1、兼容性

Vue.js 不支持 IE8 及其以下版本，因为 Vue.js 使用了 IE8 不能实现的 ECMAScript 5 特性。 Vue.js 支持所有兼容 ECMAScript 5 的浏览器。

1. 构造器

每个 Vue.js 应用都是通过构造函数 Vue 创建一个 Vue 的根实例启动的，可以扩展 Vue构造器

var vm = new Vue({

//实例

})

实例化时，需要传入选项对象，它可以包含数据、模板、挂载元素、方法、生命周期钩子等选项。

可以扩展Vue构造器，从而用预定义的选项创建可复用的组件构造器：

var myComponent = Vue.extend({

//扩展选项

})

// 所有的 `MyComponent` 实例都将以预定义的扩展选项被创建

var myComponentInstance = new MyComponent()

1. 插值

数据绑定最常见的形式就是使用 “Mustache” 语法（双大括号）的文本插值，双大括号会将数据解释为纯文本，而非 HTML。为了输出真正的 HTML ，你需要使用v-html 指令。

Mustache不能在HTML属性中使用,应使用v-bind指令

Mustache对于所有的数据绑定，Vue.js都提供了完全的JavaScript 表达式支持。

1. 过滤器

Vue.js 允许你自定义过滤器，被用作一些常见的文本格式化。过滤器应该被添加在mustache插值的尾部，由“管道符 |”指示，过滤器可以串联。

1. 指令

v-if支持 <template> 语法。

v-show不支持 <template> 语法。

v-else必须紧跟v-if和v-else的后面，否则不能识别

v-for支持一个可选的第二个参数为当前项的索引，例如v-for=”{item.index} in lists”

v-bind用来更新html的属性，例如v-bind:title=”meaasge”，缩写:title=””

v-on用来监听DOM事件，例如v-on:click=”do”，缩写@click=”do”

v-model

v-once指令执行一次性的插值，当数据改变时，该处的值不会更新

v-html双大括号将数据解释为纯文本，为了输出真正的html使用此指令

1. Class和Style绑定

v-bind:class 指令可以与普通的 class 属性共存。

v-bind:style

1. 事件修饰符

在事件处理程序中调用 event.preventDefault() 或 event.stopPropagation() 是非常常见的需求。尽管我们可以在 methods 中轻松实现这点，但更好的方式是：methods 只有纯粹的数据逻辑，而不是去处理 DOM 事件细节。

.stop

.prevent

.capture

.self

例子：

v-on:click.stop=”play”阻止单击事件冒泡

v-on:submit.prevent=”submit”提交事件不在重载页面

v-on:click.stop.prevent=”do”修饰符可以串联

v-on:click.capture事件捕获模式

v-on:click.self只有元素在该元素本身(而不是子元素)时触发

<a v-on:click.once="doThis"></a>点击事件将只会触发一次

1. 按键修饰符

v-on:keyup.enter=””

可以通过全局 config.keyCodes 对象自定义按键修饰符别名

Vue.congig.keyCodes.f1 = 122;

1. 组件

组件实例的作用域是孤立的。这意味着不能并且不应该在子组件的模板内直接引用父组件的数据。可以使用 props 把数据传给子组件。prop 是父组件用来传递数据的一个自定义属性。子组件需要显式地用 props 选项 声明 “prop”

$watch是一个实例方法

vm.$watch(“a”,function(newval,oldval){

//这个回调将在vm.a的值改变之后

})

箭头函数：绑定父上下文

生命周期：实例会调用一些生命周期钩子。

例如：created是在实例被创建之后调用。

还有一些其他的钩子在实例的不同时期被调用。mounted、updated、destroyed。钩子的this指向调用它的Vue实例

一、全局配置：Vue.config是一个对象，包含Vue的全局配置。可以在启动之前修改下列属性：

1、silent，默认值flase。Vue.config.silent=true，取消Vue所有的日志和警告

像在上学一样，有工作时好像在写作业，没工作是好像在上仔细，轻松，但要自律！！！

el,

data,

methods,

filters,

computed

创建一个Vue实例

Var vm = new Vue({

El:’some-element’,

//实例

})

1. **插值**
2. 文本

数据绑定最常见的形式就是使用 “Mustache” 语法（双大括号）的文本插值, 无论何时，绑定的数据对象上 msg 属性发生了改变，插值处的内容都会更新。

<div>{{message}}</div>

通过使用 [v-once 指令](http://cn.vuejs.org/v2/api/#v-once)，你也能执行一次性地插值，当数据改变时，插值处的内容不会更新。但请留心这会影响到该节点上所有的数据绑定

<div v-once>{{message}}</div>

1. 纯HTML

双大括号会将数据解释为纯文本，而非 HTML 。为了输出真正的 HTML ，你需要使用 v-html 指令

<div v-html=”message”></div>

1. 属性

Mustache 不能在 HTML 属性中使用，应使用 [v-bind 指令](http://cn.vuejs.org/v2/api/#v-bind)

<div v-bind:title=”message”></div>

这对布尔值的属性也有效 —— 如果条件被求值为 false 的话该属性会被移除

<button v-bind:disabled=”flag”></button>

1. 使用javascript表达式

对于所有的数据绑定， Vue.js 都提供了完全的 JavaScript 表达式支持。

<div>{{number + 1}}</div>

<div>{{ok?yes:no}}</div>

<div>{{message.split(“”).reverse().join(“”)}}</div>

<div v-bind=” ’last-‘ + id”></div>

这些表达式会在所属 Vue 实例的数据作用域下作为 JavaScript 被解析。有个限制就是，每个绑定都只能包含**单个表达式**

1. **指令**
2. 参数

一些指令只能接收一个参数，在指令后用冒号表明，例如v-bind:href=”url”，这里href是参数，告知v-bind指令将元素的href属性和表达式的url的值绑定。v-bind指令用来响应的更新HTML属性。

v-on指令用来监听DOM事件。这里的参数是监听的事件名。

<button v-on:click=”dosomething”></button>

1. 修饰符

修饰符（Modifiers）是以 . 指明的特殊后缀，用于指出一个指令应该以特殊方式绑定。例如，.prevent 修饰符告诉 v-on 指令对于触发的事件调用 event.preventDefault()

<form v-on:submit.prevent=”submit”></form>

1. **过滤器filter**
2. Vue.js 允许你自定义过滤器，可被用作一些常见的文本格式化。过滤器可以用在两个地方：**mustache 插值和 v-bind 表达式**。过滤器应该被添加在 JavaScript 表达式的尾部，由“管道”符指示。

{{ message | capitalize }}

过滤器设计目的就是用于文本转换。为了在其他指令中实现更复杂的数据变换，你应该使用[计算属性](http://cn.vuejs.org/v2/guide/computed.html)

自定义的过滤器一定要在实例化之前，Vue.filter(‘name’,function(data){})

始终以表达式的值作为第一个参数。带引号的参数视为字符串，不带引号的参数按表达式计算

1. 过滤器可以串联

{{ message | filterA | filterB }}

1. 过滤器是 JavaScript 函数，因此可以接受参数

{{ message | filterA('arg1', arg2) }}

1. 缩写
2. Vue.js 为两个最为常用的指令提供了特别的缩写

v-bind缩写：

<div v-bind:href=”url”></div>

<div :href=”url”></div>

1. v-on缩写

<button v-on:click=”do”><button>

<button @click=”do”><button>

1. **计算属性computed**
2. 计算缓存 VS methods

我们可以通过调用表达式中的 method 来达到同样的效果，可以将同一函数定义为一个 method 而不是一个计算属性。对于最终的结果，两种方式确实是相同的。然而，不同的是**计算属性是基于它们的依赖进行缓存的**。计算属性只有在它的相关依赖发生改变时才会重新求值。这就意味着只要message 还没有发生改变，多次访问 reversedMes 计算属性会立即返回之前的计算结果，而不必再次执行函数。相比而言，只要发生重新渲染，method 调用**总会**执行该函数。

1. Computed属性与watched属性

Vue 确实提供了一种更通用的方式来观察和响应 Vue 实例上的数据变动：watch 属性。当你有一些数据需要随着其它数据变动而变动时，你很容易滥用 watch——特别是如果你之前使用过 AngularJS。然而，通常更好的想法是使用 computed 属性而不是命令式的 watch 回调。

1. 计算setter

计算属性默认只有 getter ，不过在需要时你也可以提供一个 setter

1. 观察watch

虽然计算属性在大多数情况下更合适，但有时也需要一个自定义的 watcher 。这是为什么 Vue 提供一个更通用的方法通过watch 选项，来响应数据的变化。当你想要在数据变化响应时，执行异步操作或开销较大的操作，这是很有用的。

1. **Class与style绑定**
2. 对象语法

我们可以传给 v-bind:class 一个对象，以动态地切换 class 。表示 classactive 的更新将取决于数据属性 isActive 是否为[真值](https://developer.mozilla.org/en-US/docs/Glossary/Truthy) 。

<div v-bind:class=”{active : isActive}”></div>

也可以在对象中传入更多属性用来动态切换多个 class 。此外， v-bind:class 指令可以与普通的 class 属性共存。当 isActive 或者 hasError 变化时，class 列表将相应地更新

<div class=”common” v-bind:class=”{error:haserror, on : isActive}”></div>

也可以直接绑定数据里面的对象，也可以在这里绑定返回对象的[计算属性](http://cn.vuejs.org/v2/guide/computed.html)，根据条件切换列表中的 class ，可以用三元表达式

1. 数组语法

我们可以把一个数组传给 v-bind:class ，以应用一个 class 列表

1. 用在组件上
2. 绑定内联样式v-bind:style

v-bind:style 的对象语法十分直观——看着非常像 CSS ，其实它是一个 JavaScript 对象。 CSS 属性名可以用驼峰式（camelCase）或短横分隔命名（kebab-case），推荐驼峰式。

1. 自动添加前缀

当 v-bind:style 使用需要特定前缀的 CSS 属性时，如 transform ，Vue.js 会自动侦测并添加相应的前缀

1. 条件渲染v-if v-else
2. 条件组

因为 v-if 是一个指令，需要将它添加到一个元素上。但是如果我们想切换多个元素呢？此时我们可以把一个<template> 元素当做包装元素，并在上面使用 v-if。最终的渲染结果不会包含 <template> 元素。

1. v-else

你可以使用 v-else 指令来表示 v-if 的“else 块”， v-else 元素必须紧跟在 v-if 或者 v-else-if 元素的后面——否则它将不会被识别。

1. v-else-if

类似于 v-else,，v-else-if 必须紧跟在 v-if 或者 v-else-if 元素之后。

1. 可以用key管理可复用元素
2. v-show

用于根据条件展示元素的选项,，不同的是带有 v-show 的元素始终会被渲染并保留在 DOM 中。v-show 是简单地切换元素的 CSS 属性 display 。

注意， v-show 不支持 <template> 语法，也不支持 v-else。

1. v-if VS v-show

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；如果在运行时条件不太可能改变，则使用 v-if 较好。

1. 列表渲染v-for
2. 用 v-for 指令根据一组数组的选项列表进行渲染。 v-for 指令需要以 item in items 形式的特殊语法， items是源数据数组并且 item 是数组元素迭代的别名。
3. 对父作用域属性的完全访问权限。 v-for 还支持一个可选的第二个参数为当前项的索引（item,index） in lists
4. 可以用 of 替代 in 作为分隔符，因为它是最接近 JavaScript 迭代器的语法

Item of items

1. 如同 v-if 模板，你也可以用带有 v-for 的 <template> 标签来渲染多个元素块

<template v-for=”item in items”>……</template>

1. 对象迭代
2. 整数迭代
3. 组件和v-for
4. Key

为了给 Vue 一个提示，以便它能跟踪每个节点的身份，从而重用和重新排序现有元素，你需要为每项提供一个唯一 key属性。理想的 key 值是每项都有唯一 id。这个特殊的属性相当于 Vue 1.x 的 track-by ，但它的工作方式类似于一个属性，所以你需要用 v-bind 来绑定动态值（在这里使用简写）

:key=”item.id”

在数据还未加载完时是会有闪烁的情况出现，解决方法也很简单，使用v-cloak,然后定义css：

[v-cloak] { display: none }

1. 事件处理器v-on
2. 表单控件绑定v-model
3. 可以用 v-model 指令在表单控件元素上创建双向数据绑定。它会根据控件类型自动选取正确的方法来更新元素。尽管有些神奇，但 v-model 本质上不过是语法糖，它负责监听用户的输入事件以更新数据，并特别处理一些极端的例子
4. 绑定Vlaue

对于单选按钮，勾选框及选择列表选项， v-model 绑定的 value 通常是静态字符串（对于勾选框是逻辑值）

1. 修饰符
2. 组件
3. 注册

注册一个全局组件, 对于自定义标签名，Vue.js 不强制要求遵循 [W3C规则](https://www.w3.org/TR/custom-elements/#concepts) （小写，并且包含一个短杠），尽管遵循这个规则比较好。

Vue.component(tagNames,options);

例如：

Vue.component(“my-component”,

//options

)

组件在注册之后，便可以在父实例的模块中以自定义元素 <my-component></my-component> 的形式使用。要确保在初始化根实例 **之前** 注册了组件

1. 局部注册

不必在全局注册每个组件。通过使用组件实例选项注册，可以使组件仅在另一个实例/组件的作用域中可用

1. Dom模板解析说明

当使用 DOM 作为模版时（例如，将 el 选项挂载到一个已存在的元素上）, 你会受到 HTML 的一些限制，因为 Vue 只有在浏览器解析和标准化 HTML 后才能获取模版内容。尤其像这些元素 <ul> ， <ol>， <table> ， <select> 限制了能被它包裹的元素， <option> 只能出现在其它元素内部。变通的方案是使用特殊的 is 属性

<table>

<tr is=”my-row”></tr>

</table>

1. 组件中的data必须是函数
2. 组件构成
3. 组件意味着协同工作，通常父子组件会是这样的关系：组件 A 在它的模版中使用了组件 B 。它们之间必然需要相互通信：父组件要给子组件传递数据，子组件需要将它内部发生的事情告知给父组件。然而，在一个良好定义的接口中尽可能将父子组件解耦是很重要的。这保证了每个组件可以在相对隔离的环境中书写和理解，也大幅提高了组件的可维护性和可重用性。

在 Vue.js 中，父子组件的关系可以总结为 **props down, events up** 。父组件通过 **props** 向下传递数据给子组件，子组件通过**events** 给父组件发送消息

1. Prop传递数据
2. 创建组件构造器，注册组件，使用组件

Vue.extend()方法创建组件

Vue.component()方法注册组件

在Vue实例作用范围内使用组件

十二、路由

引入库vue-router.js

存在的问题：

1. vue-resource
2. prop双向绑定
3. prop验证
4. filterBy过滤器