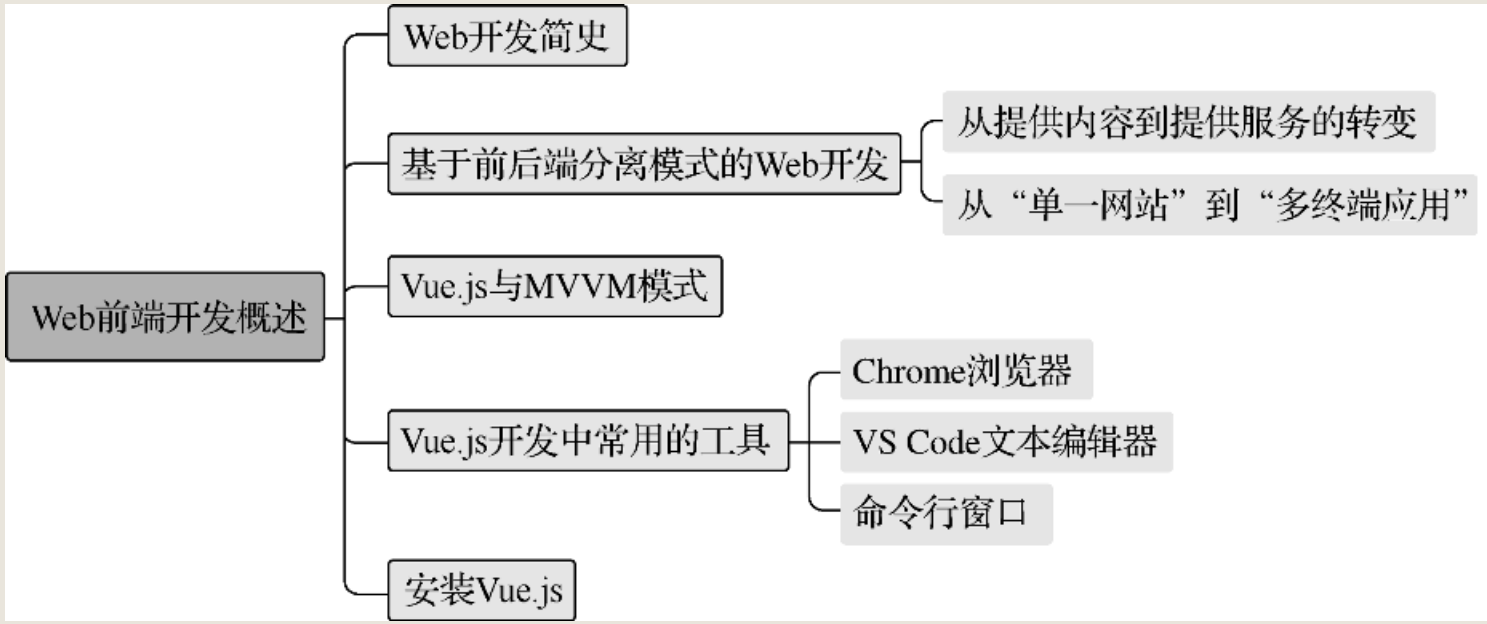
vue.js

VUE.JS 2024.7

第1章　Web前端开发概述

- Web开发简史
- 基于前后端分离模式的Web开发
- Vue.js的特性
- MVVM（Model-View-ViewModel）模式
- Vue.js的核心思想
- Vue.js开发中常用的工具
- 安装Vue.js
- DEMO（猜一个介于1和100之间的整数）

第1章　Web前端开发概述



第1章 Web前端开发概述

- Web开发简史
  - 早期阶段
  - 服务器端模板阶段
  - 服务器端MVC时代
  - 前后端分离时代

第1章 Web前端开发概述

- 基于前后端分离模式的Web开发
  - 从提供内容到提供服务转变

传统互联网3个特点：

- 使用场景固定且局限
- “内容”为主
- “服务”局限于特定领域

移动互联网3个特点：

- 使用场景触达社会每个角落
- 更多事物被连接到云端
- 海量“服务”

对技术上的3个要求：

- 客户端需求复杂化，大量应用流行，对用户体验的期望提高。
- 客户端渲染成为“刚需”。
- 客户端程序不得不具备完整的生命周期、分层架构和技术栈

第1章　Web前端开发概述

- 从“单一网站”到“多终端应用”

特点：

- 服务器端通过API输出数据，剥离“视图”。
- Web客户端变成独立开发和部署的程序，不再是服务器端Web程序中的“前端”层。
- 每个客户端都倾向于拥有专门为自己量身打造、可被自己掌控的API网站。

在移动时代，一个应用往往需要适配不同的终端形态：

- 桌面应用：传统的Windows应用、Mac应用
- 移动应用：iOS、安卓应用
- Web：通过浏览器访问的应用
- 超级APP：以微信小程序为代表的超级APP，成为新的应用程序平台。

第1章 Web前端开发概述

- Vue.js与MVVM模式
  - vue.js的特性
    - 轻量级
    - 数据绑定
    - 指令
    - 组件化管理
    - 插件化开发
    - 完整的工具链

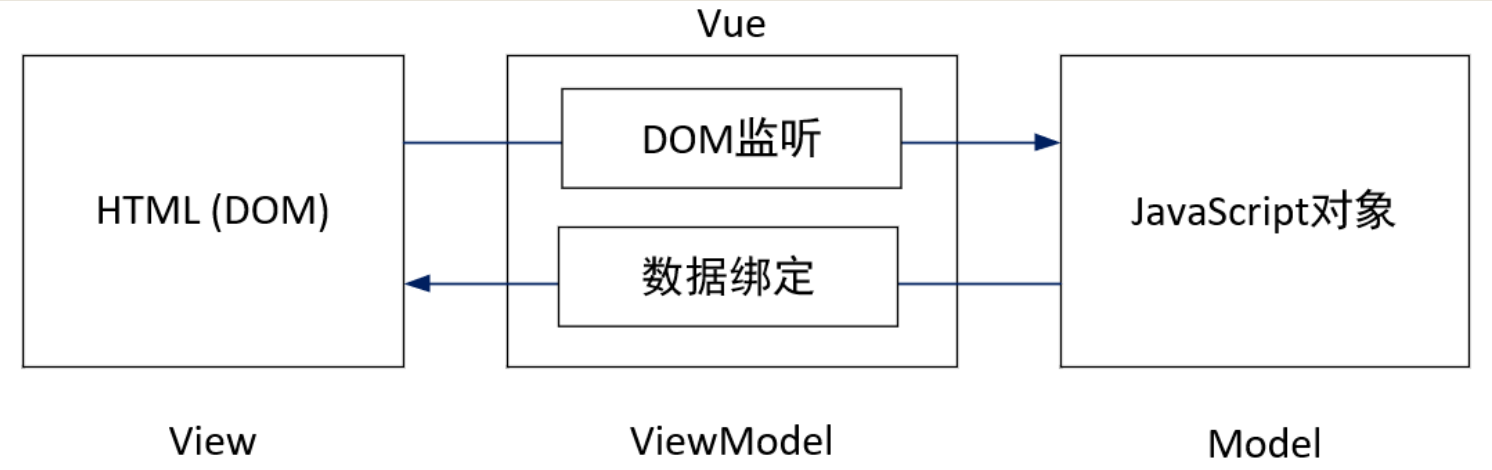
MVVM（Model-View-ViewModel）模式包括3个核心部分：

- Model：模型，核心的业务逻辑产生的数据对象，例如从数据库取出的数据，并做特定处理后得到的数据。
- View：视图，即用户界面。
- ViewModel：视图模型，用于链接匹配模型和视图之间的专用模型。

第1章 Web前端开发概述

Vue.js的核心思想包括两点：

- 数据的双向绑定，View和Model之间不直接沟通，而是通过ViewModel这个桥梁进行交互。
- 使用“声明式”的编程理念



第1章 Web前端开发概述

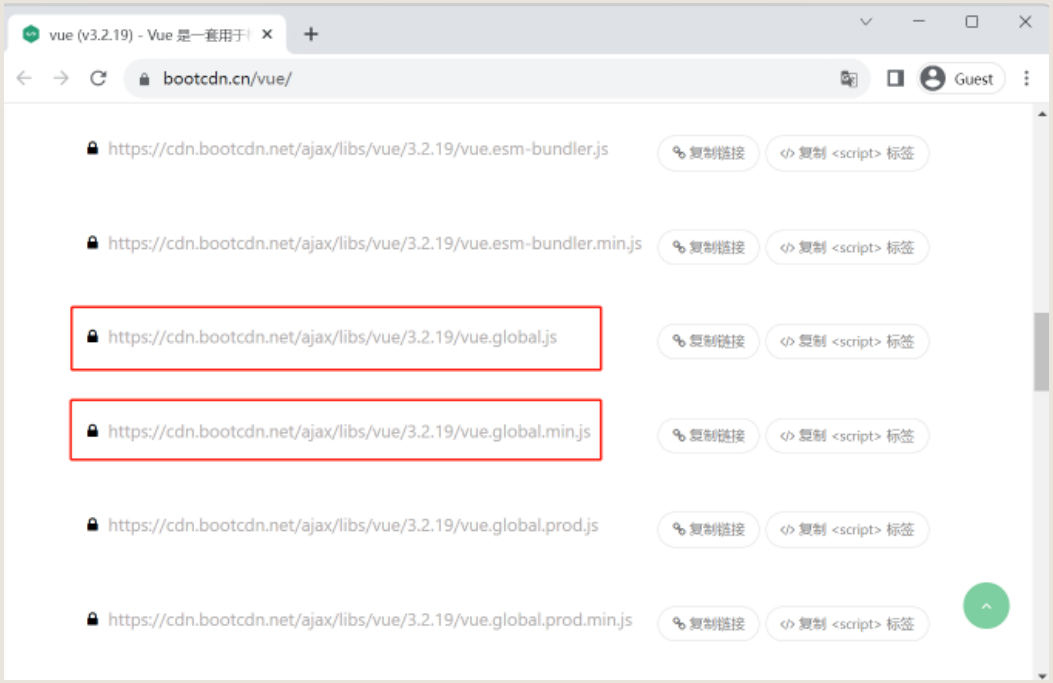
Vue.js开发中常用的工具：

- Chrome浏览器
- VS Code文本编辑器
- 命令行控制台

目前常见的浏览器内核有Trident、Gecko、Webkit、Blink这4种。

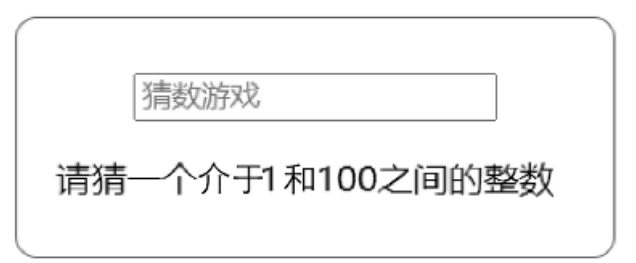
浏览器内核	说明
Trident	代表浏览器是IE，IE内核还被用在众多国内互联网公司推出的双核浏览器中，用作兼容模式
Webkit	代表浏览器是Safari、旧版的Chrome
Blink	代表浏览器是Chrome、Opera、新版的Edge
Gecko	代表浏览器是Firefox

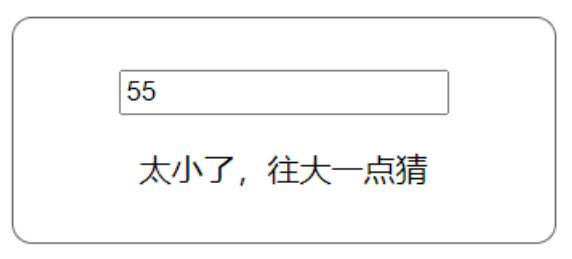
安装Vue.js

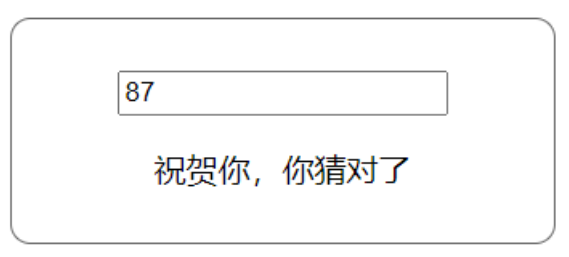



```
<script src="https://cdn.bootcdn.net/ajax/libs/vue/3.2.19/vue.global.js"></script>
```

上手实践：第一个Vue.js程序







第1章　Web前端开发概述

DEMO

第1章　Web前端开发概述

- Web开发简史
- 基于前后端分离模式的Web开发
- Vue.js的特性
- MVVM（Model-View-ViewModel）模式
- Vue.js的核心思想
- Vue.js开发中常用的工具
- 安装Vue.js
- DEMO（猜一个介于1和100之间的整数）

请看第2章——

VUE开发入门

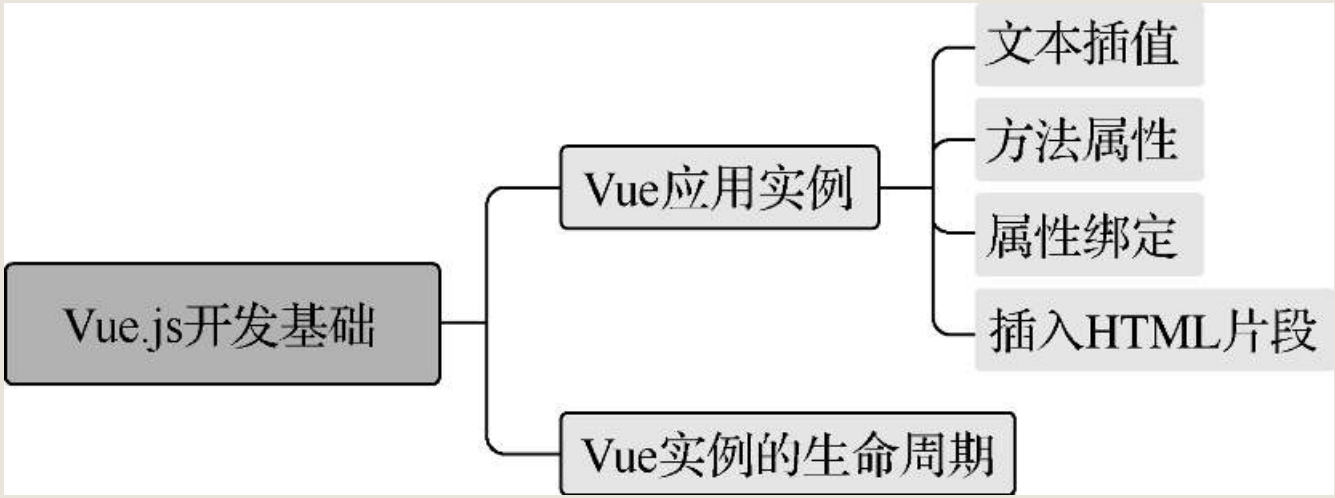
vue.js

VUE.JS 2024.7

第2章　Vue开发入门

- Vue应用实例
- DEMO
- Vue实例的生命周期
- DEMO（猜数字）

第2章　Vue开发入门



第2章　Vue开发入门

- Vue应用实例
  - 文本插值
  - 方法属性
  - 属性绑定
  - 插入HTML片段

根vue实例，具体语法形式如下：

```
const app = Vue.createApp({
  // 选项对象
});
```

DEMO

第2章 Vue开发入门

- Vue实例的生命周期
  - beforeCreate
  - created
  - beforeMount
  - mounted
  - beforeUpdate
  - updated
  - beforeUnmount
  - unmounted

DEMO

第2章　Vue开发入门

- Vue应用实例
- DEMO
- Vue实例的生命周期
- DEMO（猜数字）

请看第3章——
计算属性与侦听器

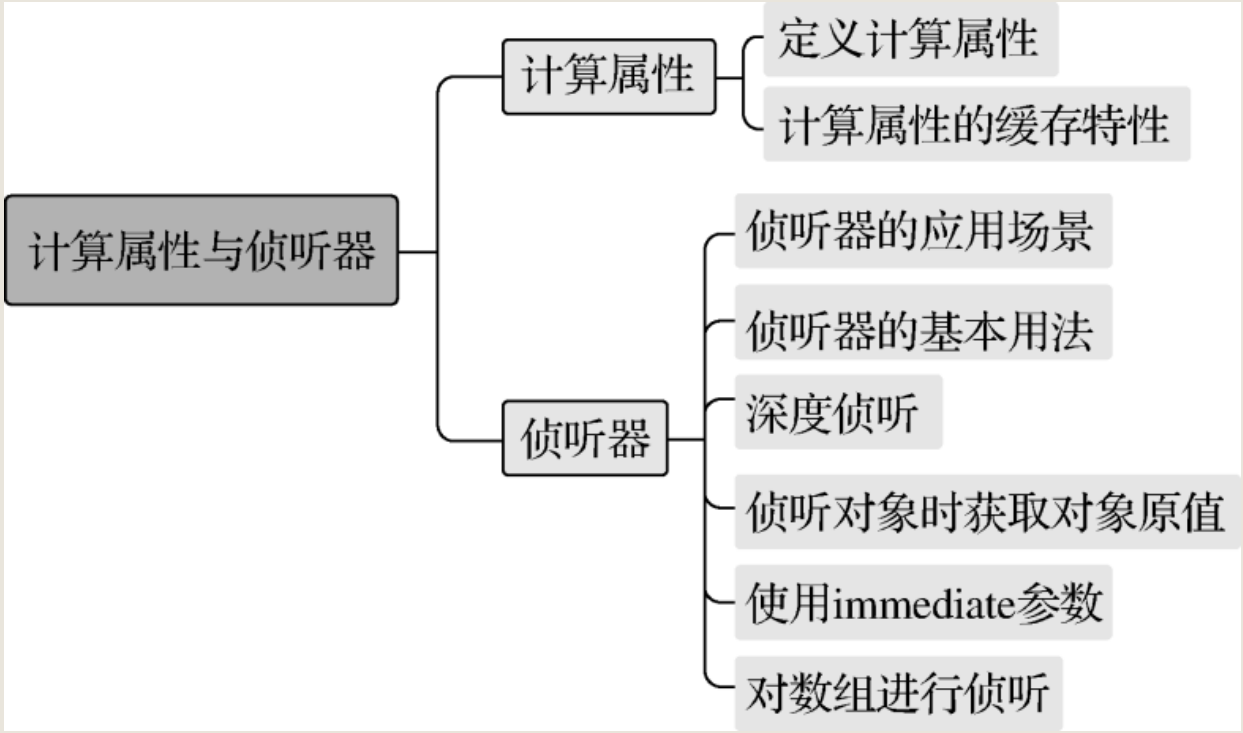
vue.js

VUE.JS 2024.7

第3章　计算属性与侦听器

- 计算属性
- DEMO
- 侦听器
- DEMO
- 对数组的侦听
- 替换数组可以被侦听到
- 使用深度侦听
- DEMO
- 总结

第3章　计算属性与侦听器



第3章　计算属性与侦听器

计算属性

- 定义计算属性
- 计算属性的缓存特性

第3章　计算属性与侦听器

DEMO

第3章　计算属性与侦听器

侦听器

- 侦听器的应用场景
  - 拦截操作
  - 耗时操作
- 侦听器的基本用法
- 深度侦听
- 侦听对象时获取对象原值
- 使用immediate参数

第3章　计算属性与侦听器

DEMO

第3章　计算属性与侦听器

- 对数组的侦听
  - 标准方法修改数组可以被侦听到
    - push() 尾部添加
    - pop() 尾部删除
    - unshift() 头部添加
    - shift() 头部删除
    - splice() 删除、添加、替换
    - sort() 排序
    - reverse() 逆序

第3章　计算属性与侦听器

替换数组可以被侦听到

- filter() 过滤
- concat() 拼接
- slice() 从已有的数组中返回选定的元素

第3章　计算属性与侦听器

使用深度侦听可以侦听到数组的某些变化：

- 直接通过下标的方式去修改数组，例如vm.items[5] = newValue。
- 直接通过修改数组的length属性的方式，例如vm.items.length = 10。

第3章　计算属性与侦听器

DEMO

第3章　计算属性与侦听器

总结：

- 如果彻底替换为一个新的数组，那么可以被侦听到。
- 如果侦听器已通过“{deep:true}”设置为“深度侦听”的，那么当修改对象元素的属性时，可以被侦听到。如果是数组本身被修改，也可以被侦听到。
- 虽然通过length属性可以修改数组长度，但尽量不要这样修改，建议改用其他标准方法显示数组长度的变化。

第3章　计算属性与侦听器

- 计算属性
- DEMO
- 侦听器
- DEMO
- 对数组的侦听
- 替换数组可以被侦听到
- 使用深度侦听
- DEMO
- 总结

请看第4章——
控制页面的CSS样式