

Vue.js

——渐进式JavaScript 框架

# 什么是Vue

Vue (读音 /vjuː/，类似于 view) 是一套用于构建用户界面的前端框架（[https://cn.vuejs.org](https://cn.vuejs.org/)）。它易于上手功能强大，并且其**渐进式**的设计理念更增加了它使用的灵活性：可以使用各个部件构建成完整的框架，也可以只当作js库来使用其核心功能，**声明式渲染**。

**渐进式框架**

<https://www.zhihu.com/question/51907207>

各个部件之间相互独立，我们可以只是简单的使用其核心功能，而不需要为了使用框架而引入其他的部件，写无用但是框架规定必需要用的代码。

**声明式渲染**

Vue.js 的核心是一个允许采用简洁的语法来声明式地将数据渲染进 DOM 的系统，此时数据和 DOM 已经被建立了关联，所有东西都是**响应式的**，即改变数据后DOM元素会自动改变，在输入控件中修改值，数据也会相应改变。

# 创建第一个Vue实例

首先要引入Vue库，Vue只依赖于js，不依赖于jquery。

<div id="testVue">

**{{**message**}}**

**{{**withSuffix**}}**

</div>

<script src="https://cdn.jsdelivr.net/npm/vue@2.6.10/dist/vue.js"></script>

<script>

var vue = new Vue({

el: "#testVue",

data: {

message: "Vue实例",

isShow: true

},

computed: {

withSuffix: function(){

return this.message + "--firstVue";

}

},

methods: {

showClick: function () {

vue.isShow = !vue.isShow;

}

},

created: function () {

this.message = "My First Vue";

}

})

</script>

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。当一个 Vue 实例被创建时，它将 data 对象中的所有的属性加入到 Vue 的**响应式系统**中。当这些属性的值发生改变时，视图将会产生“响应”，即匹配更新为新的值。值得注意的是只有当实例被创建时就已经存在于 data 中的属性才是**响应式**的。

每个 Vue 实例在被创建时都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做**生命周期钩子**的函数，这给了用户在不同阶段添加自己的代码的机会。比如 [created](https://cn.vuejs.org/v2/api/#created)。下图展示了实例的生命周期。



# Vue语法

Vue的基本语法包括以下几部分

1. 模板语法：包括插值和指令
2. 计算属性
3. 侦听器
4. Class与Style绑定
5. 条件渲染
6. 列表渲染
7. 事件处理
8. 表单输入绑定
9. 组件

其中模板语法是最基础的部分，它说明了怎么将响应式的数据绑定到html(View)中；计算属性可以看作对于数据属性的封装，并可以在数据属性改变时进行响应；侦听器用于监听某个数据属性，在其发生改变时执行方法；4~8是对于指令的具体使用；组件是对于一整块html代码的封装。下面我们具体查看每一部分的内容。

## 模板语法

Vue.js 使用了基于 HTML 的模板语法，允许开发者声明式地将 DOM 绑定至底层 Vue 实例的数据。所有 Vue.js 的模板都是合法的 HTML ，所以能被遵循规范的浏览器和 HTML 解析器解析。

在底层的实现上，Vue 将模板编译成虚拟 DOM 渲染函数。结合响应系统，Vue 能够智能地计算出最少需要重新渲染多少组件，并把 DOM 操作次数减到最少。

**插值**

插值是指使用**“Mustache”语法 (双大括号)** 直接将数据属性插入到html中，Mustache 标签中可以放入数据属性也可以放入js代码，Mustache 标签将会被替代为数据属性的值或js代码的结果。此时绑定的数据属性发生改变，插值处的内容会更新。

通过使用[**v-once 指令**](https://cn.vuejs.org/v2/api/#v-once)，你也能执行一次性地插值，当数据改变时，插值处的内容不会更新。但请留心这会影响到该节点上的其它数据绑定。

例子：**{{**message**}} {{**message + ‘我是后缀’**}} {{**1+2+3**}} <span v-once>{{**message**}}</span>**

**指令**

刚刚使用到的v-once是一个指令，指令 (Directives) 是带有 v- 前缀的特殊特性。指令特性的值预期是**单个 JavaScript 表达式** (v-for是例外情况，稍后我们再讨论)。指令的职责是，当表达式的值改变时，将其产生的连带影响，响应式地作用于 DOM。

例子：<p v-if="seen">现在你看到我了</p>

这里，v-if 指令将根据表达式 seen 的值的真假来插入/移除 <p> 元素。

一些指令能够接收一个“参数”，在指令名称之后以冒号表示。例如，v-bind 指令可以用于响应式地更新 HTML 特性：

例子：<a v-bind:href="url">...</a>

在这里 href 是参数，告知 v-bind 指令将该元素的 href 特性与表达式 url 的值绑定。

另一个例子是 v-on 指令，它用于监听 DOM 事件：

例子：<a v-on:click="doSomething">...</a>

在这里参数是监听的事件名。我们也会更详细地讨论事件处理。

动态参数、修饰符、缩写，参考官网。

# 计算属性