# vue/cli

npm install -g @vue/cli

或者

yarn global add @vue/cli

安装之后，就可以在命令行中访问 vue 命令

验证它是否安装成功。

vue --version

@vue/cli 4.5.4

升级

npm update -g @vue/cli

## 快速原型开发

npm install -g @vue/cli-service-global

安装这个

创建文件test-vue/app.vue或App.vue

<template>

<h1>Hello!</h1>

</template>

进入test-vue

vue serve

使用 vue serve对单个 \*.vue 文件进行快速原型开发Instant Prototyping

也可以显式地指定入口文件：

vue serve MyComponent.vue

vue build

在生产环境模式下零配置构建一个 .js 或 .vue 文件

## 创建一个项目

vue create project1

然后提示有两种选项，vue2或者vue3

Vue CLI 4.5.6

Please pick a preset:

>Default ([Vue 2] babel, eslint)

Default (Vue 3 Preview) ([Vue 3] babel, eslint)

还有一个自己手动创建预设的Manually select features选项

最好用Vue2这个

用vue2创建后main.js

import Vue from 'vue'

import App from './App.vue'

Vue.config.productionTip = false

new Vue({

render: h => h(App),

}).$mount('#app')

用Vue3创建后的mian.js

import { createApp } from 'vue'

import App from './App.vue'

createApp(App).mount('#app')

升级

npm update -g @vue/cli

## 插件

Vue CLI 使用了一套基于插件的架构。

# vue

## 引入

新建index.html

引入

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

## Vue实例

然后新建<script>标签在里边写东西就行了

<body>

<div id="aaa">

{{ message }}

</div>

<script>

var app = new Vue({

el: '#aaa',

data: {

message: 'Hello Vue!'

},

methods:{

log: function(){

return this.message

}

}

})

</script>

</body>

控制台修改app.message="nihao"，页面动态响应

在文档中经常使用 vm (ViewModel 的缩写) 这个变量名命名 Vue 实例。

Vue()声明方法传入的参数叫做选项对象option object

new Vue()创建的是根

Vue 实例还暴露了一些有用的实例 property 与方法，可以引用Vue实例来访问

它们都有前缀 $，以便与用户定义的 property 区分开来。

控制台输入

app.$data

{ message: 'Hello Vue!' }

app.$el

<div id=​"app">​

Hello Vue!

​</div>​

## 生命周期钩子

new Vue({

created: function () {

console.log('a is: ' + this.a)

}

})

this指向Vue实例

## 声明式渲染

<span>Message: {{ msg }}</span>

<span v-once>这个将不会改变: {{ msg }}</span>

使用 v-once 指令执行一次性地插值，当数据改变时，插值处的内容不会更新

<span v-bind:title="message">

鼠标悬停几秒钟查看此处动态绑定的提示信息！

</span>

v-bind被称为指令directive

指令是带有 v- 前缀的特殊 attribute

这里的意思是将这个元素节点的 title attribute 和 Vue 实例的 message保持一致

v-bind 缩写

<a v-bind:href="url">...</a>

<a :href="url">...</a>

v-on 缩写

<a v-on:click="doSomething">...</a>

<a @click="doSomething">...</a>

## 计算属性

<p>Computed reversed message: "{{ reversedMessage }}"</p>

data: {

message: 'Hello'

},

computed: {

reversedMessage: function () {

return this.message.split('').reverse().join('')

}

}

显示Computed reversed message: "olleH"

计算属性的属性是一个getter函数

计算属性是基于它们的响应式依赖进行缓存的。所以reversedMessages存储在缓存中，只要message不变，以后每次访问reversedMessage马上出结果，不执行函数

* 侦听属性Watched Property

data: {

firstName: 'Foo',

lastName: 'Bar',

fullName: 'Foo Bar'

},

watch: {

firstName: function (val) {

this.fullName = val + ' ' + this.lastName

},

}

有一些数据需要随着其它数据变动而变动

只要firstName发生变化，则watch中firstName函数触发

经测试有多个firstName属性只触发最后一个

## 条件与循环

<p v-if="seen">aaaaaaa</p>

也可以用 v-else 添加一个“else 块”：

<h1 v-if="awesome">Vue is awesome!</h1>

<h1 v-else>Oh no 😢</h1>

v-else 元素必须紧跟在带 v-if 或者 v-else-if 的元素的后面，否则它将不会被识别。

<li v-for="todo in todos">

{{ todo.text }}

</li>

<li v-for="(item, index) in items"></li>

可以遍历数组和对象

<div v-for="(value, name) in object"></div>

可以提供第二个参数为property 名称 (也就是键名)

最多三个参数

<div v-for="(value, name, index) in object"></div>

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 attribute：

<div v-for="item in items" v-bind:key="item.id"></div>

请用字符串或数值类型的值作为key

v-for 也可以接受整数。在这种情况下，它会把模板重复对应次数。

<span v-for="n in 10">{{ n }} </span>

## 事件处理

<button v-on:click="reverseMessage">反转消息</button>

## 表单

Vue 还提供了 v-model 指令，它能轻松实现表单输入和应用状态之间的双向绑定。

<input v-model="message">

Vue.component('todo-item', {

template: '<li>这是个待办项</li>'

})

var app = new Vue(...)

## 组件

组件本质上是一个拥有预定义选项的一个 Vue 实例。

组件是可复用的 Vue 实例

* 注册组件

<div id="app">

<item></item>

<item></item>

</div>

Vue.component('item', {

template: '<li>这是个待办项</li>'

})

var app = new Vue({

el: '#app'

})

Vue.component注册组件

之后在Vue实例创建的DOM元素节点中可以创建任意数量的item组件。作为自定义元素使用

此例中渲染的组件都是重复的，渲染相同的文本

<div id="app">

<item v-for="a in todo\_list"

v-bind:todo="a"

v-bind:key="a.id"></item>

</div>

<script>

Vue.component('item', {

template: '<li>{{todo.message}}</li>',

props: ['todo']

})

var app = new Vue({

el: '#app',

data: {

todo\_list: [

{id: 0, message: 'nihao'},

{id: 1, message: '世界'},

]

}

})

</script>

在声明组件时加入props属性，使组件能够接受一个来自父作用域的变量todo

父作用域创建组件时，将组件的todo属性绑定了父作用域的变量a

这样，在父作用域中a改变，组件接收的todo属性也改变，形成数据流

v-for需要提供一个key

<div id="nihao">

<countbutton></countbutton>

</div>

Vue.component('countbutton', {

template:'<button v-on:click="click\_count">{{count}}</button>',

data: function(){

return {

count: 0

}

},

methods: {

click\_count: function(){

this.count++;

}

}

})

var vm = new Vue({

el: '#nihao'

})

组件中有template、data属性

组件一般有template、data、methods属性

组件名countbutton只能全小写或者count-button，不能写成countButton！坑了我一上午

组件的data并不是直接提供一个对象，而必须是一个函数

# BootstrapVue

BootstrapVue官方文档里有很多使用像ml-2, py-1之类的class，这些是帮助控制padding、margin等样式的Bootstrap v4.5 utility classes

全局box-sizing设置为了content-box

## 安装

vue create project1

用>Default ([Vue 2] babel, eslint)创建

进入项目目录

npm install bootstrap-vue

把main.js添加以下

import Vue from 'vue'

import { BootstrapVue, BootstrapVueIcons } from 'bootstrap-vue'

import 'bootstrap/dist/css/bootstrap.css'

import 'bootstrap-vue/dist/bootstrap-vue.css'

Vue.use(BootstrapVue)

Vue.use(BootstrapVueIcons)

至今不知道如何在选择Default (Vue 3 Preview) ([Vue 3] babel, eslint)创建项目后引入bootstrap

## card

<div>

<b-card title="Card title" sub-title="Card subtitle">

<b-card-text>A second paragraph of text in the card.</b-card-text>

<a href="#" class="card-link">Card link</a>

<b-link href="#" class="card-link">Another link</b-link>

</b-card>

</div>

可以添加title和sub-title属性

给a标签或b-link标签添加card-link类，成为卡片中的link

### b-card-body

<div>

<b-card no-body>

<b-card-body>

<b-card-title>Card Title</b-card-title>

</b-card-body>

<b-card-body>

<a href="#" class="card-link">Card link</a>

</b-card-body>

</b-card>

</div>

<b-card>的基本构造块是<b-card-body>，<b-card-body>有padding

默认情况下<b-card>的内容直接放置在<b-card-body>里，<b-card-body>不需要显示声明

激活<b-card>组件的no-body属性，则<b-card>内容不放在<b-card-body>里，padding也不存在