都是通过用 Vue 函数创建一个新的 Vue 实例开始的：



在构造函数中传入一个对象，并且在对象中声明各种Vue需要的数据和方法，包括：

- el

- data

- methods

等等

接下来我们一 一介绍。

## 模板el或元素

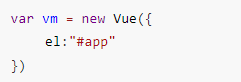
每个Vue实例都需要关联一段Html模板，Vue会基于此模板进行视图渲染。

我们可以通过el属性来指定。

例如一段html模板：



然后创建Vue实例，关联这个div

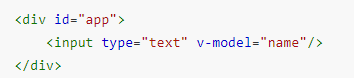


这样，Vue就可以基于id为app的div元素作为模板进行渲染了。在这个div范围以外的部分是无法使用vue特性的。

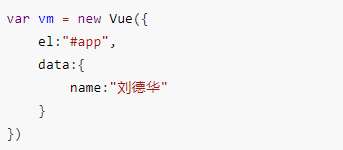
## 数据

当Vue实例被创建时，它会尝试获取在data中定义的所有属性，用于视图的渲染，并且监视data中的属性变化，当data发生改变，所有相关的视图都将重新渲染，这就是“响应式“系统。

html：



js:



- name的变化会影响到input的值

- input中输入的值，也会导致vm中的name发生改变

## 方法

Vue实例中除了可以定义data属性，也可以定义方法，并且在Vue实例的作用范围内使用。

html:



js:



# 生命周期钩子

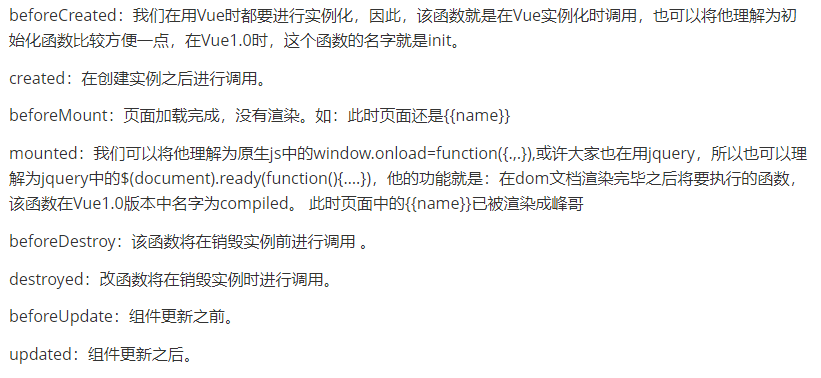
## 生命周期

每个 Vue 实例在被创建时都要经过一系列的初始化过程 ：创建实例，装载模板，渲染模板等等。Vue为生命周期中的每个状态都设置了钩子函数（监听函数）。每当Vue实例处于不同的生命周期时，对应的函数就会被触发调用。

生命周期：

图，自己看周期

## 钩子函数



例如：created代表在vue实例创建后；

我们可以在Vue中定义一个created函数，代表这个时期的钩子函数：

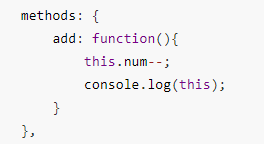


结果：

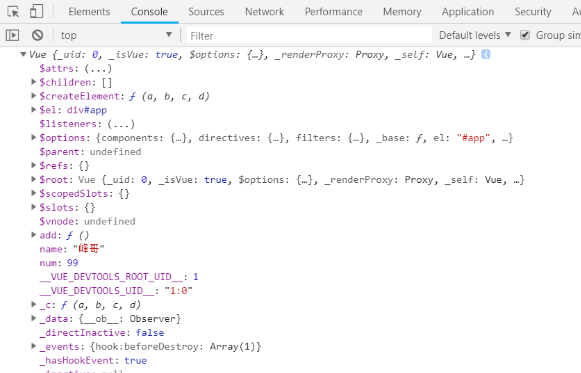


## this

我们可以看下在vue内部的this变量是谁，我们在created的时候，打印this



控制台的输出：



## this代理vue实例出现的地方

Let self = this;

Data: 校验函数

computed

Watch

method

没有：

Filter