**Vue.js-Day01**

# 课程介绍

* **1. Vue介绍(了解)**
* **2. Vue项目环境搭建（掌握）**
* **3. Vue组件化（掌握）**
* **4. Vue表达式（掌握）**
* **5. Vue常用指令（掌握）**
* **6. 计算属性computed（了解）**

# 1. Vue介绍（了解）

## 1.1 Vue是什么（了解）

Vue是一个构建用户界面（UI）的JS库。

是一个构建**数据驱动**的渐进式框架，Vue.js 的目标是通过尽可能**简单**的 API 实现响应的数据绑定和组合的视图**组件**。基于**MVVM (Model View VievModel)**设计模式书写。

Vue只关注视图层，可以快速创建用户界面。

Vue的目标是通过尽可能简单的 API 实现响应的数据绑定和组合的视图组件。

Vue学习起来非常简单。

## 1.2 Vue特点（了解）

* **体积小**

gzip压缩后20K，并且不依赖其他基础库。

* **更高的运行效率**

基于虚拟DOM，一种可以预先通过JS（在内存中）进行各种运算，把最终的DOM操作计算出来并优化的技术啊。由于这种对DOM操作的预处理操作，并没有真是操作DOM，所以叫做虚拟DOM；

* **双向数据绑定**

让开发者不用再去操作DOM对象，把更多的精力投入到业务逻辑上；

* **生态丰富、学习成本低**

市场上拥有大量成熟、稳定的基于vue.js的UI框架、常用组件！拿来即用实现快速开发；

## 1.3 Vue的历史（了解）

Vue是比较新的技术

2013年发布 草案 版本

2015年10月发布的1.0 版本

**2016年4月，发布2.0版本 (大红大紫的开始)**

**2021年 发布3.0版本**

Vue作者是 前Google 的员工尤雨溪，在2016年9月宣布以技术顾问的身份加盟阿里巴巴Weex团队。实现Vue 和 Weex 的 JavaScript 整合，目标是让Vue 的语法跨三端。

## 1.4 Vue与常见框架对比（了解）

Vue是建立于 Angular和React的基础之上，它保留了Angular和React的优点并添加了自己的独特成分，让框架的使用变得更加简单，这保证了Vue足够的美好来吸引JS开发人员的胃口。

JQ ： 兼容IE6 7，便捷的DOM操作库。

AngularJS（1.X）： 基于脏检查的数据双向绑定机制，API量大。针对大型的复杂应用。

Angular（2+) 与AngularJS没任何关系。深度兼容TS，学习曲线非常陡峭。针对大型的复杂应用。

React： 庞大的全球生态系统，语法紧跟ES官方，运行稳健，适合中大型项目。

Vue： 简单快速，项目运行快, 支持双向数据绑定，适合中型或小型项目。

# Vue项目环境搭建

Vue是一个 JavaScript 框架。它是一个以 JavaScript 编写的库。

Vue是以一个 JavaScript 文件形式发布的，可通过 script 标签添加到网页中：

## 2.1 CDN 引入(了解)

**<script src="https://cdn.bootcss.com/vue/2.3.3/vue.js"></script>**

CDN:全称是Content Delivery Network，即内容分发网络。CDN的通俗理解就是**网站加速**，可以让客户端**快速度访问资源。**

## 2.2 使用vue官方脚手架(vue-cli)【掌握】

### 2.2.1 vue-cli简介（了解）

Vue CLI 是一个基于 Vue.js 进行快速开发的完整系统，是一个官方脚手架，可以帮助我们快速创建vue项目工程目录，目前最新版本4.x。

Vue CLI 致力于将 Vue 生态中的工具基础标准化。它确保了各种构建工具能够基于智能的默认配置即可平稳衔接，这样你可以专注在撰写应用上，而不必花好几天去纠结配置的问题。与此同时，它也为每个工具提供了调整配置的灵活性。

### 2.2.2 vue-cli安装&使用步骤【掌握】

1、全局安装vue-cli

>> **yarn global add @vue/cli**  // vue新脚手架是4.X

2、创建项目

>> **vue create 项目名**

3、进入项目目录，启动项目

>> **yarn serve**

### 2.2.3 项目目录详解（了解）

**一级目录:**

node\_modules //>> 依赖的第三方模块

public //>> vue服务器静态文件目录，只有唯一的一个index.html

**src** //>> 我们的开发目录，最重要的目录，源文件（我们写的代码）目录

.gitignore //>> git忽略列表

babel.config.js //>> es6编译配置

package-lock.json //>> 包描述文件（记录更详细，记住当时的版本信息）

package.json //>> 包描述文件

README.md //>> 说明文档

**二级目录:**

src：

assets //>> 静态资源 styles images fonts

components //>> 组件,是.vue的文件,主要是公用的小组件

views //>> 页面级别的组件

App.vue //>> 整个应用的顶级组件.

main.js //>> 入口文件

备注：

需要安装**vetur**插件，支持vue代码高亮显示。

安装 Vue VSCode Snippets，写vue起飞！！！

### 2.2.4 扩展（了解）

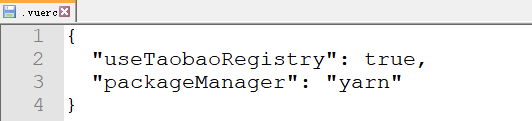
脚手架常见问题：

脚手架模板安装源，默认？淘宝镜像？

脚手架的包管理模式。npm？Yarn？

还原到最初：

1. 找到c:/Users/你的用户/.vuerc
2. 删除文件即可重新选择

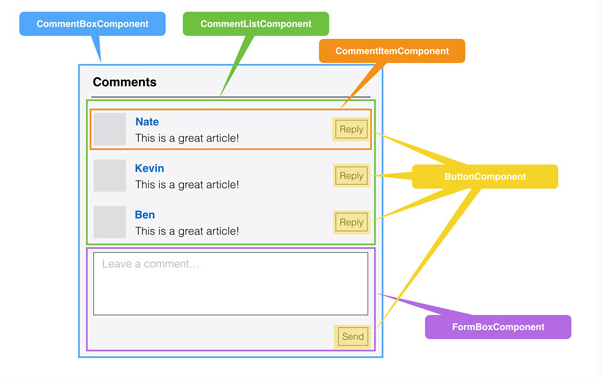


# 组件【掌握】

## 3.1 什么是组件（了解）

>> 所谓组件，**即封装起来的具有独立功能的UI部件**。

开发者从功能的角度出发，将UI分成不同的组件，每个组件都独立封装。



整个UI是一个通过小组件构成的大组件，每个组件只关心自己部分的逻辑，彼此独立。

>> 组件特点：

**可重用**

**可组合**

**易维护**

## 3.2 单文件组件【掌握】

>> 一个 **XXX.vue**文件就是一个单文件组件，功能逻辑独立，由 template、scripts、style三部分组成。

|  |
| --- |
| // XXX.vue  <template>    <div>      <!-- HTML: 这里写HTML结构，注意有且仅有一个根节点-->    </div>  </template>  <script>  // 这里写JS代码，默认需要暴露出去一个实例对象  export default {};  </script>  <style>  /\* 这里写css样式 \*/  </style> |

注意：

>> style标签上可以通过 lang =”less”选择预处理语言（有些需要安装对应的loader模块）。

>> style标签上可以通过 scoped,让样式私有化。

## 3.3 使用自定义组件【掌握】

>> 使用自定义组件分为三个步骤：

1. 引入组件 （引入一个已经写好的xxx.vue组件）
2. 注册组件 （在components里面注册）
3. 使用组件 （通过标签名使用）

|  |
| --- |
| <template>    <div id="app">      <!-- 3. 使用组件(也可以通过单标签使用) -->      <Hello></Hello>    </div>  </template>  <script>  // 1. 引入组件  import Hello from "./components/Hello.vue";  export default {    // 2. 注册组件    components: {      Hello // 等同于： "Hello": Hello    }  };  </script>  <style lang="less">  </style> |

# 4. Mustach表达式和配置选项【掌握】

## 4.1 data配置选项 && mustach模板语法 【掌握】

>> 目前template模板中的数据是写死的，如何渲染动态的数据呢？这个时候就需要data配置选项和mustach表达式派上用场了。

|  |
| --- |
| <template>    <div>      <!-- mustach胡须表达式，用于把data中的数据，输出到html模板中 -->      <h1>Hello {{ userName }} !</h1>    </div>  </template>  <script>  export default {    // data选项用于准备数据，data是一个“函数”， 返回一个对象，对象中再写各种数据    data() {      return {        userName: "李寻欢"      };    }  };  </script>  <style lang="scss" scoped>  </style> |

## 4.2 Mustach表达式和配置选项【掌握】

### 4.2.1 Mustach模板表达式【掌握】

**语法: {{ 表达式 }}**

注意： 表达式必须能输出唯一结果，不能写if else条件等其他JavaScript代码。

### 4.2.2 配置选项【掌握】

**el: //>> 挂载dom**

**data: //>> 准备数据,必须是函数，返回一个对象**

**methdos：{} //>> 方法（函数）**

**components: { //>> 注册组件**

**“标签名”: 引入的组件的变量名**

**}**

# 5. 指令【掌握】

## 5.1 什么是指令？【掌握】

指令是写在标签上的一种自定义属性，主要是把vue实例中的数据，输出到html中（vue实例对象和html关联）。

**<标签 id=”” class=”” V-\*=”xxx”></标签> // “ V-“ 开头的就是指令**

## 5.2 常用指令【掌握】

**>> v-text 和 v-html**

都可以把数据渲染到一对标签中间，但是v-text不能识别html标签，v-html可以解析html标签。

**作用： 页面动态数据的回显（常见：v-text直接用“{{表达式}}”替代，v-html显示富文本数据）。**

**>> v-show和v-if**

v-show: 通过控制css的display属性，来控制显示和隐藏。

v-if: 删除或重建DOM。

区别： 如果是频繁的切换显示隐藏，就是用v-show；v-if频繁操作dom，性能低。

**作用： 控制页面内容显示隐藏（常见：v-show显示/隐藏切换频繁，v-if显示/隐藏切换不频繁，但对隐藏安全要求较高）。**

**>> v-if 和 v-else-if 和 v-else**

这几个指令需要配合使用，逻辑和js中的条件判断语句一致，会从上往下，找到满足条件的第一个表达式，渲染该DOM.（指令元素间，必须紧密相间）

**作用： 用于页面结构中的逻辑判断。（常见：根据场景的不同，制定不同的显示规则）**

**>> v-for 循环**

循环数组和对象

//>> 循环数组

<ul>

<li v-for=”(元素的值， 元素的索引) in 数组”></li>

</ul>

//>> 循环对象

<ul>

<li v-for=”(元素的值， 键名，元素的索引) in 对象”></li>

</ul>

**作用： 用于根据模板，批量生产显示内容。（常见：数据列表）**

**>> v-model:**

只能使用于表单，让表单和数据双向绑定。

适用的标签：

input(输入框 单选框 多选框) select textarea

**作用： 用于获取用户输入表单的值（常见：表单数据获取）**

# 事件处理

## 6.1 事件简单回顾（了解）

事件核心4要素：

事件源（HTML元素）

事件句柄（事件名称）

事件处理函数（事件回调函数）

事件对象（记录事件上下文信息）

## 6.2 基础语法【掌握】

**>> v-on: 事件名**

绑定事件 （简写 **@**）

<标签 **v-on:事件类型=”函数名”**></标签>

<标签 **@事件类型=”函数名”**></标签>

## 6.3 事件内联处理器中的方法【掌握】

<标签 **@事件类型=”函数名(参数)”**></标签>

# 计算属性

## 计算属性computed【掌握】

>> 计算属性computed也是vue的一个配置选项，写法和methods一样，把复杂的计算逻辑都写在计算属性函数中，返回结果，通过函数的名字，可以直接使用该函数的返回结果。

**语法：**

**computed: {**

**函数名（）{ //>> 函数名可以直接在template中使用，可以输出此函数返回的结果。**

**// 函数的功能逻辑**

**return 结果**

**}**

**}**

**>> computed特点&和methods的区别 [重要]：**

1. **写法上一样，都是一个方法，但是computed必须返回一个结果。**
2. **computed有依赖缓存，如果依赖的数据没有发生改变，会直接使用使用缓存的结果，不会重新计算，只有依赖的数据发生了改变才会重新计算，性能高！而methods每次都会重新计算执行一遍。**

**>> 计算属性示例**

|  |
| --- |
| <template>    <div>      <!-- 使用计算属性 直接通过函数的名字使用 -->      <h1>总价： {{ totalPrice }}</h1>    </div>  </template>  <script>  export default {    data() {      return {        // 购物车数据        shopCart: [          { name: "apple", price: 3, count: 6 },          { name: "banana", price: 5, count: 7 },          { name: "orange", price: 6, count: 8 }        ]      };    },    computed: {      // 计算属性      totalPrice() {        let total = 0;        for (let i = 0; i < this.shopCart.length; i++) {          total += this.shopCart[i].price \* this.shopCart[i].count;        }        return total; // 返回结果      }    }  };  </script>  <style lang="scss" scoped>  </style> |

# 课程总结

## 8.1重点

* + - 1. Vue环境搭建。
      2. Vue组件化开发，自定义组件的使用。
      3. Vue指令的使用（特别是几个重点指令！！！）。
      4. Vue的事件绑定。
      5. 计算属性的使用。

## 8.2如何掌握？

* + - 1. 通过代码验证原理，多写增加熟练度。
      2. 大量练习才能带来熟练的操作！
      3. 将常见的用法截图保存到笔记本中，时常回顾。

# 9.课后练习

## 9.1 简答题（手抄笔记）

**难度：☆**

1. 手写vue的常见配置（名称和值格式）。（**☆**）
2. 简述组件使用的步骤。（**☆**）

## 9.2 操作题（电脑操作）

**难度：** **☆☆☆**

1、完成课堂代码。（**☆**）

2、使用Vue完成一个户籍管理系统 （**☆☆☆**）



1. 编写实现的思路
2. 实现户籍管理系统的页面
3. 实现用户的显示
4. 实现用户的添加和删除

3、根据数据实现购物车数据展示和汇总计算(扩展)（**☆☆☆☆☆**）

# 面试题

* + - 1. **v-if 跟 v-show 的区别？**
      2. **为什么组件中data是函数而不是对象？**
      3. **计算属性computed和方法methods的区别？**