# JavaScript从初级往高级走系列————MVVM-Vue

 发表于 2018-05-31 |  分类于 [vue](https://finget.github.io/categories/vue/)|  [2](https://finget.github.io/2018/05/31/mvvm-vue/" \l "comments)

JavaScript从初级往高级走系列，一次学习记录吧，一直在路上

## **MVVM**

* 如何理解 MVVM
* 如何实现 MVVM
* 是否解读过 Vue 的源码

### **Jquery 与 框架的区别**

#### **jquery 实现 todo-list**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 | <div>  <input type="text" name="" id="txt-title">  <button id="btn-submit">submit</button> </div> <div>  <ul id="ul-list"></ul> </div>  <script src="https://cdn.bootcss.com/jquery/3.2.1/jquery.min.js"></script> <script type="text/javascript">  var $txtTitle = $('#txt-title');  var $btnSubmit = $('#btn-submit');  var $ulList = $('#ul-list');  $btnSubmit.click(function () {  var title = $txtTitle.val();  if (!title) {  return  }  var $li = $('<li>' + title + '</li>');  $ulList.append($li);  $txtTitle.val('');  }) </script> |

#### **vue 实现 todo-list**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 | <div id="app">  <div>  <input v-model="title">  <button v-on:click="add">submit</button>  </div>  <div>  <ul>  <li v-for="item in list">{{item}}</li>  </ul>  </div> </div>  <script type="text/javascript">  // data 独立  var data = {  title: '',  list: []  }  // 初始化 Vue 实例  var vm = new Vue({  el: '#app',  data: data,  methods: {  add: function () {  this.list.push(this.title);  this.title = '';  }  }  }) </script> |

#### **两者的区别**

数据和视图的分离，解耦（开放封闭原则，对扩展开放，对修改封闭）

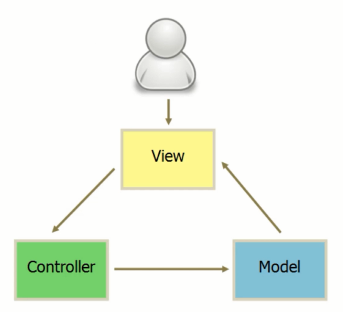
在jQuery中在jQuery代码中操作视图和数据，混在一块了

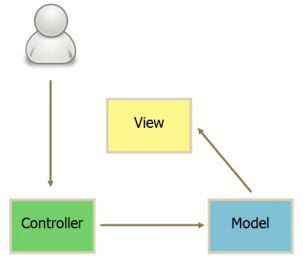
以数据驱动视图，只关心数据变化，DOM操作被封装

只改数据，视图自动更新

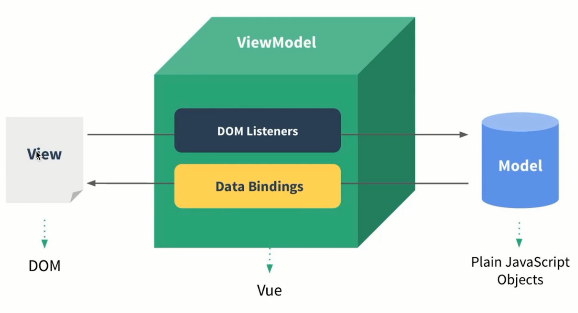
### **MVVM的理解**

MVC （Model View Controller）

[](https://i.imgur.com/VBt5082.png)

[](https://i.imgur.com/I8pCHF8.png)

MVVM （Model View ViewModel）

[](https://i.imgur.com/nbERAJ8.png)

View 通过 事件绑定 (DOM Listeners) 操作Model; Model通过 数据绑定 (Data Bindings)操作View。

## **Vue 三要素**

* 响应式： Vue 如何监听到 data 的每个属性变化？
* 模板引擎： Vue 的模板如何被解析，指令如何处理？
* 渲染：Vue 的模板如何被渲染成Html？

### **Vue中如何实现响应式**

#### **什么是响应式**

* 修改 data 属性之后， Vue 立刻监听到
* data 属性被代理到 vm上

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 | var vm = new Vue({  el: '#app',  data: {  name: 'zhangsan',  age: 20  } }) // vm.name = 'zhangsan' // vm.age = '20' |

#### **Object.defineProperty，Vue核心函数**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 | var obj = {  name: 'zhangsan',  age: 25 } console.log(obj.name); // 获取属性的时候，如何监听 obj.age = 26; // 赋值属性的时候，如何监听 |

上面是无法监听对象的属性的访问以及赋值操作的，直接就产生了操作的结果。

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | var obj = {} var \_name = 'shangsan' Object.defineProperty(obj, 'name', {  get: function () {  console.log('get', \_name) // 监听  return \_name  },  set: function (newVal) {  console.log('set', newVal) // 监听  \_name = newVal  } }) console.log(obj.name); // 可以监听到 obj.name = 'lisi'; // 可以监听到 |

### **Vue 中何如解析模板**

#### **模板是什么**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 | <div id="app">  <div>  <input v-model="title">  <button v-on:click="add">submit</button>  </div>  <div>  <ul>  <li v-for="item in list">{{item}}</li>  </ul>  </div> </div> |

* 本质： 字符串
* 有逻辑， 如v-if v-for 等
* 与 html 标签格式很像，但有很大区别（html是静态的，模板是动态的）
* 最终还要转换为 html 来显示
  + 模板最终必须转换成 JS 代码
  + 有逻辑（v-if v-for 等），必须用JS才能实现（图灵完备）
  + 因此，模板最重要转成一个JS函数（render函数）

#### **render函数**

##### **with – 实际开发不推荐用**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 | var obj = {  name: 'zhangsan',  age: 20,  getAddress: function () {  alert('beijing')  } } // 不使用with function fn() {  alert(obj.name)  alert(obj.age)  obj.getAddress() } fn()  // 使用with function fn1() {  with(obj) {  alert(age)  alert(name)  getAddress()  } } fn1() |

##### **render**

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 | <div id="app">  <p>{{price}}</p> </div>  <script>  var vm = new Vue({  el: '#app',  data: {  price: 100  }  }) </script> |

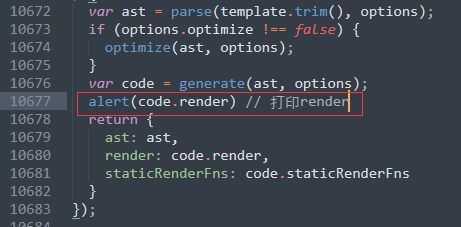
模板将变成下面这个样子：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 | function render() {  with(this) { // this 就是 vm  return \_c(  'div',  {  attrs: {'id': 'app'}  },  [  \_c('p', [\_v(\_s(price))])  ]  )  } } |

[](https://i.imgur.com/anzI5vL.png)

##### **看todo-list的render**

在vue源码里alert render 函数

[](https://i.imgur.com/uBpSBab.png)

以上面vue实现的todolist为例：

|  |  |
| --- | --- |
| 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 | with(this){ // this 就是 vm  return \_c( // \_c创建一个标签  'div',  {  attrs:{"id":"app"}  },  [  \_c(  'div',  [  \_c(  'input',  {  directives:[  {  name:"model",  rawName:"v-model",  value:(title),  expression:"title"  }  ],  domProps:{  "value":(title)  },  on:{  "input":function($event){  if($event.target.composing)return;  title=$event.target.value  }  }  }  ),  \_v(" "),  \_c(  'button',  {  on:{  "click":add  }  },  [\_v("submit")]  )  ]  ),  \_v(" "),  \_c('div',  [  \_c(  'ul',  \_l((list),function(item){return \_c('li',[\_v(\_s(item))])}) // \_l 解析 v-for 循环  )  ]  )  ]  ) } |

##### **render 与 Vdom**

[可以先看一下virtualDom](https://finget.github.io/2018/05/22/virtualDom/" \t "https://finget.github.io/2018/05/31/mvvm-vue/_blank)

* vm.\_c 其实相当于 snabbdom 中的 h 函数
* render 函数执行之后，返回的是 vnode

[](https://i.imgur.com/2C0Lz42.png)

* updateComponent 中实现了 vdom 的 patch
* 页面首次渲染 执行updateComponent
* data 中每次修改属性， 执行 updateComponent

##### **vue 的整个实现流程**

* 第一步： 解析模板成render函数
  + with 的用法
  + 模板中的所有信息都被render函数包含
  + 模板中用到的data中的属性，都变成了js变量
  + 模板中的v-model v-if v-on 都变成了 js逻辑
  + render 函数返回 vnode

[](https://i.imgur.com/vXDR2CX.png)  
[](https://i.imgur.com/IM7NA6B.png)

* 第二部： 响应式开始监听
  + Object.defineProperty
  + 将 data 的属性代理到 vm 上

[](https://i.imgur.com/gA2rWmu.png)

* 第三步： 首次渲染，显示页面，且绑定依赖

[](https://i.imgur.com/1b0LjjK.png)

* 第四步： data 属性变化，触发 rerender