组件

组件：实现 局部 功能的 代码 和 资源 (css、inage)的集合

非单文件组件

一个文件中包含多个组件

补充：

```
function data () {
   return {a:1,b:2}
}

const x1 = data();
const x2 = data();
```

> 这里的data方法每次调用都会返回一个新的对象（通过字面量构建一个新的对象）
>
> 对 x1 的更改 不会影响 x2

定义-注册-使用

定义

如何定义一个组件?

使用 Vue.extend (options)创建，其中options和new Vue(options)时传入的那个options几乎一样，但也有点区别；

区别如下：

1. el不要写 ，为什么？ —— 最终所有的组件都要经过一个vm的管理，由vm中的el决定服务哪个容器。

2. data必须写成函数 ，为什么？ ——— 避免组件被复用时，数据存在引用关系。

备注：使用template可以配置组件结构。

注意：

组件名：

脚手架：每一个单词首字母大写

html里面：全小写+‘-’ my-name

组件配置项 name

配置了name name开发者工具里面组件就是配置的name 不管怎么注册的

```
//第一步：创建school组件
const school = Vue.extend({
  template: `
                <div class="demo">
                    <h2>学校名称：{{schoolName}}</h2>
                    <h2>学校地址：{{address}}</h2>
                    <button @click="showName">点我提示学校名</button>
                </div>
            `,
  // el:'#root', //组件定义时，一定不要写el配置项，因为最终所有的组件都要被一个vm管理，由vm决定服务于哪个容器。
  data() {
    return {
      schoolName: "尚硅谷",
      address: "北京昌平",
    };
  },
  methods: {
    showName() {
      alert(this.schoolName);
    },
  },
});
```

注册

1.全局注册

```
Vue.component('my-component-name', xxx})
```

2.局部注册

```
new Vue({
  el: '#app',
  components: {
    'component-a': ComponentA,
    'component-b': ComponentB
  }
}
```

使用

```
<school></school>
```

VueComponent

关于VueComponent：

1.school组件本质是一个名为VueComponent的构造函数，且不是程序员定义的，是Vue.extend生成的。

2.我们只需要写或，Vue解析时会帮我们创建school组件的实例对象， 即Vue帮我们执行的：new VueComponent(options)。

3.特别注意：**每次调用Vue.extend，返回的都是一个全新的VueComponent！！！！**

4.关于this指向：

(1). `组件` 配置中：

data函数、methods中的函数、watch中的函数、computed中的函数 它们的this均是【**VueComponent实例对象**】。

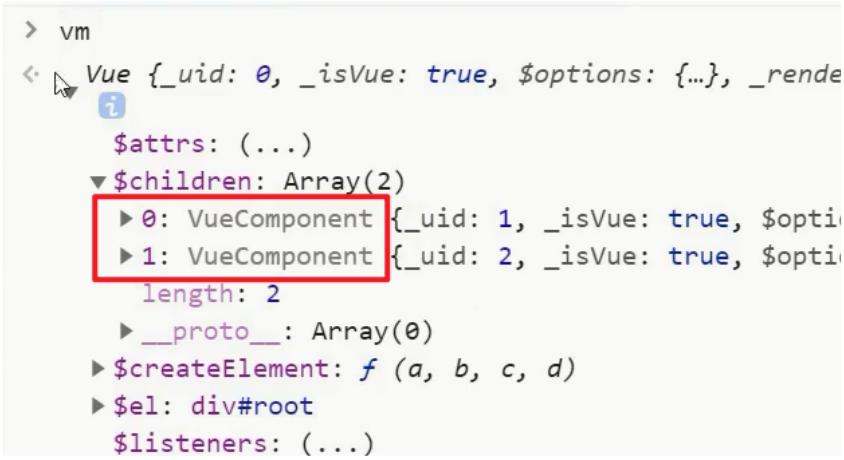
(2). `new Vue(options)` 配置中：

data函数、methods中的函数、watch中的函数、computed中的函数 它们的this均是【**Vue实例对象**】。

5.VueComponent的实例对象，以后简称vc（也可称之为：组件实例对象）。

Vue的实例对象，以后简称vm。

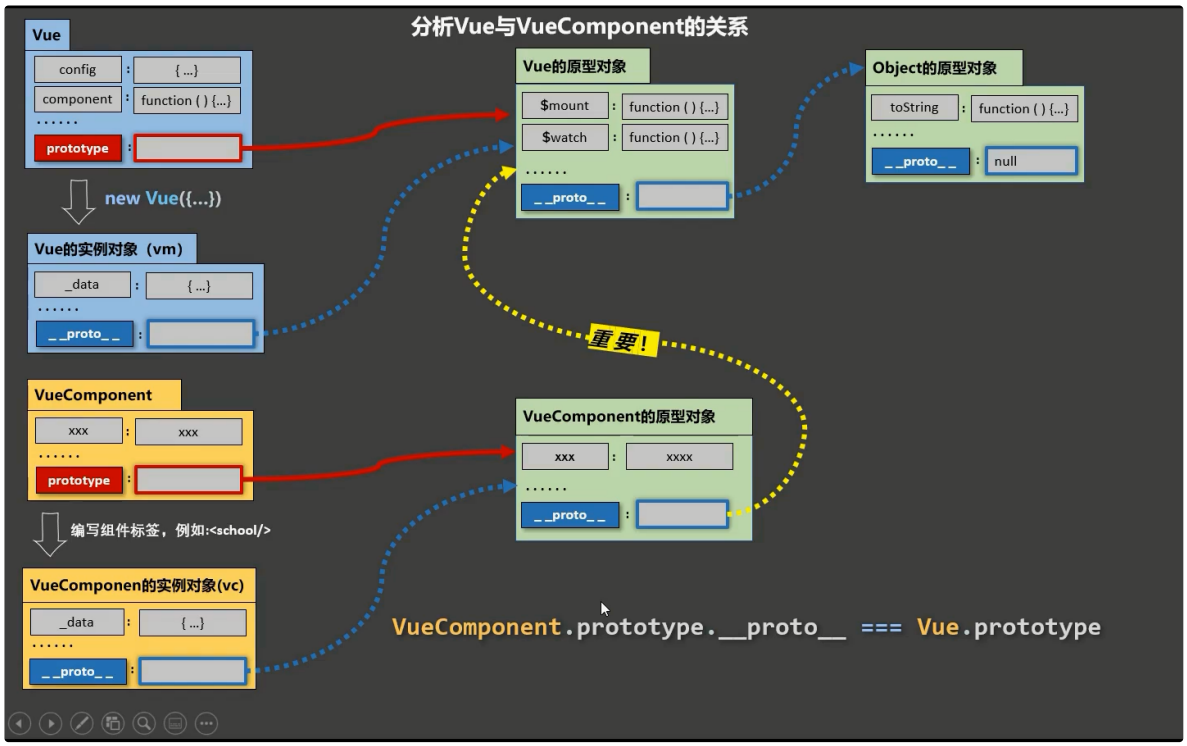
vm管理着vc



vm和vc的关系

1.一个重要的内置关系：VueComponent.prototype.proto === Vue.prototype

2.为什么要有这个关系：让组件实例对象（vc）可以访问到 Vue原型上的属性、方法。



原型

```
//定义一个构造函数
function Demo() {
  this.a = 1;
  this.b = 2;
}
//创建一个Demo的实例对象
const d = new Demo();

console.log(Demo.prototype); //显示原型属性

console.log(d.__proto__); //隐式原型属性

console.log(Demo.prototype === d.__proto__);

//程序员通过显示原型属性操作原型对象，追加一个x属性，值为99
Demo.prototype.x = 99;

console.log("@", d);
```

```
Vue.prototype.$bus = this //安装全局事件总线

this.$bus.$emit('hello',this.name)

this.$bus.$on('hello',(data)⇒{})
```