一：组件化和模块化的区别

模块：按照逻辑来分层

组件：根据界面来分层

二：创建组件的三种方式

创建一个全局的Vue组件

1:通过Vue.extend方法，接收一个返回值 是创建出来的组件模板对象，里面传一个对象

通过 template 属性，把组件的结构写进去

2:Vue.component("组件名称-驼峰命名规则")

注调用组件的时候，要全部写成小写，中间的驼峰的地方要用“-”连接

```
<body>
  <div id="app">
    <!-- 如果要使用组件，直接，把组件的名称，以 HTML 标签的形式，引入到页面中，即可 -->
    <my-com1></my-com1>
  </div>

  <script>
    // 1.1 使用 Vue.extend 来创建全局的Vue组件
    var com1 = Vue.extend({
      template: '<h3>这是使用 Vue.extend 创建的组件</h3>' // 通过 template 属性，指定了组件要
      展示的HTML结构
    })
    // 1.2 使用 Vue.component('组件的名称', 创建出来的组件模板对象)
    Vue.component('myCom1', com1)


    // 创建 Vue 实例，得到 ViewModel
    var vm = new Vue({ ···
    });
  </script>
```

```
<script>
  // 1.1 使用 Vue.extend 来创建全局的Vue组件
  var com1 = Vue.extend({
    template: '<h3>这是使用 Vue.extend 创建的组件</h3>' // 通过 template 属性，指定了组件要
    展示的HTML结构
  })
  // 1.2 使用 Vue.component('组件的名称', 创建出来的组件模板对象)
  // Vue.component('myCom1', com1)
  // 如果使用 Vue.component 定义全局组件的时候，组件名称使用了 驼峰命名，则在引用组件的时候，需要
  把 大写的驼峰改为小写的字母，同时，两个单词之前，使用 - 链接；
  // 如果不使用驼峰，则直接拿名称来使用即可；
  Vue.component('mycom1', com1)


  // 创建 Vue 实例，得到 ViewModel
  var vm = new Vue({…
  });
```

将两个步骤合并成了一步：

写法如下：

```
// Vue.component 第一个参数:组件的名称,将来在引用组件的时候,就是一个 标签形式 来引入 它的
// 第二个参数: Vue.extend 创建的组件 ,其中 template 就是组件将来要展示的
Vue.component('mycom1', Vue.extend({
  template: '<h3>这是使用 Vue.extend 创建的组件</h3>'
}))


// 创建 Vue 实例，得到 ViewModel
```

2：创建的第二种方式：写法更简便

```
<body>
  <div id="app">
    <!-- 还是使用 标签形式,引入自己的组件 -->
    <mycom2></mycom2>
  </div>

  <script>
    Vue.component('mycom2', {
      // 注意:不论是哪种方式创建出来的组件,组件的 template 属性指向的模板内容,必须有且只能有唯一的一
      个根元素
      template: '<div><h3>这是直接使用 Vue.component 创建出来的组件
      </h3><span>123</span></div>'
    })

    // 创建 Vue 实例，得到 ViewModel
```

3:创建的第3种方式：结构更好一点

```html
<body>
  <div id="app">
    <mycom3></mycom3>
  </div>

  <!-- 在 被控制的 #app 外面使用 template 元素,定义组件的HTML模板结构 -->
  <template id="tmpl">
    <div>
      <h1>这是通过 template 元素,在外部定义的组件结构,这个方式,有代码的只能提示和高亮</h1>
      <h4>好用,不错!</h4>
    </div>
  </template>

  <script>
    Vue.component('mycom3', {
      template: '#tmpl'
    })
```

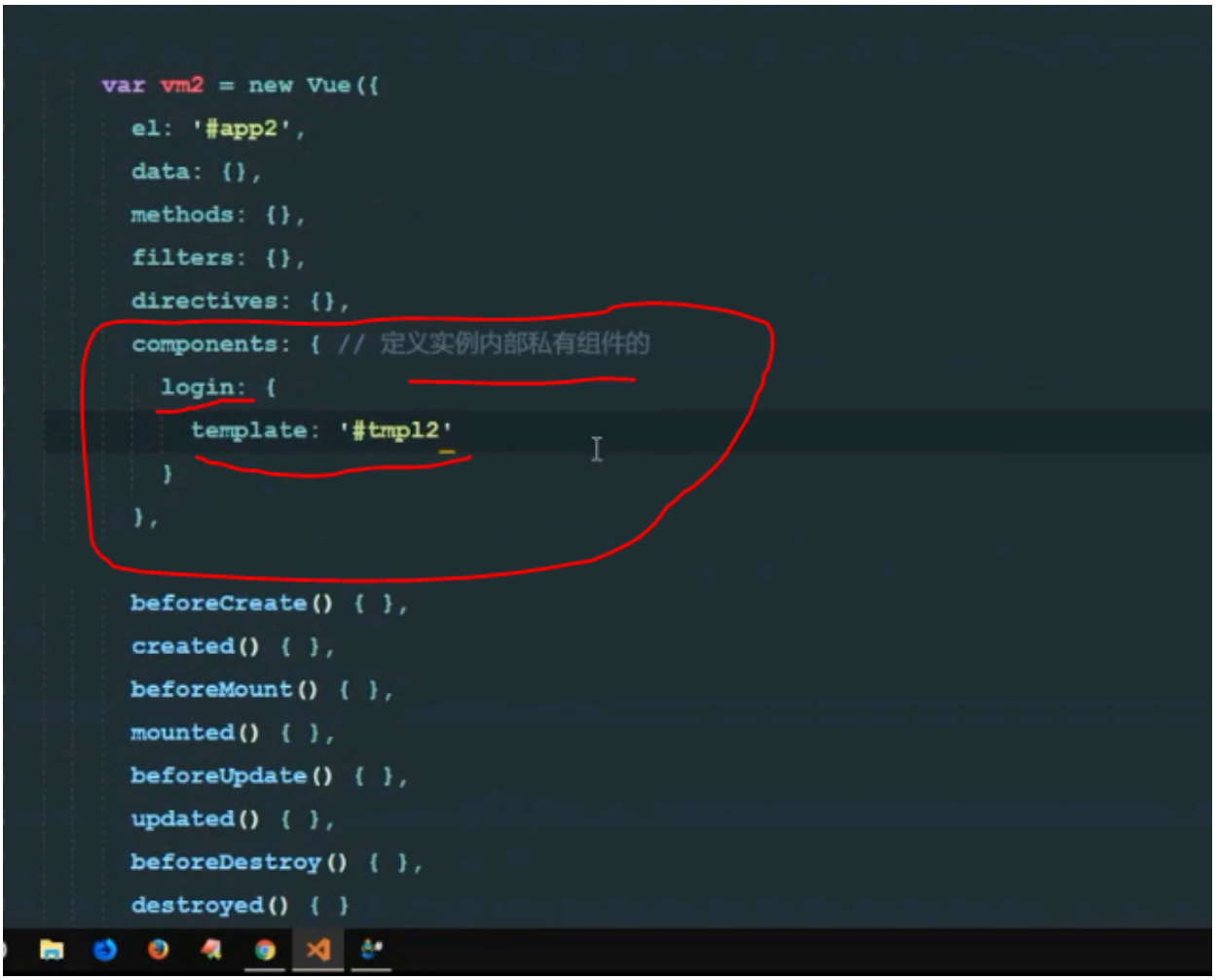
所有的参数

```
var vm2 = new Vue({
    el: '#app2',
    data:{},
    methods:{},
    filters:{},
    directives:{},
    components:{

    },

    beforeCreate(){},
    created(){},
    beforeMount(){},
    mounted(){},
    beforeUpdate(){},
    updated(){},
    beforeDestroy(){},
    destroyed(){}
})
```

定义私有组件：



三：组件中的data对象

必须是一个函数，并且需要返回一个对象。

```
<script>
  // 1. 组件可以有自己的 data 数据
  // 2. 组件的 data 和 实例的 data 有点不一样,实例中的 data 可以为一个对象,但是 组件中的 data
  必须是一个方法
  // 3. 组件中的 data 除了必须为一个方法之外,这个方法内部,还必须返回一个对象才行;
  // 4. 组件中 的data 数据,使用方式,和实例中的 data 使用方式完全一样!!!
  Vue.component('mycom1', {
    template: '<h1>这是全局组件 --- {{msg}}</h1>',
    data: function () {
      return {
        msg: '这是组件的中data定义的数据'
      }
    }
  })

  // 创建 Vue 实例,得到 ViewModel
  var vm = new Vue({
    el: '#app',
    data: {},
    methods: {}
```

四：组件的稍微的使用

```
<body>
  <div id="app">
    <counter></counter>
    <hr>
    <counter></counter>
    <hr>
    <counter></counter>
  </div>


  <template id="tmpl">
    <div>
      <input type="button" value="+1" @click="increment">
      <h3>{{count}}</h3>
    </div>
  </template>

  <script>
    var dataObj = { count: 0 }
```

五：登陆注册组件的切换

```
<script>
  var dataObj = { count: 0 }

  // 这是一个计数器的组件，身上有个按钮，每当点击按钮，让 data 中的 count 值 +1
  Vue.component('counter', {
    template: '#tmpl',
    data: function () {
      // return dataObj
      return { count: 0 }
    },
    methods: {
      increment() {
        this.count++
      }
    }
  })

  // 创建 Vue 实例，得到 ViewModel
```

登陆注册组件的切换2

```
<div id="app">
    <!-- <a href="" @click.prevent="flag=true">登陆</a>
    <a href="" @click.prevent="flag=false">注册</a>
    <login v-if='flag'></login>
    <register v-else="flag"></register> -->
    <a href="" @click.prevent="comName=login">登陆</a>
    <a href="" @click.prevent="comName=register">注册</a>

   <component :is="comName"></component>
</div>
<script>
    Vue.component('login', {
        template: '<h3>登陆组件</h3>'
    });
    Vue.component('register', {
        template: "<h3>注册组件</h3>"
    })

    var vm = new Vue({
        el: "#app",
        data: {
            comName:"register"
        },
        methods: {}
    });


</script>
```

默认

```
<!-- Vue提供了 component ,来展示对应名称的组件 -->
<!-- component 是一个占位符, :is 属性,可以用来指定要展示的组件的名称 -->
<component :is="comName"></component>
```

```
<component :is="comName"></component>

<!-- 总结:当前学习了几个 Vue 提供的标签了??? -->
<!-- component, template, transition, transitionGroup -->
```

六：通过transition 标签的mode属性，切换组件的动画切换方式，out-in 代表先出去再进来

```
<a href="" @click.prevent="comName='register'">注册</a>

<!-- 通过 mode 属性,设置组件切换时候的 模式 -->
<transition mode="out-in">
  <component :is="comName"></component>
</transition>
```

七：flag标识符的作用

```
},
afterEnter(el) {
    // 这句话， 第一个功能，是控制小球的显示与隐藏
    // 第二个功能： 直接跳过后半场动画，让 flag 标识符 直接变为 false
    // 当第二次再点击 按钮的时候， flag false -> true
    this.flag = !this.flag
    // el.style.opacity = 0.5

    // Vue 把一个完整的动画，使用钩子函数，拆分为了两部分：
    // 我们使用 flag 标识符，来表示动画的切换；
    // 刚以开始，flag = false -> true -> false
}
```