大家好，今天由我来为大家简单介绍下vue的相关概念和使用。

一、引言（6分钟）

在具体介绍vue之前，分享一篇我大半年前看过的文章，至今还觉得说到心坎里去了。文章中作者对vue的评价是优雅而简洁，vue在可读性、可维护性和趣味性之间做到了很好的平衡，广泛借鉴了react和angular的设计理念。我们都知道，angular是Google力推的框架，而react是Facebook力推的框架。Vue一开始只是尤雨溪的个人项目。因此，vue从出现到现在，能在开发框架中大放光彩，并不是由于粉丝的过度追捧，也不是某个大公司的权威推动，它能有今天，其实力毋容置疑。

也许你们会认为这只是一个使用三大框架的作者个人感觉罢了，那我们可以看下vue的热度图，使用数据进行说话。可以看出，vue虽然年轻，但是github上的stars却是高速增长，截止现在已经不断接近react。

题外话： 说下如何学习一个框架：首先看框架的官方文档，因为官方文档更具有权威性，是我们学习的最好教程；看完官方文档我们对整个框架的使用已经有大致的了解；这样的话，我们可以在github上面去阅读并下载别人的项目，研究别人项目源码。通过对项目的学习能更快速入门新框架。

1. 目录介绍（1分钟20秒）

回归正题，这是我今天讲解目录。对vue的介绍，主要分三部分进行介绍：

Vue是轻量级的MVVM框架，因为第一节我们首先了解下什么是MVVM设计模式; 对MVVM设计模式有个大概的印象和概念后，第二节我们主要是讲解组件式开发，在讲解组件式开发过程中，我会带大家用实例进一步来了解MVVM设计模式；而最后一点，我们将会一起了解下工程化的目录结构，并从目录结构来认识项目。在讲解过程中，对于项目中比较重要的目录结构和配置文件，我都会提及到。

1. MVVM设计模式
2. MVVM的含义。（4分钟）

MVVM是 Model-View-ViewModel的缩写。也许大家会比MVVM模式比较陌生，但是相信在座的各位都是计算机专业的，应该都知道“MVC”模式吧。MVVM是由MVC发展而来, 对比MVC，MVVM最大的区别点是通信双向的，采用了双向数据绑定。

MVVM将前端页面划分为三层： Model、View、 ViewModel。

Model：是数据层，它是对于纯数据的处理，如增删改查，不涉及业务逻辑的处理。

View: 是视图模板层。这一层除了定义结构、布局外，她展示的是ViewModel层的数据和状态，View层会进行一系列的声明，如数据绑定的声明、指令的声明、事件绑定的声明。

ViewModel: 它是MVVM中的核心，负责链接View 和 Model，保证视图和数据的一致性，处理一切JS可视业务逻辑，这这一层会进行业务的处理。

因此总结而言： Model对应的是数据部分，可以粗略理解为json，不太正确；

View对应的html部分，而ViewModel是他们两者的桥梁。Model和View并不进行直接的通信，而是借助一个中间层来完成。这也就是为什么使用angualr、react、vue等框架时候，我们都会强调一点：开发者尽可能不去操作DOM。

1. MVVM运作模式（4分钟）

接下来我们纵观前后端的运作图来大致了解下MVVM框架运作模式。

从图上可以看出，前端通过ajax请求，从服务器端获取到数据，形成了model层。在浏览器端，Model和ViewModel之间的交互是双向的，而ViewModel作为View和model之间的桥梁帮助其交互，从而实现了View数据的变化可以同步到Model上，而Model数据的变化也会立即反应到View上，这也就是我们一直说的双向数据绑定了。

1. 组件式开发

对于组件，vue的官方文档提到：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。接下来，让我们一起了解下vue的组件式开发。

1. 认识组件

每一个封装的组件，都应该具备：独立、完整和自由组合的特点。组件式开发，类似于我们小时候玩的积木游戏。每一块积木我们可以把它看成是一个组件，是既独立又统一的。因为独立，所以它可以自由组合，也可以随意替换和删除其中一个组件，并不会影响整体。但是它又统一于整体，比如上面的积木都是六边形的，你不可能拿一个三角形放进去啊。

接下来，以element框架的Rate组件为例；rate组件有区分颜色和不区分颜色这两种类型，但是他们的总体效果是基本一样，因此可以封装为同一个组件。在使用的时候，我们可以将封装好的组件以html标签的形式插入到视图中。和平常的html标签不同的是，它会绑定较多的事件、数据以及标签的声明等，方便我们操作。

也许说到这里，大家心里有疑惑。我不用vue，用JQ甚至原生JS可以封装组件，这样是不是没多大必要使用vue？ 其实组件式开发在前端中一直都存在，只是以angular、vue和react这几个框架为代表，将组件式开发这个概念进一步提升，并且实现起来更为优雅。下面我将以具体的 demo进行vue组件讲解

1. 组件demo --- 列表

右边是我们要实现的效果，从图片可以看出，列表的每一项出了所渲染的数据不一样外，其他都是一摸一样的。这时候我们就可以将这个列表写成一个组件。我们父组件开始看。首先我们使用import语法，将列表项的组件引入，在view层中，以分隔符的形式写入。在getData（）方法中，我们通过ajax获得了数据并存在于变量listData中。在list-pandel标签中，我们以自定义属性props的方式，将数据传递给子组件。

接下来，我们看一下子组件的编写。子组件通过script下的props接收到父组件传递过来。在view中，我们是v-for标签遍历接收到的数据。接收到的数据有多少条，我们就可以遍历出多少项列表。view中的编写类似于我们平常使用的template模板语法，且功能更为强大。我们要想在页面中输出列表的title,可以使用双花括号的形式，它可以接收一个变量或表达式。对于图片image的渲染，图片的路径必须是动态的，这时候我们使用 在src钱加冒号，然后对其赋值动态的数据。其他同理。这时候，一个简单的表单组件便编写完成。

通过上面的例子，我们可以看出。组件可分为两种：子组件和父组件。子组件是负责对具体样式及功能等的封装。注意在编写过程中，千万千万不要在子组件里进行ajax请求，因为子组件的数据应该是从父组件里获取到。否则子组件就失去了独立性和可复用性，失去了迁移性。

JQ对比： 接下来，同样的效果，我会以JQ来实现一遍。大家注意对比下他们两个的实现过程，你会更喜欢哪个？在JQ中更新列表，是通过操作DOM，动态去插入dom来实现的。实现过程中，我们是将编写好结构的html片段，以字符串拼接的形式插入到页面中。在整个编写过程中，我们是通过操作DOM动态插入列表。拼接字符串，我们很容易可以看出，在一堆字符串类型中插入变量，是一个比较容易拼接错的过程。同时由于html和js的高耦合度，通过代码，列表结构并不清晰。此外，这样写很明显复用性低。这时候你可能会有疑问，对于复用性这个问题。同样我也可以将其用JQ封装成组件，但是现实开发过程中，假如这个组件只在项目中使用了三四次，其使用JQ来封装的成本并不低，所以我们在使用JQ开发过程中，对于这类组件一般不会封装。

2、组件demo --- 表单

现在我们开始看第二个组件例子，表单。在划分组件时候，我们要注意一个问题，切勿将组件划分过大。否则就失去了组件的意义。曾经很多人刚入门学习vue时候，因为缺少对vue组件的理解，很容易会犯这样一种错误：一个页面我直接用一个vue文件写完，即使一个页面上会有各种各样的表单。这种做法虽然也能实现效果，但是并不可取。

由于页面上的书库款、单选按钮和普通按钮都是比较基本的组件，在项目中其复用率往往很高。因此，对于这个页面我们可以划分为三个组件：带label的输入框组件、若干个单选按钮的组件、普通按钮组件。这样的话，也许大家会有疑虑，这样页面不就要新建文件了吗？整个项目的体积不就变大了吗？这个大家不用担心，因为等版本发布的时候，项目中所有开发代码都会进行打包压缩在一起，甚至连代码间的空格也会压缩掉。

现在我们主要以“输入框组件”进行讲解。代码很简单，view中，label标签的文字是从父组件中获取，同样input标签的提示文字也是由父组件获取。这样数据都是从父组件中获取，才能实现子组件的可复用性。在输入框中我们绑定了一个change事件监听输入框值得变化，当输入框值变化后，使用$emit()方法触发父组件的“textInputChange”方法，并传递了输入框的key和name值，这对最后表单提交时候，自动拼接接口参数很有用。

看完子组件，我们看下父组件。这一次，我们使用MVVM分层的角度进行阅读代码。在view层中，使用v-for标签将model层中的数据遍历生成三个输入框，在v-input标签中，我们可以看到，子组件的label文字、提示文字、以及key名称都是通过属性props方式传递的。此外，textInputChange前面加一个@符号，对父组件触发的方法进行了监听，同时绑定了绑定了ViewModel中的方法。所以我们再次来看下demo。demo中我们在每个表单输入值后，提交，将会自动拼接出一个json数据。

1. 数据驱动代替dom操作

首先，我们了解下什么事数据驱动： 由数据自动去驱动UI去自动更新UI，UI的改变又同时自动反馈到数据，数据成为主导因素，这样使得在业务逻辑处理只要关心数据，方便而且简单很多。

曾经我大四刚出来实习那会，公司在对比react、vue和angular框架时候，给我们展示过这样一个例子。同样，今天我也就把他们展现给你们，不同的是这一次对比的是vue和JQ。

我们用一个比较极端的例子，开一个定时器，将他的延迟时间设置为最小，0s,观看他们两个的运作效率。页面上最多显示300个小方块。具体的实现实现效果我们可以观看下vue 的 demo。

接下来，我们同时在屏幕左右同时打开vue和jQ demo，比较他们的运作效率如何？比较指标主要是：他们小方块的刷新速度（通过数据变化速度来看）。

通过实例我们可以看出，很明显vue使用数据驱动的方式，代码的运行效率会更高。像这个例子，实现同样的页面效果，数据驱动比DOM操作所消耗的性能会少很多。

1. Vue优势，同时也是MVVM设计模式的优势。

（1）低耦合： 视图（View）可以独立于Model变化和修改，一个ViewModel可以绑定到不同的"View"上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变

（2）可重用性： 一个View Model复用到多个View中，同样的一份数据，用不同的UI去做展示，对于版本迭代频繁的UI改动，只要更换View层就行。

（3）独立开发: 分工是非常明显的，由于View和View Model之间是松散耦合的。一个是处理业务和数据，一个是专门的UI处理。完全有两个人分工来做，一个做UI（xml 和 Activity）一个写ViewModel，效率更高。

（4）可测试: 测试是很方便,不管是UI的单元测试还是业务逻辑的单元测试，都是低耦合的

1. 从目录结构看项目
2. 传统目录与工程化目录的对比。

首先我们来对比一下，传统目录与工程化目录。传统目录结构的划分比较简单，主要是将静态资源文件以文件类型进行划分； 而工程化目录结构 主要是根据功能和所扮演的角色进行划分，通过目录可以看出，现在前端更偏向工程化，能做的事情也更多了。通过截图我们可以看出，目录结构划分为：src: