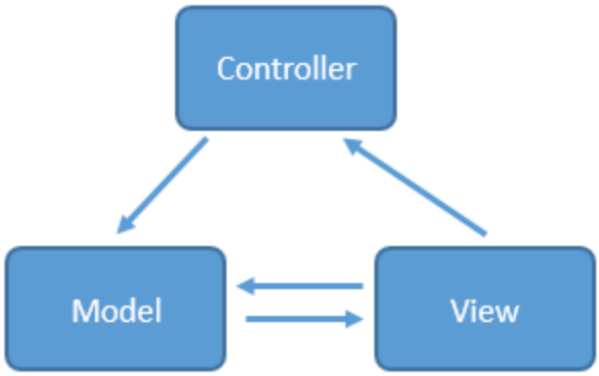
Vue.js 从懵逼到入门

- ## 1、前端发展大事件
  - 浏览器性能弱，标准化程度低
  - 2008谷歌V8引擎诞生，前端的计算能力提升
  - 2009年国际标准化组织ECMA发布了第五代JS规范（ES5），前端的装备得到了整体性的提高
  - 2009年AngularJS诞生。
  - 2011年React和Ember诞生。
  - 2014年Vue.js诞生。
  - 2015年ECMA在发布了第6代JS代码规范（ES6），带来很多令人欣喜的功能特性。
  - ES7-ES8....

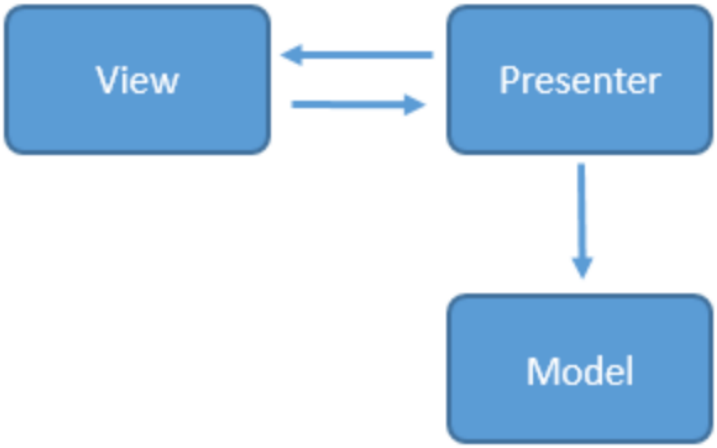
- 2、啥是VUE？
- 是一套构建用户界面（viwe）的MVVM框架

3、MVVM （Model–view–viewmodel）来龙去脉

- MVC
  - M（data model）
    - 业务逻辑、后台逻辑
  - V(view):
    - 视图的架构、输出和展示，用户眼睛看到的内容。
  - C(Controller层)
    - 操作model层的数据，并且返回给view层展示(如：onclick之后的数据变化)

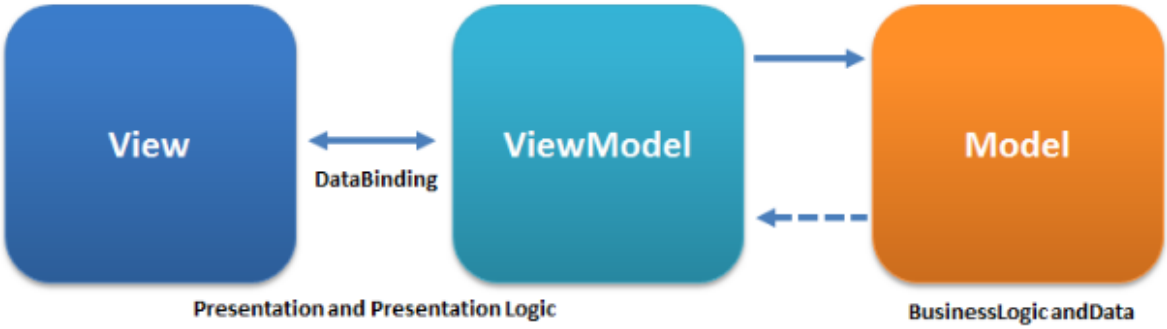


  - 特点
    - view层和model层强耦合
    - Controller层（activity）既属于view也属于model
    - 造成代码混乱冗余、可读性差、不好管理
- MVP

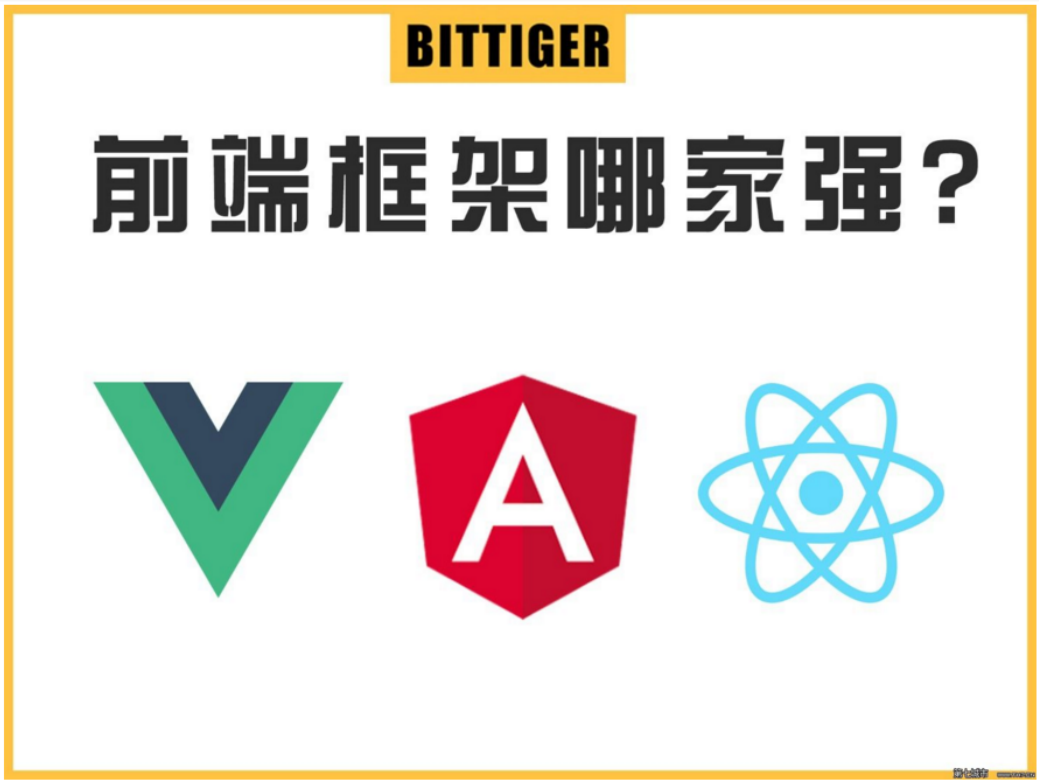


  - 特点：presenter层负责view的行为和传递行为数据给model（偏VIEW）
  - 优势：基本可以做到视图的内容由前端处理，便于管理，比如ajax加载的状态通过前端去显示和处理
  - 缺点：
    - 没有形成统一的规范

        - view的渲染和控制presenter处理，view层和presenter层耦合程度高
    - 现官网模式
- MVVM



    - M（data model）
        - 业务逻辑、后台逻辑
    - V(view):
        - 视图的架构、输出和展示，用户眼睛看到的内容。
    - VM（view model）：value converter
        - 根据后台提供的数据对象，转换、整理成便于管理和更好低为视图层服务的对象
        - 描述和说明后台数据
        - 为后台提供服务，组织在视图层的 use cases(会具体描述用户与系统的交互行为和系统如何响应) 交互数据输出给后端
    - Binder：
        - Binder负责在V和VM之间进行协调和沟通，实现数据绑定
        - 能自动绑定，保持v vm层数据保持规范、一致
- 官网模式对比
    - 页面请求 -> 后端加载FreeMarker模板 -> 解析 -> 使用数据模型来填充该模板 -> 再将最后的HTML页面输出给客户端 -> 浏览器渲染
        - 解析FreeMarker模板中变量的顺序如下:
        - 1,FreeMarker模板内建的变量;
        - 2,ValueStack中的变量;
        - 3,ActionContext中的变量;
        - 4,HttpServletRequest中的属性;
        - 5,HttpSession中的属性;
        - 6,ServletContext范围的属性.
    - 页面请求（异步优势) -> 服务器输出数据模型给客户端 -> 前台显示数据处理 -> 浏览器渲染
- 4、为何出现、解决什么问题
    - 简化数据绑定（NG比较麻烦,控制器-模型-视图）
    - 开发大型单页面应用
    - 代码复用、提高效率、易维护（支持组件化，积木式编程，）
    - 提高可读性高（相较NG）
    - 规范化管理
- 5、优势



- JavaScript框架中最快最小
  - 全面优化过的 React 通常也会慢于开箱即用的 Vue

> 如果你懒得去做，下面的数值是来自于一个 2014 年产的 MacBook Air 并在 Chrome 52 版本下运行所产生的。为了避免偶然性，每个参照项目都分别运行 20 次并取自最好的结果：

	Vue	React
Fastest	23ms	63ms
Median	42ms	81ms
Average	51ms	94ms
95th Perc.	73ms	164ms
Slowest	343ms	453ms

- 学习曲线平缓，渐进式
  - React 学习曲线陡峭，在你开始学 React 前，你需要知道 JSX 和 ES6、需要学习构建系统
  - Vue的应用模式简单到只有一条代码<script src="https://unpkg.com/vue/dist/vue.js"></script>
- 可读性强
  - vue：templates React:JSX

```
render () {
    let { items } = this.props
    let children
    if ( items.length > 0 ) {
        children = (
            <ul>
            {items.map( item =>
              <li key={item.id}>{item.name}</li>
            )}
            </ul>
        )
    } else {
        children = <p>No items found.</p>
    }
    return (
        <div className = 'list-container'>
            {children}
        </div>
    )
}
```

```
<template>
    <div class="list-container">
        <ul v-if="items.length">
        <li v-for="item in items">
            {{ item.name }}
        </li>
        </ul>
        <p v-else>No items found.</p>
    </div>
</template>
```

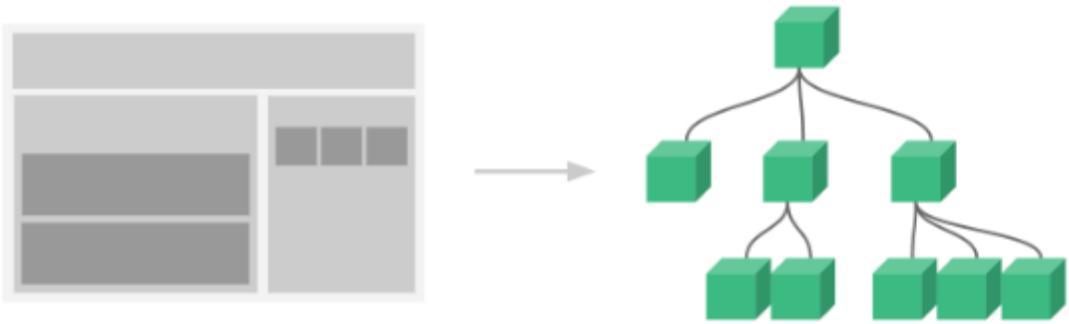
- 容易与已有库项目整合
- Vue-cli 脚手架
  - 能让你非常容易地构建项目(webpack、安装插件)

6、特性

- 双向数据绑定
  - MVVM,视图模型（VM）中数据发生改变，用户视图(V)跟着变，用户视图(V)改变时，视图模型（VM）数据同时发发生改变
  - 底层Object.defineProperty对象
    - vue.js主要利用了accessor descriptors的set和get来更新视图
    - 举个栗子
      - defineProperty绑定实例

```
> var bValue = 38;
  Object.defineProperty(o, 'b', {
    get: function() { return bValue; },
    set: function(newValue) { bValue = newValue; },
    enumerable: true,
    configurable: true
  });
  o.b; // 38
< 38
> bValue = 333
< 333
> o.b
< 333
```

- 组件化
  - 组件系统是 Vue 的另一个重要概念，因为它是一种抽象，允许我们使用小型、独立和通常可复用的组件构建大型应用。仔细想想，几乎任意类型的应用界面都可以抽象为一个组件树



- 虚拟DOM、提升页面刷新速度
  - JS已使用V8引擎，速度之快与JAVA是一个量级，浏览器性能的耗费主要用在DOM的操作上（重绘、重排）
  - 真实的DOM渲染为何慢？
    - DOM本身是个非常复杂的对象，慢！
    - JS访问和处理DOM，慢！
      - DOM对象是给JS操作HTML提供API，两个相互独立的功能只要通过接口彼此连接

    - 渲染页面，最慢！
      - DOM渲染到页面，调用浏览器渲染引擎，浏览器调用layout算法和调用 CSS 重算算法，然后输出（重绘、重新、组合）。
  - 什么是虚拟DOM(VDOM)

```
var element = {
  tagName: 'ul', // 节点标签名
  props: { // DOM的属性，用一个对象存储键值对
    id: 'list'
  },
  children: [ // 该节点的子节点
    {tagName: 'li', props: {class: 'item'}, children: ["Item 1"]},
    {tagName: 'li', props: {class: 'item'}, children: ["Item 2"]},
    {tagName: 'li', props: {class: 'item'}, children: ["Item 3"]},
  ]
}
```

```
<ul id='list'>
  <li class='item'>Item 1</li>
  <li class='item'>Item 2</li>
  <li class='item'>Item 3</li>
</ul>
```

- 虚拟DOM原理
  - 1、用 JavaScript 对象结构模拟 DOM 树的结构，然后用这个树构建一个真实 DOM 树
  - 2、当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异
  - 3、步骤2记录的差异应用到步骤1所构建的真实DOM树上，视图就更新了

- VDOM为何快？--最小化实现DOM渲染
  - 比如：表格的更新，VDOM将直接更新单中文本内容，杜绝对表格重新渲染的可能性
  - 使用createDocumentFragment方法（创建文档碎片节点）创建dom片段，一次性处理
    - 10个元素的循环

```
for (var i=5; i<15; i++) {
var li = document.createElement("li");
li.innerHTML = arr[i];

ul.appendChild(li);
}
```

```
docfrag = document.createDocumentFragment();
for (var i=5; i<15; i++){
    let li = document.createElement("li");
    li.textContent = i;
    docfrag.appendChild(li);
};
ul.appendChild(docfrag);
```

- 便于扩展的插件系统
  - vux vue-router vue-resource axios

7、前后端分离实例

- 旧代码说明、结构分析
- 旧代码特点
- 工作模式
- 新代码说明、结构分析

    - 新代码特点
    - 工作模式
- 8、数据绑定实例
    - 官网列表筛选
- 9、介绍SCM架构
    - nodejs + webpack + vue + vue-router + vue-resource + vuex + elementUI+ 众多依赖
    - nodejs:搭建前端运行环境
    - webpack:编译、打包工具，运行环境热更新，生成环境实现按需动态打包
    - vuex:将数据模型完全和模板分开，负责状态管理，生成全局可复用的数据和数据处理操作
    - vue-resource: ajax
    - vue-router:路由管理，实现单页面应用
    - elementUI：使用第三方组件UI，使用这些组件我们只负责数据模型和逻辑
    - vue-devtools调试
        - localhost:8088/Logon.do
        - admin abcd.1234
- 10、SCM枚举值管理功能实例