第01课：给你一个不学 Vue 的理由

什么是 Vue

Vue 是尤雨溪一个人的项目，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。

Vue 通过 Api 来进行统一性的管理，可以让整个团队的代码都用统一的风格和方法标准去运作，而且对组件系统也有强大的支持，在封装组件时通过 Props 和 Event 两个标准性的原则去调用，可以让开发更加得心应手。

请给一个不学 Vue 的理由

如果非要让我说一个不学 Vue 的理由，可能是它的写法太方便了……你也可能觉得它借鉴的太多，没有亮眼的地方，那我只能说同样实现的东西就是那么方便，简洁的教程和 Api 文档接入整个开发体系，相当符合中国市场的开发——业务变动大、版本要求上线快、需求改动频繁、学习成本低……相比之下，同样有着高效的功能，集成了组件系统和 Virtual DOM。

阅读这一系列课程你可以学到什么？

掌握 Vue 主要就是正确理解教程和深入掌握 Api 的用法，不但要会用，更重要的是学会对症下药，在任何一种场景下使用最简洁、最正确、最合理的代码才是关键。只有对 Api 和教程有了一定程度的项目实战和组件库实战经验才能把它用的游刃有余。

在 Vue2.0 起步的时候我在掘金上进行了 Vue 课程的一系列套课的讲解，从基础到 Vuex，最后到组件库的实战都进行了简单的讲解。期间也经过大量的项目实战和组件库的实践，通过一步步总结，对 Api 文档的深入理解和测试性模拟，总结了一些真实场景的正确用法和常用案例需求，让你在开发中少走弯路，少刨坑。

掘金的个人主页

除了 Vue 还能学到什么

在 Vue 开发中，我们不但要准确的运用 Api，还要结合 es6 的新语法，用更深更强大的新特性来组织代码，这同样也是下一代 Javascript 的标准：
* let 和 const 命令
* 变量的解构赋值
* 字符串的扩展
* 函数的扩展
* 数组的扩展
* 对象的扩展
* Promise 对象
* async 函数

在此次教程中将会展示 es6 大量的新语法进行，只有不断的进行尝试，才能有不同的成效。

本课程的优势

如果你想快速上手进行一个特别面向 C 端的 Mobile 产品开发，甚至是一个中大型的项目开发，如果你能完全阅读完所有课程，并且跟着一步步实践，那么你同样也能给自己的C端产品设计一套属于自己的组件库，毕竟通用型的组件库仍然具备面对市场竞争需求的独特性。

本课程分享的内容是 Vue 的最新版本，可以说这是一套独一无二的教程，不但会结合官方教程和 Api，最主要的是告诉大家在什么场景用什么方式组织代码，避开不必要的坑。

数据驱动架构体系永远离不开组件模式。在这里我会给大家分享级别组件的划分内部原则性，在自己打造组件库的同时，也大量借鉴了各大厂商团队的优秀组件写法，进行比对优缺点，总结相应的理论。

课程大纲如下

- 01 开启 Vue 之旅
- 02 灵活的 data，死板的 props
- 03 `$on`， `$emit`， v-on 三者关系
- 04 .sync 王者回归，v-model 使命将至
- 05 `$attrs`， `$listeners` 深组件通信
- 06 智能组件与木偶组件的正确通信
- 07 你不知道的中央事件通信

本课设计想法

当我在掘金写下第一篇文章的时候，虽然只是很基础的部分，但文章在两天内获得了大量关注，这充分显示出了中国市场的开发者们对 Vue 的渴望程度。与此同时，我也收集到了一些批评意见，对于读者的反馈能及时做出响应才更能体现出一个课程的价值。

不是能写出源码的教程就是对你有帮助，也并不是写的很基础就对你没有帮助。不是每个人都能当大上牛、进入大公司的研发团队，大多数程序员都是面对业务层面的开发。因此如何在市场上有立足之地，能快速接手项目，这才是大部分人应该最需要发力的地方。

学习本课程你需要做些什么?

学习本课程的同学需要对 Html 和 JavaScript 的基础知识有一定了解，理解 es6 基础新特性，了解 npm 和 node 的基本用法。

推荐：

阮一峰老师的es6入门

别浪费时间看别的了，如果你能静下来看完整本书，比任何 es6 其它书籍都好，为什么呢？平民化，就像 Vue 一样，很容易让人理解。

同时在学习本教程的时候，尽量跑一遍 Vue 中文官网结合 Api 你能看懂的示例。

https://cn.vuejs.org/

可能有些 Api 或者教程只有一个简单的解释，还特别官方话，没关系，跟着我一步一步敲遍所有的 Demo。

开启 Vue 之旅

在整理好心情开始旅程之时，我们往往都会带上许多必备工具，同样 Vue 在面向开源之时，周边的身态也向其靠拢。

> vue-devtool

以往 Dom 操作的时候，我们都是通过 dubger 断点来进行错点查找和基础数据驱动，dubger 已经派不上什么用场了，只有通过观察数据的变化，才能准确的定位到错识变化的数据和是否执行了需要的事件。

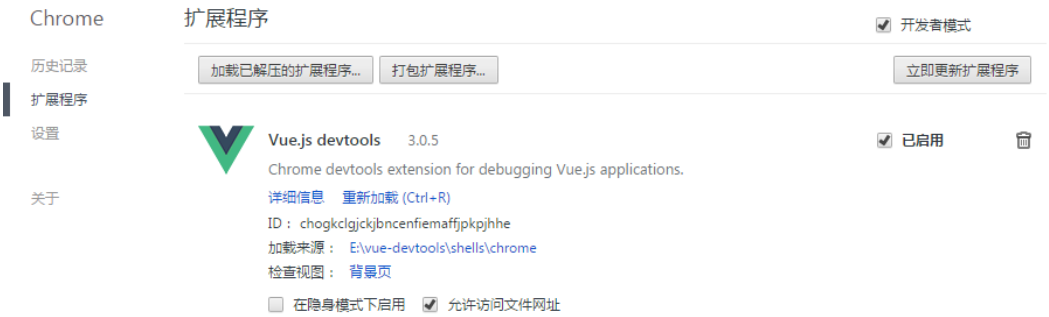
**1.github下载地址
有 git 的同学直接 git clone**

2.下载安成之后打开 cmd 进入 vue-devtools 文件夹把依赖装好 npm install 之后再进行 npm run build

3.然后打开 shells>chrome>src>manifest.json 把里面的”persistent”: false 改为 true

4.一切都万事具备了，打开 Chrome

- 打开里面的设置 > 点击扩展程序 > 点击开发者模式
- 再点击加载已解压的扩展程序，然后把shells>chrome这个文件夹放入就ok了



5.再打开一个用 Vue 写的网页，打开Chrom调试工具你就会发现——大功告成！

> vue-cli

Vue.js 提供一个官方命令行工具，可用于快速搭建大型单页应用。该工具提供开箱即用的构建工具配置，带来现代化的前端开发流程。只需几分钟即可创建并启动一个带热重载、保存时静态检查以及可用于生产环境的构建配置的项目：

进入 Node.js 官网，下载 Node.js 安装包；

为了下载安装包快速一点，走淘宝源进入 cli 终端；

```
运行npm install -g cnpm --registry=https://registry.npm.taobao.org
 全局安装 vue-cli
$ npm install --global vue-cli
 创建一个基于 webpack 模板的新项目
$ vue init webpack my-project
 安装依赖，走你
$ cd my-project
$ npm install
$ npm run dev
```

打开文件夹，本次教程的示例全部通过 Components 文件夹来定义单个组件，进行 SPA 的应用开发，用单 .vue 文件也更加直观，一个文夹可能是一个 Page，也可能是一个 Component；

在开启 Vue 的旅程之时，拿 todo-list 尝试一下它的神奇魔法，通过 Vue 实例和模板进行数据与行为的交互绑定；

实例的每个选项如何与定义的模板值进行一一对应，通过数据驱动、事件绑定，来轻松高效的实现一个 todoList 应用。相比 Juqery 这种操作 Dom 的冷兵器时代，给开发者的感觉是完全变了一种模式，延续着 Html 写法的友好性和适应度，同样还提供了 JSX 语法，Vue 官网说是一个渐进式框加，写法也同样是渐近式，让开发者以不畏惧的心态使用，而且 Vue 的数据驱动模式提供了大量的 Api，每个 Api 无论是实例选项还是实例属性都负责着自己的职责，它们就像五金店的零件一样，只有正确的使用每个 Api 特性并且作用到恰当的地方，Vue 工程代码组织结构和后续的维护才会显得易如反掌。在组件化工程化没到来的时候，业务的实现复杂度并不是最难的，反而令人头疼的是对代码后续的版本迭代、重构、复用等一系列问题，希望通过简单的 todo-list 应用，可以对前端开发革命有新的认识！


```
<template>
   <div>
     <input type="text" v-model.trim="msg"  @keyup.enter="push">
      <ul>
        <li v-for="(item,index) in list" :key="index"
@click="deleteItem(index)">
          {{index}} {{item.name}}
        </li>
      </ul>
   </div>
</template>

<script>
  export default {
    name: 'todo-list',
    data () {
      return {
         msg: "",
         list: []
      }
```

```
    },
    methods: {
      push () {
         this.list.push({name:this.msg})
         this.msg = ""
      },
      deleteItem (index) {
        this.list.splice(index,1)
      }
    }
  }
</script>
```

本章通过这个示例 Demo 表现 Vue 数据驱动式框架运作是如何简单到令人窒息。

一个 todo-list 应用集成了两个事件，两条 data 数据就完成了！

通过 Template 里的 Html 模版能清楚的观察到绑定信息，数据联动和时时改动：
* v-model 里的 msg 和实列 data 里存放的数据进行了绑定
* @keyup.enter=”push” 对键盘事件 keyup 进行监听，同时用 enter 修饰符进行 enter 按键进行监听，当触行 methods 里的 push 函数，向整个 list 列表里添加一条 object 数据
* 通过 v-for 指令循环出整个 list 里的数据，循环出相对应的节点数
* 点击每个节点的时候执行 deleteItem 事件，删除对应的节点

> 对于往时操作dom写法和当前的数据驱动有什么区别？

- 数据渲染，我们会通过第三方的模版引擎，比如 artTemplate，Jade 等等，渲染完毕之后再 append 到根元素中。
- Vue 只是通过一个 v-for 指令循环所有对应的节点，先前只要在 Html 中写好循环模板。

- 执行事件，需要获取 DOM 元素，对 DOM 元素 addEventlistener 事件，再进行函数。
- Vue 直接通过你的 Template 集成的模版在需要发生事件的元素上直接绑定事件，只要执行一个 v-on 结合你需要绑定的事件，所有原生的事件都支持。

- 需要存储数据时，我们需要定一堆变量，有局部变量和全局变量，导致后续的变量难以维护，甚至可能会导致变量名冲突，作用域调用错误，无法准确定位到正确的数据源。
- Vue 通过 data 选项，用每个属性去保存渲染的数据和临时过度的数据，用统一 data 选项去保存，让使用者一目了然。

- 所有执行的函数，无论是事件所需要执行的，还是封装所需要调用的函数，通过函数式声明在 script 标签内写入，代码量大了，也会存在变量名冲突，和无法准确的定位方法。
- Vue 通过 Methods 选项专门为事件所执行的函数和所封装需要调用的函数，就像垃圾桶一样，有一个准确的、可投放的位置，需要找到执行和所需调用的函数，直接可以准确定位到 Methods 选项。

- 平时我们要对有些数据进行一些处理，比方说去除有后空格，按键的定位，都要通过 js 去过滤或者判断。
- Vue 提供了大量的修饰符封装了这些过滤和判断，让开发者少写代码，把时间都投入的业务、逻辑上，只需要通过一个.修饰符去调用。

以上只是一个简单 todo-list Demo 总结出来的例子，文中所提到的也只是部分功能优势，还有很多功能可以让开发路径更加快速。重点在于数据驱动的模式，只要把组件与组件之间的通信掌握了，也就相当于你就手握大半江山，因为一切的一切都是基于组件通信模式和结构用法来的。

> 下篇课程导读：

数据驱动一切都是一数据，只有灵活把控对数据的理解，才能自如的运用，在 Vue 里灵活的 data，死板的 props，是存放数据的和传递数据的基点。