什么是Vue:

Vue是基于MVVM设计模式的渐进式的前端js框架

渐进式: 可以有选择的逐步使用框架中的组件

"全家桶": 必须全盘使用所有组件

前端js框架: 不需要nodejs，仅靠浏览器就可独立运行

为什么后续需要nodejs？

将Vue框架中浏览器不认识的新技术翻译为浏览器认识的ES5的对等标准——已经封装好了

何时: 侧重于以数据操作为主的前端项目开发

使用

下载：cn.vuejs.org 最新版本：2.6

开发版：未压缩，包含完备注释和错误提示信息

优点：可读性好

缺点：体积大，不便于传输

生产版：代码经过压缩，删除了备注和错误提示信息

优点：体积最小化，便于传输

缺点：不偏于学习和阅读

使用vue方式

下载独立的vue.js，在网页中引用

使用脚手架代码

MVVM设计模式

旧的前端代码划分:

html: 定义网页的内容

css: 定义网页的样式

js: 增删改查，事件绑定

MVVM设计模式将前端内容重新划分: （步骤）

1. 界面(View): html+css

为HTML添加了动态功能: 变量，if else, for

2. 模型数据(Model): 所有页面上需要的/可能发生变化的数据。集中定义在data={ 数据1:值1, 数据2:值2, ... }

往往模型数据都是ajax从服务端请求来的。

3. 控制器(ViewModel)：将视图View和模型数据(Model)绑定在一起。

绑定: 监控视图和模型，始终保持模型数据与页面自动同步

控制器ViewModel中包含两大子系统:

1. 响应系统: 监控模型中每个变量的变化

只要有变量发生变化，立刻发出通知！

本质: 将data中每个属性都提升为new Vue对象的访问器属性。只要修改new Vue对象的访问器属性，就可修改data中的变量。但是，同时会发出通知！

2. 虚拟DOM树:

什么是: Vue临时生成的仅保存可能变化的元素和属性的DOM树

何时生成: new Vue()边扫描受监控的页面元素，边创建虚拟DOM树，仅保留可能发生变化的元素

何时使用: 响应系统通知某个变量被改变时，告知虚拟DOM树。虚拟DOM树快速遍历自己，找到受影响的DOM元素，仅修改受影响的DOM元素

总结: 虚拟DOM树:

1. 内容少，遍历极快

2. 仅修改受影响的DOM元素的属性或内容

4. 绑定语法:

{{变量名}} Interpolation 插值

告知Vue框架，这里需要哪个变量

{{}}中还可写一切正确的有返回值的js表达式:

比如: 运算, 三目, 函数调用，访问数组元素，访问对象属性，模板字符串

问题: 只能绑定内容，不能绑定属性和事件等...

5. 指令directive

什么是: Vue新增的增强HTML功能的特殊属性

为什么: HTML缺少动态原因所需的要素: 运算, 分支，循环

包括:

1. 专门绑定属性值的指令: v-bind

何时: 绑定属性值

如何: <ANY v-bind:属性名="js表达式">

v-bind: 让普通的属性的""中也可以执行js程序

其实可省略: v-bind

所以: 今后只要绑定属性都用 :属性名="js表达式"

2. 事件绑定: v-on

如何:

HTML中: <ANY v-on:事件名="处理函数名">

<script>中

new Vue({

el:"#app",

data:{ 模型变量 },

methods:{

处理函数 (){

this.模型变量

}

}

})

其实: v-on: 可简写为@，

所以，今后绑定事件都用@事件名="处理函数"

3. v-if: 为HTML添加分支结构

v-if 控制一个元素的显示隐藏

<ANY v-if="条件表达式"

v-if v-else 可控制两个元素二选一显示

<ANY v-if="条件表达式"

<ANY v-else

v-if v-else-if v-else 可控制多个元素多选一显示

<ANY v-if="条件表达式"

<ANY v-else-if="条件表达式"

<ANY v-else

4. v-show: 控制一个元素显示隐藏

用法同v-if: <ANY v-show="条件表达式"

本质: 用display:none隐藏元素

v-if的本质: 用删除元素节点的方式，隐藏元素

5. v-for: 根据数组反复生成多个相同结构的元素

用法: <ANY v-for="elem of 数组">

//elem可当做模型变量使用，用作绑定

</ANY>

of会依次读取数组中每个元素

每读取一个元素就创建一个<ANY>

6. 绑定html片段:

问题: {{}}无法绑定HTML片段，它会努力保持HTML片段原样显示

解决: v-text="变量" 不行，v-text等效于{{}}

v-html="变量" 可以，会被浏览器解析为正文

总结: 只要绑定的内容是HTML片段，都用v-html

7. v-cloak: 在vue加载完之前，短暂隐藏{{}}语法，不让用户看到

如何:

1. 在界面中想隐藏的元素/父元素上添加v-cloak自定义属性

2. 在页面的<style>中用属性选择器找到所有有v-cloak属性的元素，设置隐藏:

<style>[v-cloak]{ display:none }</style>

结果: 当vue对象加载完成，会自动查找所有v-cloak属性，自动移除。所有元素就重新显示出来

8. v-pre: 保护内容中碰巧出现的{{}}不被VUE编译，保持原样显示

9. v-once: 控制一个变量只在开始时绑定一次，之后即使模型变量变化，也不再更新.

6. 双向绑定:

单向绑定: 将内存中模型数据中的值绑定到界面上(Model->View)

问题: 无法绑定表单元素的值

表单元素值的修改，无法自动映射到内存中

解决: 今后只要绑定表单元素，都用双向绑定

什么是双向绑定: 即能将Model中的数据绑定到界面上（Model->View），又能自动将界面上的修改，映射到内存中(View->Model)

如何: <表单元素 v-model:value="模型变量">

简写: 1. <input type="text"

<textarea>

<select>

都可简写为: v-model="模型变量"

省略:value，默认绑定value属性

2. <input type="checkbox">

可简写为: v-model="模型变量"

但默认绑定的是checked

3. <input type="radio">

可简写为: v-model="模型变量"

但用模型变量和radio的value做比较

如果模型变量的值=radio的value时，就选中

如果模型变量的值!=radio的value时，就不选中

绑定class和style

绑定style:

1. 其实style也是一个普通的字符串属性，也可以:

<ANY :style="模型变量"

模型变量: "css属性:值; css属性:值; ..."

问题: 字符串拼接很繁琐，不直观

2. 将style看做一个对象绑定:

<ANY :style="对象名"

data:{

对象:{

css属性:值,

css属性:值

}

}//css属性都要去横线变驼峰！

绑定class

1. 其实class属性也只是一个字符串属性，也可以:

<ANY :class="拼接字符串"

2. 都用对象方式绑定:

<ANY :class="对象名"

//"success"

data:{

对象名:{

hide: false,

success: true,

fail:false

}

}

3. 计算属性：

什么是: 自己不保存值，每次访问都要根据其他属性计算出来

何时: 要显示的值，需要根据其他属性动态计算才能获得时

如何:

1. 定义计算属性:

computed:{

计算属性名(){

根据其他属性，返回计算结果

return ... ...

}

}

2. 绑定时: 和普通属性完全一样，不要加()

vs methods:

methods: 调用时，必须加()，调用几次，就重复计算几次

computed: 绑定时，不加()，即使多次绑定，也只计算一次。vue会缓存计算属性的计算结果

总结: 如果更关心值/结果，优先选计算属性

如果更关心操作/执行过程，优先选方法

自定义指令:

如果13种指令不够用，可自定义指令:

如何:

1. 定义指令:

Vue.directive(

"指令名",//不写v-前缀

{

inserted(elem){//当拥有该指令的元素被添加到DOM树后自动执行

//elem: 接住当前标有该指令的DOM元素对象

对elem执行各种想要的DOM操作

}

}

)

2. 使用指令:

<ANY v-指令名>

当该元素被加载到DOM树上后，自动执行指令中的inserted

过滤器:

什么是: 对从模型中绑定来的原始值，进行再加工后，再显示

何时: 如果模型中的原始值不能直接使用

如何:

1. 定义过滤器:

Vue.filter("过滤器名", function(value,形参1,...){

//value会接住要处理的原始值

return 改造后的新值

})

2. 使用过滤器:

在绑定时:

{{原始模型变量|过滤器(实参1,...)|下一个过滤器 }}

:属性="原始模型变量|过滤器(实参1,...)|下一个过滤器"

axios:

什么是: 基于Promise的专门发送http请求的函数库

何时: 今后在vue中发送ajax请求首选axios

为什么:

1. 手写ajax四步/封装自定义函数

2. jQuery: $.ajax() —— Vue中没有jQuery

3. Vue官方曾经提供了专门发送ajax请求的组件：vue-resource——官方不推荐使用

4. Vue官方推荐axios作为Vue中标准的发送ajax请求的组件

如何:

1. 下载并引入:axios.min.js

内存中，就多出一个axios对象，有两个方法: get() post()

2. 发送get请求:

axios.get("url",{

params:{

uname: "dingding",

upwd:"123456"

}

}).then(function(res){

//res是经过axios再封装的响应对象，不仅是result

//result： res.data才是从前的result

})

发送post请求:

axios.post("url","uname=dingding&upwd=123456")

组件:

什么是: 拥有专属的HTML，CSS，js和数据的独立页面区域

为什么: 重用！

何时: 只要一个页面功能，需要反复使用时

如何:

1. 定义组件:

Vue.component(

"组件名"，//不要用驼峰命名，而用"单词-单词"{

template:"#tplID", //找到要反复使用的HTML片段

data:function(){//每次重复创建组件时，都自动调用data函数，为本次组件创建一个专属的模型对象

return { 模型数据 }//类似于以前的data

},

methods:{ 方法/事件处理函数 },

computed:{ 计算属性 },

watch:{ 监听函数 }

}

)

HTML片段都要保存在<template>元素中并其一个唯一的id

2. 使用组件:

其实vue中的一个组件，就是一个可重复使用的自定义标签而已。组件名其实就是一个标签名:

<组件名></组件名>

会被template的HTML片段代替！