

Vue 笔记

曹思远

2021 年 1 月 15 日

目录

| | |
|--------------------------------------|-----------|
| 1 起手式 | 2 |
| 1.1 Vue 自学路线图 | 2 |
| 1.2 项目搭建 | 2 |
| 1.2.1 两种方法 | 2 |
| 1.2.2 @vue/cli 用法 | 2 |
| 1.2.3 安装 Vue 的选项 | 3 |
| 1.3 Vue 实例 | 3 |
| 1.3.1 如何使用 Vue 实例 | 3 |
| 1.3.2 Vue 实例的作用 | 5 |
| 1.3.3 SEO 基本原理 | 5 |
| 1.4 理解两种 vue 的区别 | 6 |
| 1.4.1 深入理解 | 6 |
| 1.4.2 引用错了会怎样 | 6 |
| 2 构造选项 | 7 |
| 2.1 new Vue 里有什么 | 7 |
| 2.2 options 里有什么 | 7 |
| 2.2.1 options 里的五类属性 | 7 |
| 2.2.2 属性分段 | 8 |
| 2.2.3 入门属性 | 8 |
| 3 数据响应式 | 9 |
| 3.1 Vue 对 data 做了什么 | 9 |
| 3.1.1 小实验 | 9 |
| 3.1.2 小结 | 9 |
| 3.1.3 示意图 | 10 |
| 3.2 深入理解数据响应式 | 10 |
| 3.3 Vue 的 bug | 10 |
| 3.3.1 Vue.set 和 this.\$set | 11 |
| 3.3.2 data 中有数组怎么办? | 11 |
| 3.3.3 ES6 写法 | 11 |
| 3.3.4 ES5 写法-原型 | 12 |
| 3.4 总结 | 12 |
| 3.5 刁钻的题 | 12 |
| 4 computed 和 watch | 12 |

| | |
|-----------------------|----|
| 5 模板，指令与修饰符 | 12 |
| 6 进阶构造属性 | 12 |
| 7 表单与 v-model | 12 |
| 8 Vue Router-前端路由实现思路 | 12 |
| 9 深入理解 Vue 动画原理 | 12 |

1 起手式

1.1 Vue 自学路线图



图 1: Vue 自学路线图

1.2 项目搭建

搞出一个使用 Vue 的项目

1.2.1 两种方法

1. 方法一: 使用 @vue/cli

- 搜索 @vue/cli，进入官网
- 打开文档，打开创建一个项目章节

2. 方法二: 自己从零搭建 Vue 项目

- 使用 webpack 或者 rollup 从零开始
- 适合老手

1.2.2 @vue/cli 用法

- 全局安装: yarn global add @vue/cli
- 创建目录: vue create 路径 (路径可以用. 点)
- 选择使用哪些配置
- 进入目录，运行 yarn serve 开启 webpack-dev-server
- 用 WebStorm 或 VSCode 打开项目开始干
- 进入 @vue/cli 官网看目录

1.2.3 安装 Vue 的选项

1. 选错了请 Ctrl+C 中断然后重来
2. 没有截图的都使用默认选项
3. 真实项目自行斟酌

```
? Please pick a preset:  
  default (babel, eslint)  
> Manually select features
```

```
? Check the features needed for your project:  
  (*) Babel  
  ( ) TypeScript  
  ( ) Progressive Web App (PWA) Support  
  ( ) Router  
  ( ) Vuex  
  (*) CSS Pre-processors  
  (*) Linter / Formatter  
  (*) Unit Testing  
>( ) E2E Testing
```

```
? Pick additional lint features:  
  ( ) Lint on save  
>(*) Lint and fix on commit  
  
? Pick a unit testing solution:  
  Mocha + Chai  
> Jest  
  
? Save this as a preset for future projects? (y/N) n  
  
$ cd hello-world  
$ yarn serve
```

4. [vue demo 地址](#)

1.3 Vue 实例

很重要，做项目必须用到

1.3.1 如何使用 Vue 实例

1. 方法一：从 HTML 得到视图
 - 也就是文档里说的完整版 Vue
 - 从 CDN 引入 vue.js 或 vue.min.js

- 也可以通过 import 引入 vue.js 或者 vue.min.js
- 完整版视图支持从 html 引入
- 也可以用 template 写在 js 里面
- 完整版不好的地方是给用户的体积变大了

2. 方法二: 用 JS 构建视图

- 使用 vue.runtime.js, 也就是非完整版
- 不支持从 html 里获得视图, template 也不行
- 这种方法不方便, 但实际是对的
- 必须要用 render(createElement) 的方式把所有元素构造出来

```
new Vue({
  el: "#app",
  render(createElement) {
    const h = createElement;
    return h('div', [this.n, h('button', {
      on:{click:this.add}, '+1'
    })])
  },
  data: {
    n: 0
  },
  methods: {
    add() {
      this.n += 1
    }
  }
})
```

- 好处是更加的独立, 不需要编译器, 减少百分之 30 的体积

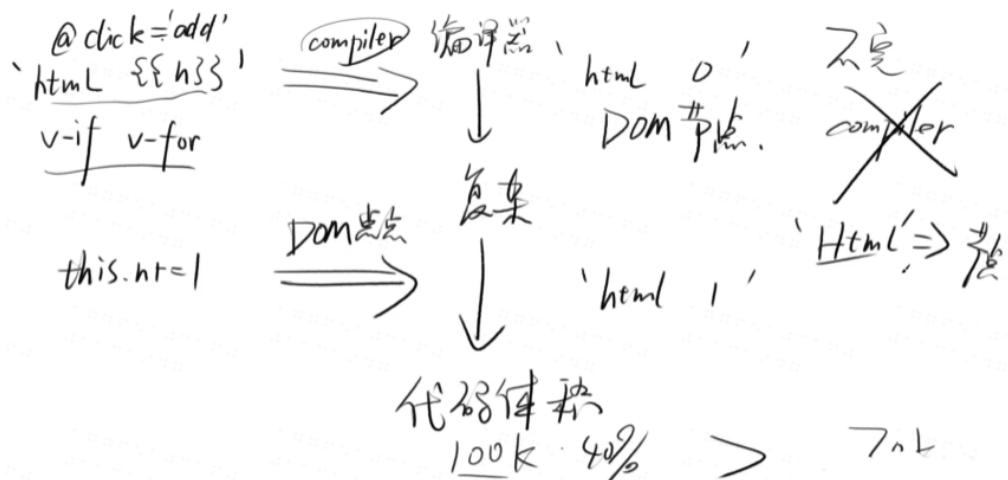


图 2: 完整版与不完整版的区别

3. 方法三: 使用 vue-loader

- 可以把.vue 文件以及其 template 里的东西翻译成 h 构建方法
- 但这样做 HTML 就只有一个 div#app, SEO 不友好
- 通过 webpack 让用户写的时候是完整版, 但实际下载的时候是非完整版

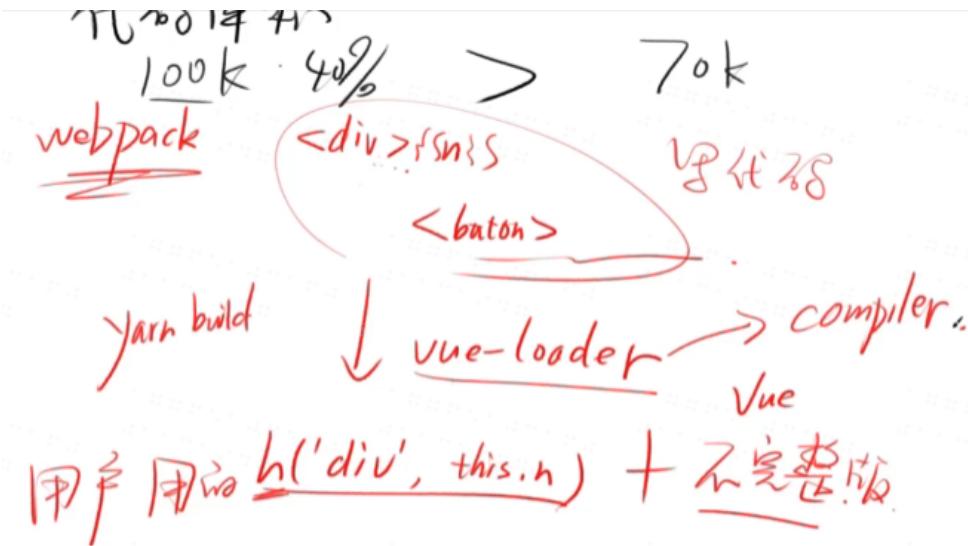


图 3: 用 webpack 通过 vue-loader 转化减少代码体积

1.3.2 Vue 实例的作用

1. Vue 实例就如同 jQuery 实例

- 封装了对 DOM 的所有操作
- 封装了对 data 的所有操作

2. 操作 DOM

- 无非就是监听事件，改变 DOM

3. 操作 data

- 无非就是增删改查
- Vue2 还有一个 bug(面试考，在后面响应式原理里)

4. 没有封装 ajax

- 用 axios 的 ajax 功能

1.3.3 SEO 基本原理

1. seo 友好

- 搜索引擎优化
- 你可以认为搜索引擎就是不停地 curl
- 搜索引擎根据 curl 结果猜测页面内容
- 如果你的页面都是用 JS 创建的 div，那么 curl 就很瞎

2. 那怎么办

- 给 curl 一点内容
- 把 title, description, keyword, h1, a 写好即可
- 原则：让 curl 能得到页面的信息，seo 就能正常工作

1.4 理解两种 vue 的区别

1.4.1 深入理解

| 深入理解两种区别 | | | |
|-------------|-------------------------------|----------------------------|-----------------------|
| | Vue 完整版 | Vue 非完整版 | 评价 |
| 特点 | 有 compiler | 没有 compiler | compiler 占 40% 体积 |
| 视图 | 写在 HTML 里 或者写在 template 选项 | 写在 render 函数里 用 h 来创建标签 | h 是尤雨溪写好传给 render 的 |
| cdn 引入 | vue.js | vue.runtime.js | 文件名不同，生成环境后缀为 .min.js |
| webpack 引入 | 需要配置 alias | 默认使用此版 | 尤雨溪配置的 |
| @vue/cli 引入 | 需要额外配置 | 默认使用此版 | 尤雨溪、蒋豪群配置的 |

最佳实践：总是使用非完整版，然后配合 vue-loader 和 vue 文件
思路：
1. 保证用户体验，用户下载的 JS 文件体积更小，但只支持 h 函数
2. 保证开发体验，开发者可直接在 vue 文件里写 HTML 标签，而不写 h 函数
3. 脏话让 loader 做，vue-loader 把 vue 文件里的 HTML 转为 h 函数
真 TM 聪明，这就是工程师干的事

图 4: 两种 vue 版本的区别

1.4.2 引用错了会怎样

1. vue.js 错用成了 vue.runtime.js
 - 无法将 HTML 编译成视图
2. vue.runtime.js 错用成 vue.js
 - 代码体积变大，因为 vue.js 有编译 HTML 的功能

2 构造选项

2.1 new Vue 里有什么

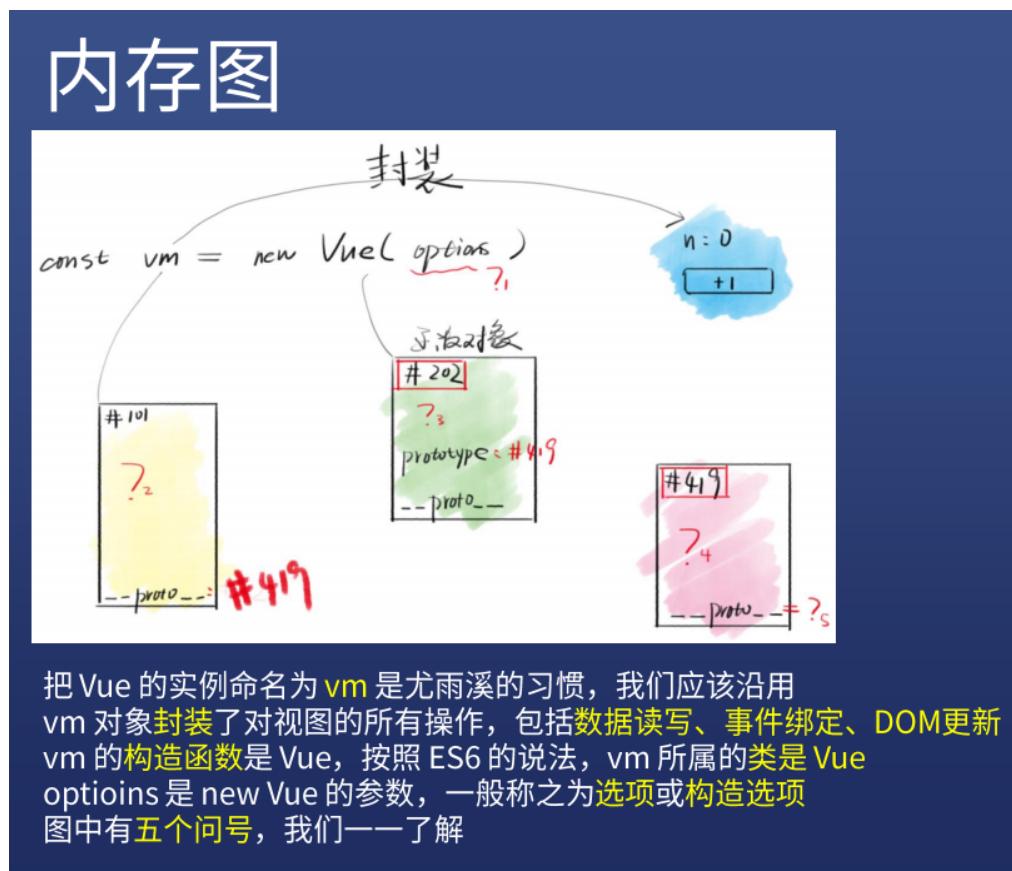


图 5: 实例的内存图

2.2 options 里有什么

2.2.1 options 里的五类属性

1. 数据

- `data, props, propsData, computed, methods, watch`

2. DOM

- `el, template, render, renderError`

3. 生命周期钩子

- `beforeCreate, created, beforeMount, mounted, beforeUpdate, updated`
- `activated, deactivated, beforeDestroy, destroyed, errorCaptured`

4. 资源

- `directives, filters, components`

5. 组合

- `parent, mixins, extends, provide, inject`

2.2.2 属性分段

1. 红色属性 (9)

- 好学，必学，几句话就能说明白

2. 黄色属性 (9)

- 高级属性，稍微费点力

3. 绿色属性 (5)

- 简单

4. 蓝色属性 (2)

- 不常用，可学可不学

5. 紫色属性 (2)

- 比较特殊，重点

6. 黑色属性 (3)

- 很不常用，用的时候看一下文档

2.2.3 入门属性

重点！ 演示代码地址

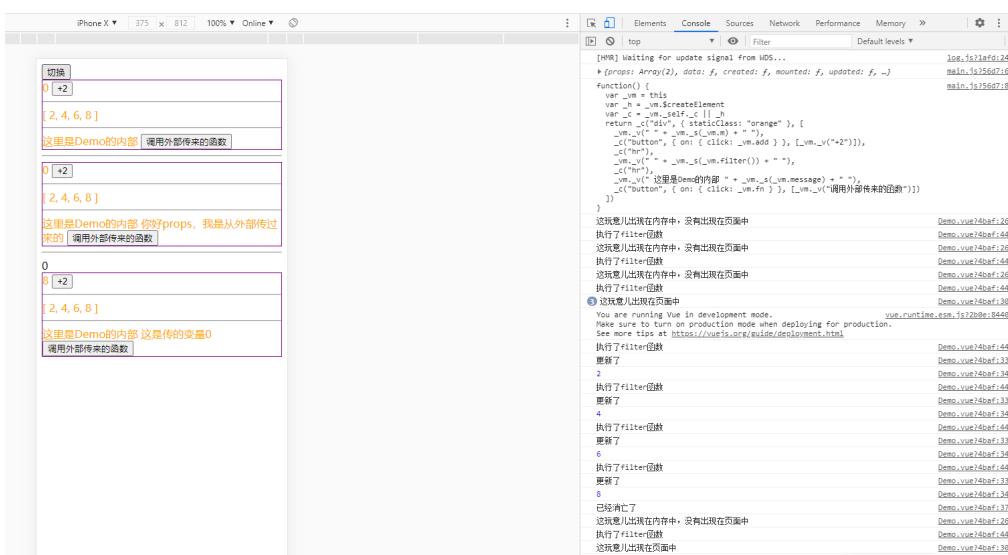


图 6: 代码演示结果

1. el - 挂载点

- 可以用 \$mount 代替

2. data - 内部数据

- 支持对象和函数，优先用函数，如果不这样，组件间就会共用 data

3. methods - 方法

- 事件处理函数或者是普通函数

- 每次执行都会调用，可能是毫无意义的，这个时候就需要 computed

4. components

- Vue 组件，注意大小写
- 三种引入方式，优先使用模块化

5. 四个钩子

- created - 实例出现在内存中
- mounted - 实例出现在页面中
- updated - 实例更新了
- destroyed - 实例从页面和内存中消亡了 (具体看代码演示)

6. props - 外部数据

- 也叫属性
- message='n' 传入字符串
- :message='n' 传入 this.n 数据
- :fn='add' 传入 this.add 函数

3 数据响应式

3.1 Vue 对 data 做了什么

3.1.1 小实验

1. data 变了

- [示例代码](#)
- myData 居然变了
- 一开始是 {n:0}, 传给 new Vue 之后立马变成 {n:(...)}
• {n:(...)} 是个什么玩意，为什么表现和 {n:0} 一致？
- 需要先学习一下 es6 的[getter 和 setter](#)
- [示例代码](#)
- 理解 n:(...) 还需要学习一下[Object.defineProperty](#)
- [示例代码](#)
- 代理模式继续理解
- 注意代码思路，研究方法比知识本身更重要，不读源码也能了解真相

3.1.2 小结

1. Object.defineProperty

- 可以给对象添加属性 value
- 可以给对象添加 getter/setter
- getter/setter 用于对属性的读写进行监控

2. 啥是代理 (设计模式)

- 对 myData 对象的属性读写，全权由另一个对象 vm 负责
- 那么 vm 就是 myData 的代理 (类比房东租房)
- 比如 myData.n 不用，便要用 vm.n 来操作 myData.n

3. `vm=new Vue({data:myData})`

- 一，会让 vm 成为 myData 的代理 (proxy)
- 二，会对 myData 的所有属性进行监控
- 为什么要监控，为了防止 myData 的属性变了，vm 不知道
- vm 知道了又如何？知道属性变了就可以调用 `render(data)` 呀！
- UI=`render(data)`

3.1.3 示意图

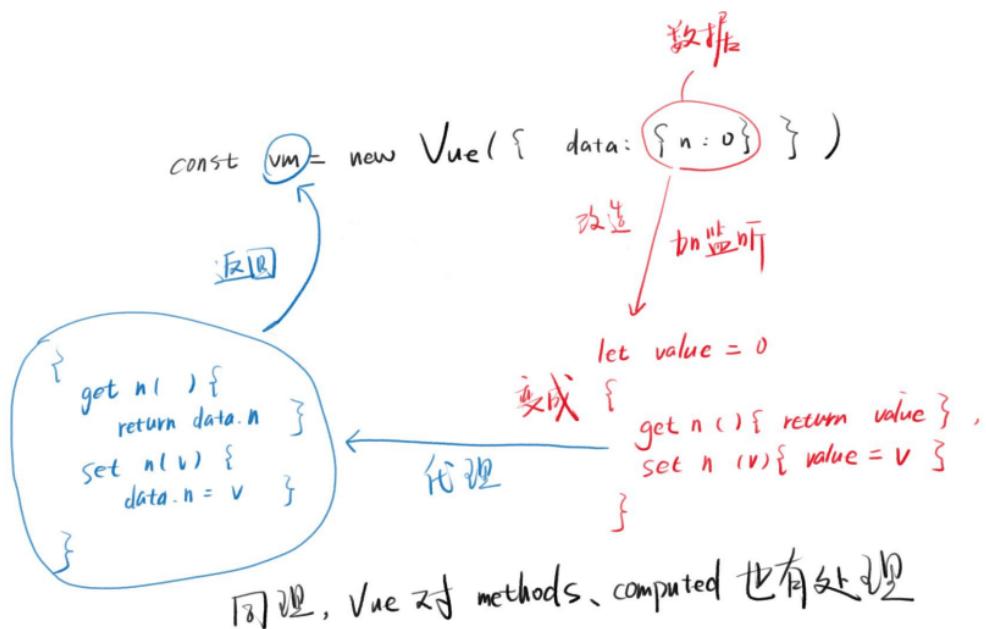


图 7: 如果 data 有多个属性 n,m,k, 那么就会有 get n/get m/get k 等

3.2 深入理解数据响应式

1. 什么是响应式

- 我打你一拳，你会喊疼，那你就是响应式的
- 若一个物体能对外界的刺激做出反应，它就是响应式的

2. Vue 的 data 是响应式

- `const vm = new Vue(data:n:0)`
- 我如果修改 `vm.n`，那么 UI 中的 n 就会响应我
- Vue2 通过 `Object.defineProperty` 来实现数据响应式

3.3 Vue 的 bug

1. `Object.defineProperty` 的问题

- `Object.defineProperty(obj, 'n', {...})`

- 必须要有一个'n'，才能监听 & 代理 obj.n
- 如果前端开发者比较水，没有给 n 怎么办
- 示例一：Vue 会给出一个警告
- 示例二：Vue 只会检查第一层属性
- 此时如果我点击 set b，请问视图中会显示 1 吗？
- 答案是：不会
- 为什么：因为 Vue 没办法监听一开始不存在的 obj.b

2. 解决办法

- 一，那我把 key 都声明好，后面不再加属性不就行了
- 使用 Vue.set 或者 this.\$set

3.3.1 Vue.set 和 this.\$set

1. 作用

- 新增 key
- 自动创建代理和监听 (如果没有创建过)
- 触发 UI 更新 (但并不会立即更新)

2. 举例

- this.\$set(this.object, 'm', 100)

3.3.2 data 中有数组怎么办？

1. 你没法提前声明所有 key

- 示例 1：数组的长度可以一直增加，下标就是 key
- 你看，你没有办法提前把数组的 key 都声明出来
- Vue 也不能检测对你新增了下标
- 难道每次改数组都要用 Vue.set 或者 this.\$set

2. 尤雨溪的做法

- 篡改数组的 API，见文档中变异方法章节
- 这 7 个 API 都会被 Vue 篡改，调用后会更新 UI
- 如何篡改见下面代码

3.3.3 ES6 写法

注：这不代表 Vue 的真实实现，但基本思路是这样，程序员不需要看源码也能实现出来的

```
class VueArray extends Array {
  push(...args) {
    const oldLength = this.length //this 就是当前数组
    super.push(...args)
    console.log('你push了')
    for(let i = oldLength; i < this.length; i++) {
      Vue.set(this, i, this[i])
      //将每个新增的key都告诉Vue
    }
  }
}
```

3.3.4 ES5 写法-原型

这个需要前端基础笔记中原型链的知识

```
const vueArrayPrototype = {
  push: function() {
    console.log('你push了')
    return Array.prototype.push.apply(this, arguments)
  }
}
vueArrayPrototype.__proto__ = Array.prototype
//上面这句话用的不是标准属性，仅学习使用

const array = Object.create(vueArrayPrototype)
array.push(1)
```

3.4 总结

1. 对象中新增的 key

- Vue 没办法事先监听和代理
- 要使用 set 来新增 key，创建监听和代理，更新 UI
- 最好提前把属性都写出来，不要新增 key
- 但数组做不到不新增 key

2. 数组中新增的 key

- 也可用 set 来新增 key，更新 UI
- 不过尤雨溪篡改了 7 个 API 方便你对数组进行增删
- 这 7 个 API 会自动处理监听和代理，并更新 UI
- 结论：数组新增 key 最好通过 7 个 API

3.5 刀钻的题

4 computed 和 watch

5 模板，指令与修饰符

6 进阶构造属性

7 表单与 v-model

8 Vue Router-前端路由实现思路

9 深入理解 Vue 动画原理