# Vue

## 什么是mvvm

MVVM最早由微软提出来，它借鉴了桌面应用程序的MVC思想，在前端页面中，把Model用纯JavaScript对象表示，View负责显示，两者做到了最大限度的分离 把Model和View关联起来的就是ViewModel。 ViewModel负责把Model的数据同步到View显示出来，还负责把View的修改同步回Model View 和 Model 之间的同步工作完全是自动的，无需人为干涉（由viewModel完成，在这里指VUE） 因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理

#### ViewModel如何编写？

需要用JavaScript编写一个通用的ViewModel，这样，就可以复用整个MVVM模型了

一个MVVM框架和jQuery操作DOM相比有什么区别？ 我们先看用jQuery实现的修改两个DOM节点的例子：

<!-- HTML -->  
<p>Hello, <span id="name">Bart</span>!</p>  
<p>You are <span id="age">12</span>.</p>  
  
Hello, Bart!  
  
You are 12.

用jQuery修改name和age节点的内容：

var name = 'Homer';  
var age = 51;  
  
$('#name').text(name);  
$('#age').text(age);

如果我们使用MVVM框架来实现同样的功能，我们首先并不关心DOM的结构，而是关心数据如何存储。最简单的数据存储方式是使用JavaScript对象：

var person = {  
 name: 'Bart',  
 age: 12  
}

我们把变量person看作Model，把HTML某些DOM节点看作View，并假定它们之间被关联起来了。

要把显示的name从Bart改为Homer，把显示的age从12改为51，我们并不操作DOM，而是直接修改JavaScript对象：

person.name = 'Homer';  
person.age = 51;

执行上面的代码，我们惊讶地发现，改变JavaScript对象的状态，会导致DOM结构作出对应的变化！这让我们的关注点从如何操作DOM变成了如何更新JavaScript对象的状态，而操作JavaScript对象比DOM简单多了！

这就是MVVM的设计思想：关注Model的变化，让MVVM框架去自动更新DOM的状态，从而把开发者从操作DOM的繁琐步骤中解脱出来！ 下图可以很好的解释view viewModel model之间的关系 ![mvvm](data:text/html; charset=utf-8;base64,)

## Vue的优点是什么

mvvm的优点即是vue的优点，在这里再总结一下： 数据和视频之间的同步工作完全是自动的，无需人为干涉，所以开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题， 复杂的数据状态维护完全由 MVVM 来统一管理，节省了很多精力。

## 对于生命周期的理解

创建一个Vue实例，是一个漫长的过程，要经历初始化，数据合并，模板解析，数据渲染等等一系列过程。 所以，为了能实现在这个过程里面插入自己想要提前做的事情，就有了生命周期钩子函数。

#### 举个例子：

一辆公交车，从出发点A站到终点站B，中间有很多站点，公交车每到一个站点，就得停下来，  
等待客人上车，然后再驶往下一个站点，一直到终点站为止。  
A和B之间的站点，就像是这个路程的生命周期。每一个站点都是一个不同的生命周期（站点名不同），  
只要到了站点，就得执行该站点对应的生命周期函数，  
只不过每个站点的生命周期函数都是一样的（等待客人上车）。

Vue中的生命周期也是一样，对应了Vue实例从创建到结束之间的每一个过程。 例如，Vue的beforeCreate周期，指的就是Vue在实例初始化之后，数据观测 (data observer) 和 event/watcher 事件配置之前被调用。

至于Vue具体的生命周期函数有哪些，请看官网[API文档](https://cn.vuejs.org/v2/api/#%E9%80%89%E9%A1%B9-%E7%94%9F%E5%91%BD%E5%91%A8%E6%9C%9F%E9%92%A9%E5%AD%90)

## 组件传值

* 父组件通过prop向子组件传值
* 子组件通过事件向父组件传值
* 子组件与子组件之间不能直接传值，需要通过父组件来做间接传值，在这种情况下推荐使用vuex

具体例子请看[官方文档](https://cn.vuejs.org/v2/guide/components.html#%E9%80%9A%E8%BF%87-Prop-%E5%90%91%E5%AD%90%E7%BB%84%E4%BB%B6%E4%BC%A0%E9%80%92%E6%95%B0%E6%8D%AE)

## vue数据绑定原理

Vue的数据双向绑定都是依据Object.defineProperty()这一方法来做的 Object.defineProperty到底有什么作用呢？ [MDN](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty)

Object.defineProperty(obj, prop, descriptor)  
  
obj  
要在其上定义属性的对象。  
  
prop  
要定义或修改的属性的名称。  
  
descriptor  
将被定义或修改的属性描述符。

简单来说 这个方法可以定义一个对象某个属性的描述符

我们需要用到的就是描述符当中的getter和setter

const obj = {a:1}  
obj.a // 1  
  
obj.a = 2

像上面代码中的两个操作 读取和赋值 就是在访问obj.a的getter和setter 当我们输入obj.a时 就是在访问obj对象a属性的getter 当输入obj.a = 2 时就是在访问obj对象a属性的setter

Object.defineProperty(obj, 'a', {  
 get : function(){  
 return val  
 },  
 set : function(newValue){  
 val = newValue  
 },  
 enumerable : true,  
 configurable : true  
})

getter和setter都是一个函数 我们还可以这样做 例如

get: function() {  
 // 每次访问obj.a时都会执行这段代码  
 console.log('hello, 你在读取a的值')  
 return val  
}  
set: function(newValue) {  
 val = newValue  
 // 每次给obj.a赋值时都会执行这段代码  
 console.log('你设置了a的值')  
}

Vue的双向数据绑定就是根据上面的原理来实现的 只要在读取值时收集观察者 在赋值时触发观察者更新函数 就可以实现数据变更 从而实现DOM重新渲染

说到这可能还不是很明白 不要急 慢慢来 先看一下这段代码 复制放到HTML文件里自己运行一下 然后打开网页 在控制台里输入data.user.name看看 会有惊喜

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>动态数据绑定（一）</title>  
</head>  
<body>  
 <script>  
 var data = {  
 user: {  
 name: 'xiaoming',  
 age: 18,  
 occupation: 'frontend'  
 },  
 address: {  
 city: 'shaoguan'  
 }  
 };   
 function Observer(data) {  
 this.data = data;  
 this.walk(data);  
 }  
 Observer.prototype = {  
 walk: function(obj) {  
 var value,  
 key;  
 for (key in obj) {  
 if (obj.hasOwnProperty(key)) {  
 value = obj[key];  
 if (typeof value === 'object') {  
 new Observer(value);  
 }  
 this.convert(key, value);   
 }   
 }  
 },  
 convert: function(key, value) {  
 Object.defineProperty(this.data, key, {  
 get : function(){   
 console.log("你访问了" + key);  
 return value;   
 },  
 set : function(newValue){   
 value = newValue;   
 console.log('你设置了' + key + '=' + value);  
 }  
 });  
 }  
 }   
 var example = new Observer(data);  
 </script>   
</body>  
</html>

## vue-router原理

说简单点，vue-router的原理就是通过对URL地址变化的监听，继而对不同的组件进行渲染。

每当URL地址改变时，就对相应的组件进行渲染。原理是很简单，实现方式可能有点复杂，主要有hash模式和history模式。

如果想了解得详细点，请查看[面试官: 你了解前端路由吗?](https://juejin.im/post/6844903589123457031)。

## vuex原理

vuex的原理其实非常简单，它为什么能实现所有的组件共享同一份数据？ 因为vuex生成了一个store实例，并且把这个实例挂在了所有的组件上，所有的组件引用的都是同一个store实例。 store实例上有数据，有方法，方法改变的都是store实例上的数据。由于其他组件引用的是同样的实例，所以一个组件改变了store上的数据， 导致另一个组件上的数据也会改变，就像是一个对象的引用。 如果对vuex的实现有兴趣，可以看看我自己造的一个vue轮子对应的[vuex插件](https://github.com/woai3c/mini-vuex/blob/master/mini-vuex.js)。它实现了除vuex模块外的所有功能。

## v-if和v-show的区别

v-if 是“真正”的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建。

v-if 也是惰性的：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。

相比之下，v-show 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 进行切换。

一般来说，v-if 有更高的切换开销，而 v-show 有更高的初始渲染开销。因此，如果需要非常频繁地切换，则使用 v-show 较好；如果在运行时条件很少改变，则使用v-if 较好。

https://cn.vuejs.org/v2/guide/conditional.html#v-if-vs-v-show

## vue怎么实现页面的权限控制

利用 vue-router 的 beforeEach 事件，可以在跳转页面前判断用户的权限（利用 cookie 或 token），是否能够进入此页面，如果不能则提示错误或重定向到其他页面，在后台管理系统中这种场景经常能遇到。

## keep-alive有什么作用

在 Vue 中，每次切换组件时，都会重新渲染。如果有多个组件切换，又想让它们保持原来的状态，避免重新渲染，这个时候就可以使用 keep-alive。 keep-alive 可以使被包含的组件保留状态，或避免重新渲染。

## 计算属性有什么作用

先来看一下计算属性的定义：

当其依赖的属性的值发生变化的时，计算属性会重新计算。反之则使用缓存中的属性值。

计算属性和vue中的其它数据一样，都是响应式的，只不过它必须依赖某一个数据实现，并且只有它依赖的数据的值改变了，它才会更新。

## router的区别

$route 是路由信息对象，包括path，params，hash，query，fullPath，matched，name 等路由信息参数。

而 $router 是路由实例对象，包括了路由的跳转方法，钩子函数等

## watch的作用是什么

watch 主要作用是监听某个数据值的变化，它和计算属性很相似。

借助 watch 还可以做一些特别的事情，例如监听页面路由，当页面跳转时，我们可以做相应的权限控制，拒绝没有权限的用户访问页面。

watch 与计算属性的区别有两个：一是计算属性依赖其他属性和有缓存，它没有；二是 watch 可以进行异步操作。

为什么计算属性不能进行异步操作？因为计算属性必须将计算后的值 return 回去，如果在计算属性中使用异步操作，那会返回一个 undefined：

computed: {  
 test() {  
 let value  
 setTimeout(() => value = 10)  
 return value  
 }  
}

就像上述代码，还没等异步操作完成就已经执行 return value 了。

如果把 test() 改成 async test() 再结合 await 来进行异步操作呢？

async test() {  
 let value = 1  
 value = await new Promise(r => r(10))  
 return value  
}

也不行，因为 async() 返回的是一个 Promise。

## vue-loader是什么？使用它的用途有哪些？

vue-loader 是解析 .vue 文件的一个加载器，将 template/js/style 转换成 js 模块。

用途：js 可以写 es6、style 样式可以 scss 或 less；template 可以加 jade 等。

## 假设定义了一个数组a=[1,2,3]，相应的，页面上显示的值为1,2,3，现设a[0]=5，页面上的值会变成5,2,3吗？为什么？

不会

因为 Vue 是使用 Object.defineProperty 来监听数值变化的，而直接修改数组的值的这种操作无法监听。

例如：vm.items[indexOfItem] = newValue 这种操作是无法监听的。

如果需要直接修改数组元素的值，可以使用 Vue.set

Vue.set(vm.items, indexOfItem, newValue)

[数组更新检测](https://cn.vuejs.org/v2/guide/list.html#%E6%95%B0%E7%BB%84%E6%9B%B4%E6%96%B0%E6%A3%80%E6%B5%8B)

## 父子组件生命周期执行顺序

// 渲染  
parent beforeCreate  
parent created  
parent beforeMount  
sub beforeCreate  
sub created  
sub beforeMount  
sub mounted  
parent mounted  
  
// 数据更新  
parent beforeUpdate  
sub beforeUpdate  
sub updated  
parent updated  
  
// 销毁组件  
parent beforeDestroy  
sub beforeDestroy  
sub destroyed  
parent destroyed

注意 mounted 不会保证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，可以在 mounted 内部使用 vm.$nextTick：

mounted: function () {  
 this.$nextTick(function () {  
 // Code that will run only after the  
 // entire view has been rendered  
 })  
}

## 实现虚拟 DOM 和 diff 算法

* 一个 DOM 具有非常多的属性，DOM 进行增删操作开销大。所以使用虚拟 DOM 能有效降低消耗。

[如何理解虚拟DOM?](https://www.zhihu.com/question/29504639)