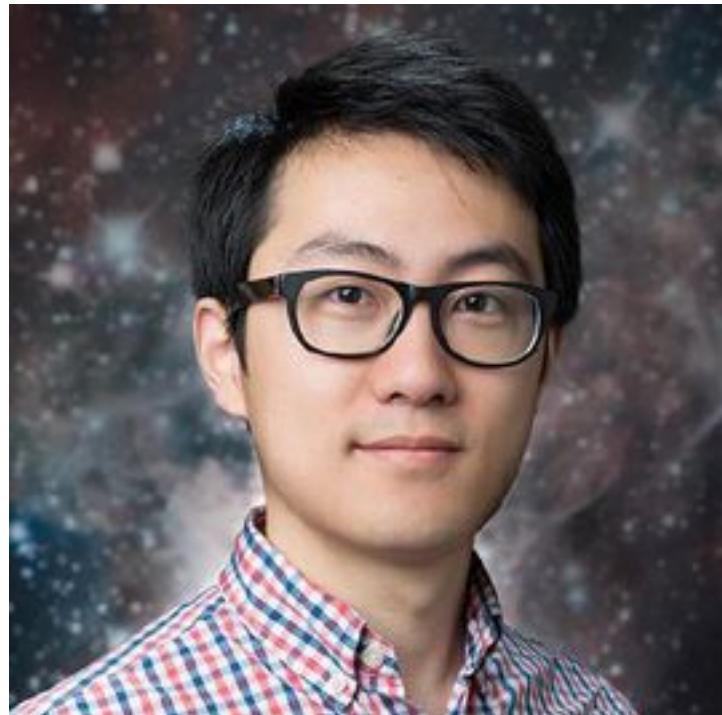


W

WEICHUANG

謝成





尤雨溪

[**Vue.js**](#)框架的作者，尤雨溪毕业于上海复旦附中，在美国完成大学学业，本科毕业于**Colgate University**，后在**Parsons**设计学院获得**Design & Technology**艺术硕士学位，现任职于纽约**Google Creative Lab**。

2014年2月，开源了一个前端开发库 **Vue.js**。**Vue.js** 是的组件构建 Web 界面的 [**JavaScript**](#) 库，是一个通过简洁的**API** 提供高效的数据绑定和灵活系统。

什么是Vue.js



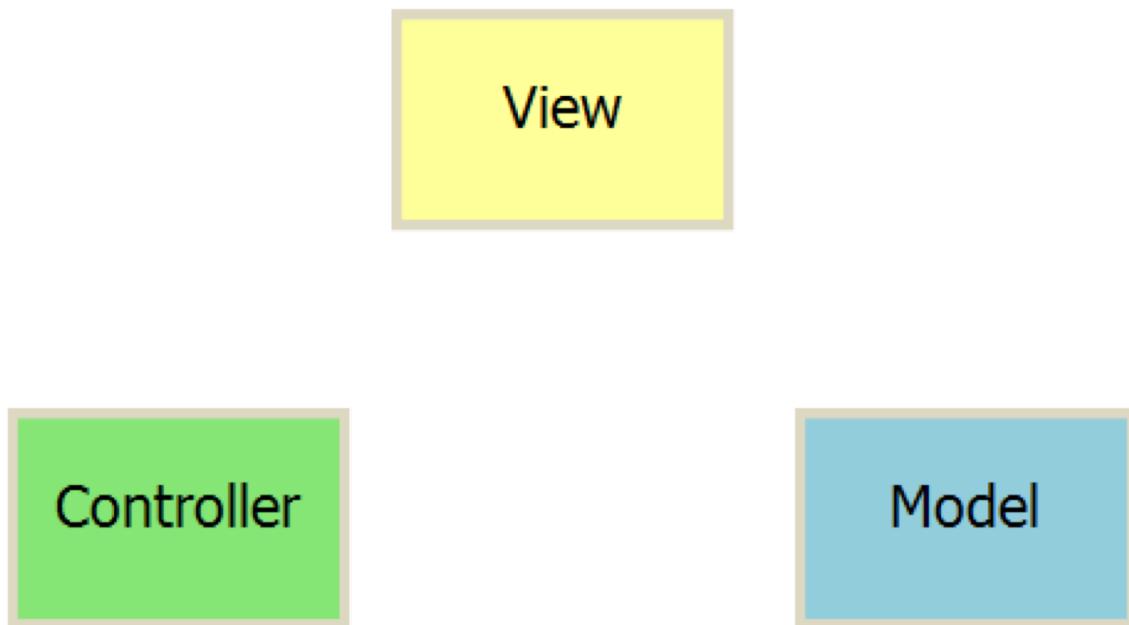
网站: <https://cn.vuejs.org/>

Github: <https://github.com/vuejs/vue>

Vue.js (读音 /vju:/, 类似于 **view**) 是一套构建用户界面的渐进式框架。与其他重量级框架不同的是, **Vue** 采用自底向上增量开发的设计。**Vue** 的核心库只关注视图层, 它不仅易于上手, 还便于与第三方库或既有项目整合。另一方面, 当与[单文件组件](#)和[Vue 生态系统支持的库](#)结合使用时, **Vue** 也完全能够为复杂的单页应用程序提供驱动。

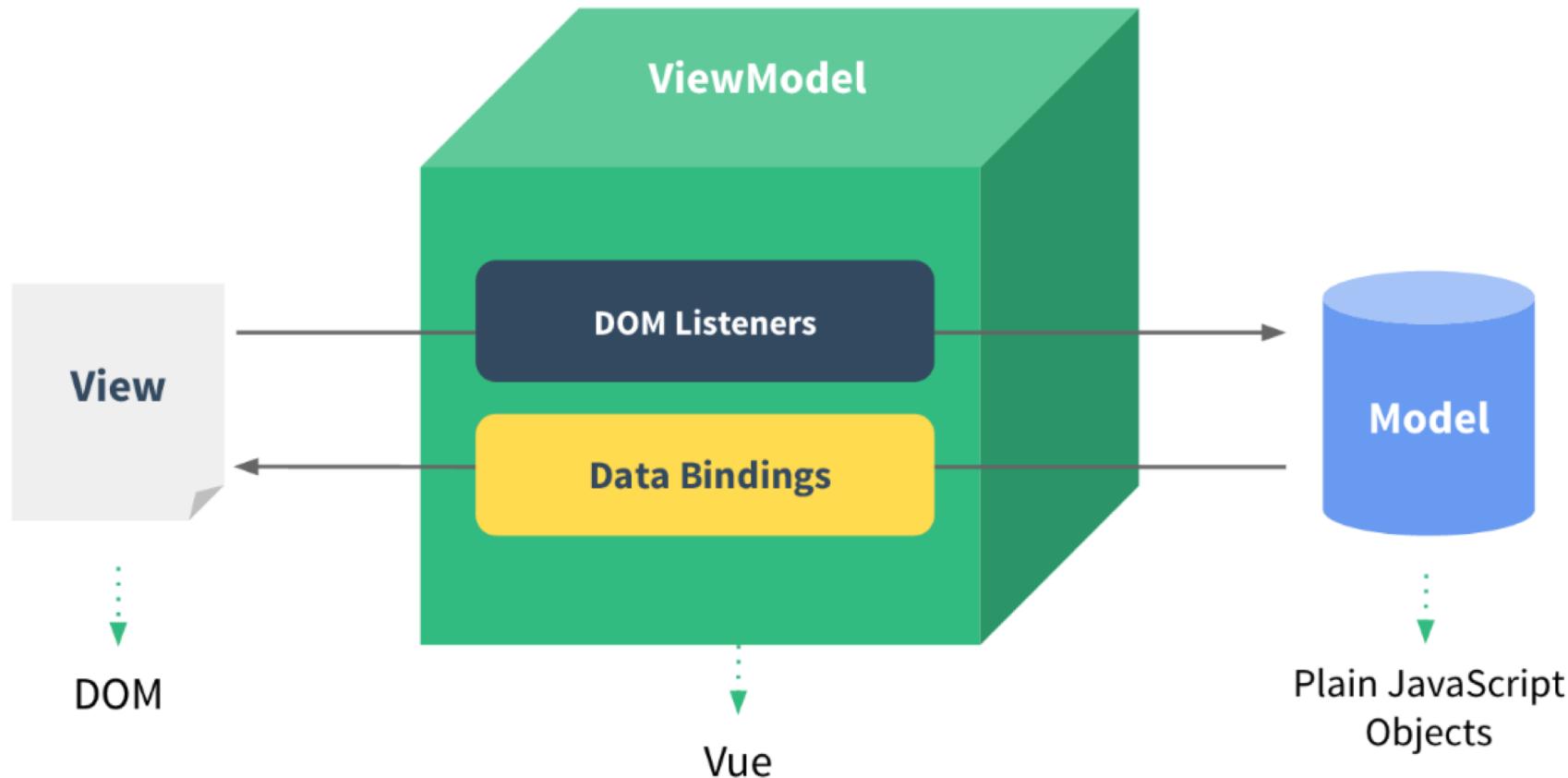
[推荐文章1](#)
[推荐文章2](#)

MVC



- 视图（View）：用户界面。
- 控制器（Controller）：业务逻辑
- 模型（Model）：数据保存
- V-C-M-C-V

MVVM



1. Git检出项目：<https://github.com/vuejs/vue-devtools.git>
2. npm install
3. npm run build
4. 添加至chrome浏览器扩展程序



引入Vue.js

script标签引入

```
<script src="vue.js"></script>
```

CDN

```
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
```

- **Hello World**
- **v-if v-else v-else-if v-show**
- **v-for** 数组、对象
- **v-text & v-html**
- **v-on \$event @**
- **v-model lazy、number、trim 表单**
- **v-bind :** 绑定属性
- **v-pre 原样输出**
- **v-cloak 渲染完成后显示**
- **v-once 只渲染一次**

Vue.directive 自定义指令

```
Vue.directive('weichuang', {  
  bind: function (el, binding, vnode) {//被绑定  
    console.log('1 - bind');  
    el.style = 'background:' + binding.value;  
  },  
  inserted: function () {//绑定到节点  
    console.log('2 - inserted');  
  },  
  update: function () {//组件更新  
    console.log('3 - update');  
  },  
  componentUpdated: function () {//组件更新完成  
    console.log('4 - componentUpdated');  
  },  
  unbind: function () {//解绑  
    console.log('1 - bind');  
  }  
});
```

- **bind:**只调用一次，指令第一次绑定到元素时调用，用这个钩子函数可以定义一个绑定时执行一次的初始化动作。
- **inserted:**被绑定元素插入父节点时调用（父节点存在即可调用，不必存在于**document**中）。
- **update:**被绑定于元素所在的模板更新时调用，而无论绑定值是否变化。通过比较更新前后的绑定值，可以忽略不必要的模板更新。
- **componentUpdated:**被绑定元素所在模板完成一次更新周期时调用。
- **unbind:**只调用一次，指令与元素解绑时调用。**vm.\$destroy();**

Vue.set 扩展实例构造器

```
new Vue({
  el: '#app',
  data: {
    list: ['a', 'b', 'c']
  },
  methods: {
    change: function(){
      // this.list[1] = 'd';
      Vue.set(this.list, 1, 'd');
    }
  }
});
```

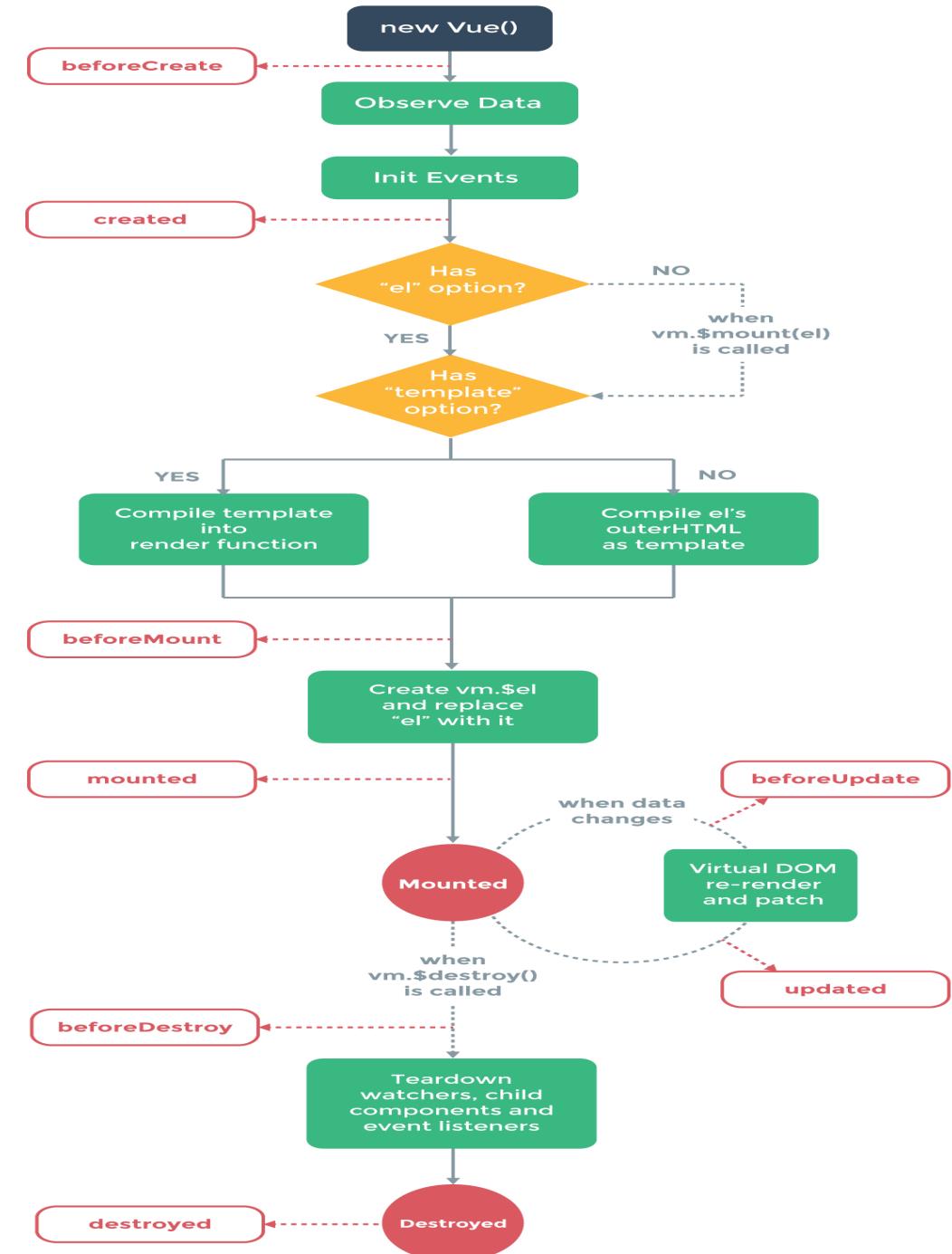
- **Vue**不能通过索引设置数组的值

生命周期

Lifecycle hooks

生命周期钩子应该算 vue 这次升级中 broken changes 最多的一部分了，对照 1.0 的[文档](#)和[release note](#)，作了下面这张表

vue 1.0+	vue 2.0	Description
init	beforeCreate	组件实例刚被创建，组件属性计算之前，如 data 属性等
created	created	组件实例创建完成，属性已绑定，但 DOM 还未生成，\$el 属性还不存在
beforeCompile	beforeMount	模板编译/挂载之前
compiled	mounted	模板编译/挂载之后
ready	mounted	模板编译/挂载之后（不保证组件已在 document 中）
-	beforeUpdate	组件更新之前
-	updated	组件更新之后
-	activated	for <code>keep-alive</code> ，组件被激活时调用
-	deactivated	for <code>keep-alive</code> ，组件被移除时调用
attached	-	不用了还说啥哪...
detached	-	那就不说了吧...
beforeDestory	beforeDestory	组件销毁前调用
destoryed	destoryed	组件销毁后调用



template 模板

- 一个字符串模板作为 Vue 实例的标识使用。模板将会 替换 挂载的元素。挂载元素的内容都将被忽略，除非模板的内容有分发插槽。

component 1

```
<div id="app">
  <weichuang></weichuang>
  <my-component :aa='msg'></my-component>
</div>

<script src="vue.js"></script>
<script>
  Vue.component('weichuang', {
    template: '<h1>这是一个全局组件</h1>'
  });
  new Vue({
    el: '#app',
    data: {
      msg: 'hello'
    },
    components: {
      'my-component': {
        template: '<h1>这是一个局部组件， {{aa}}</h1>',
        props: ['aa']
      }
    }
  );
</script>
```

- 组件 (**Component**) 是 **Vue.js** 最强大的功能之一。组件可以扩展 **HTML** 元素，封装可重用的代码。在较高层面上，组件是自定义元素，**Vue.js** 的编译器为它添加特殊功能。
- 全局组件
- 局部组件
- **props:[]**
- 参数中带-，去掉横杠，后面首字母大写
- **data**的值传递给组件：**v-bind**

component 2

```
let web = {
    template: '<h1>{{fe}}</h1>',
    props: ['fe']
};

let wcwx = {
    template: `
        <div>
            <h1>唯创网讯</h1>
            <web fe='最专业的web前端'></web>
        </div>
        `,
    components: {
        "web": web
    }
};

new Vue({
    el: '#app',
    components: {
        weichuang: wcwx
    }
});
```

- 组件嵌套
- 组件传值
- :is 切换组件
- .native 监听组件根元素的原生事件

slot

```
<div id="app">
  <user>
    <span slot="name">{{user.name}}</span>
    <span slot="age">{{user.age}}</span>
  </user>
</div>

<template id="temp">
  <div>
    <div>名字是:
      <slot name="name"></slot>
    </div>
    <div>年龄是:
      <slot name="age"></slot>
    </div>
  </div>
</template>
```

```
var user = {
  template: '#temp'
};
new Vue({
  el: '#app',
  data: {
    user: {
      name: 'zhangsan',
      age: 18
    },
    components: {
      'user': user
    }
});
```

- `<slot>` 元素可以用一个特殊的特性 `name` 来进一步配置如何分发内容。多个插槽可以有不同的名字。具名插槽将匹配内容片段中有对应 `slot` 特性的元素。

computed & watch

- **computed**: 计算属性将被混入到 **Vue** 实例中。所有 **getter** 和 **setter** 的 **this** 上下文自动地绑定为 **Vue** 实例。**计算属性的结果会被缓存**，除非依赖的响应式属性变化才会重新计算。

```
computed:{  
    aPlus: {  
        get: function () {  
            return this.a + 1  
        },  
        set: function (v) {  
            this.a = v - 1  
        }  
    }  
}
```

- **watch**: 一个对象，键是需要观察的表达式，值是对应回调函数。

```
watch: {  
    a: function(newVal, oldVal){  
    }  
}
```

- **watch**擅长处理的场景：一个数据影响多个数据
- **computed**擅长处理的场景：一个数据受多个数据影响



Thank you

谢

谢

观

看