**孙氏黑马web总结**

目录

[一. HTML 12](#_Toc32483468)

[1. HTML概念 12](#_Toc32483469)

[1.1 什么是HTML？ 12](#_Toc32483470)

[1.2 HTML用来做什么 12](#_Toc32483471)

[1.3 HTML的版本 12](#_Toc32483472)

[2. HTML基本语法 12](#_Toc32483473)

**[2.1](#_Toc32483474)** [标签语法 12](#_Toc32483474)

[2.2 单标签 12](#_Toc32483475)

[2.3 双标签 13](#_Toc32483476)

[2.4 标签属性语法 13](#_Toc32483477)

[2.5 注释 13](#_Toc32483478)

[2.6 基本结构 13](#_Toc32483479)

[2.7 字符集 13](#_Toc32483480)

[3. 常用标签 14](#_Toc32483481)

[3.1 标题hn标签 14](#_Toc32483482)

[3.2 段落p标签 14](#_Toc32483483)

[3.3 换行符br 14](#_Toc32483484)

[3.4 加粗b和 strong标签 14](#_Toc32483485)

[3.5 斜体效果i和 em 标签 14](#_Toc32483486)

[3.6 删除线del标签 14](#_Toc32483487)

[3.7 行内标签span标签 14](#_Toc32483488)

[3.8 块标签div标签 15](#_Toc32483489)

[3.9 语义标签-头部header标签 15](#_Toc32483490)

[3.10 语义标签-导航nav标签 15](#_Toc32483491)

[3.11 语义标签-节section标签 15](#_Toc32483492)

[3.12 语义标签-边栏aside标签 15](#_Toc32483493)

[3.13 语义标签-文章article标签 15](#_Toc32483494)

[3.14 语义标签-尾部footer标签 16](#_Toc32483495)

[3.15 超连接a标签 16](#_Toc32483496)

[3.16 图片img标签 16](#_Toc32483497)

[3.17 音频audio标签 16](#_Toc32483498)

[3.18 视频video标签 16](#_Toc32483499)

[4. 引入外部CSS、js 17](#_Toc32483500)

[4.1 引入外部CSS 17](#_Toc32483501)

[4.2 声明内部CSS 17](#_Toc32483502)

[4.3 引入外部JS 17](#_Toc32483503)

[4.4 声明内部JS 18](#_Toc32483504)

[5. 列表 18](#_Toc32483505)

[5.1 无序列表ul 18](#_Toc32483506)

[5.2 有序列表ol 18](#_Toc32483507)

[5.3 定义列表dl 19](#_Toc32483508)

[6. 表格 19](#_Toc32483509)

[7. 表单 20](#_Toc32483510)

[7.1 提交地址、方式 20](#_Toc32483511)

[7.2 输入框 20](#_Toc32483512)

[7.3 下拉框 21](#_Toc32483513)

[7.4 多行文本框（文本域） 22](#_Toc32483514)

[7.5 按钮标签 22](#_Toc32483515)

[7.6 7.6 相对与绝对路径 23](#_Toc32483516)

[8. 特殊符号 23](#_Toc32483517)

[二. CSS样式表 23](#_Toc32483518)

[1. CSS概念 23](#_Toc32483519)

[1.1 什么是CSS？ 23](#_Toc32483520)

[1.2 CSS的作用 23](#_Toc32483521)

[1.3 CSS的版本 24](#_Toc32483522)

[2. CSS基础语法 24](#_Toc32483523)

[2.1 CSS的样式属性语法 24](#_Toc32483524)

[2.2 引入CSS的三种方式 24](#_Toc32483525)

[3. CSS选择器 25](#_Toc32483526)

[3.1 什么是选择器？ 25](#_Toc32483527)

[3.2 3.2基本选择器 25](#_Toc32483528)

[3.3 层次选择器 26](#_Toc32483529)

[3.4 伪类 27](#_Toc32483530)

[3.5 伪元素 27](#_Toc32483531)

[3.6 属性选择器 28](#_Toc32483532)

[3.7 选择器优先级 28](#_Toc32483533)

[4. 文本字体 28](#_Toc32483534)

[4.1 字体大小 28](#_Toc32483535)

[4.2 字符粗细 28](#_Toc32483536)

[4.3 字体样式 29](#_Toc32483537)

[4.4 行高 29](#_Toc32483538)

[4.5 字体名称 29](#_Toc32483539)

[4.6 字体复合样式五合一 29](#_Toc32483540)

[4.7 字体颜色 30](#_Toc32483541)

[4.8 下划线 30](#_Toc32483542)

[4.9 字间距 31](#_Toc32483543)

[4.10 水平对齐 31](#_Toc32483544)

[4.11 文本缩进 31](#_Toc32483545)

[4.12 文字阴影 31](#_Toc32483546)

[5. 背景 31](#_Toc32483547)

[5.1 背景颜色 31](#_Toc32483548)

[5.2 设置一个背景图 32](#_Toc32483549)

[5.3 背景图平铺 32](#_Toc32483550)

[5.4 背景图位置 32](#_Toc32483551)

[5.5 背景图是否滚动 32](#_Toc32483552)

[5.6 背景复合五合一 33](#_Toc32483553)

[5.7 背景图大小 33](#_Toc32483554)

[5.8 CSS精灵图 33](#_Toc32483555)

[6. 盒子模型 33](#_Toc32483556)

[6.1 尺寸 33](#_Toc32483557)

[6.2 内边距 34](#_Toc32483558)

[6.3 外边距 34](#_Toc32483559)

[6.4 边框 35](#_Toc32483560)

[6.5 阴影和圆角边框 36](#_Toc32483561)

[6.6 盒子模式 37](#_Toc32483562)

[7. 布局定位 38](#_Toc32483563)

[7.1 浮动 38](#_Toc32483564)

[7.2 清除浮动造成的影响 38](#_Toc32483565)

[7.3 超出处理 39](#_Toc32483566)

[7.4 显示与隐藏 39](#_Toc32483567)

[7.5 定位 40](#_Toc32483568)

[8. 表格列表手型 41](#_Toc32483569)

[8.1 细边框 41](#_Toc32483570)

[8.2 清除引导符 41](#_Toc32483571)

[8.3 手型 41](#_Toc32483572)

[三. JavaScript交互 41](#_Toc32483573)

[1. Js概念 41](#_Toc32483574)

[1.1 什么是js 41](#_Toc32483575)

[1.2 Js的作用 42](#_Toc32483576)

[1.3 Js的版本 42](#_Toc32483577)

[2. 基本语法 42](#_Toc32483578)

[2.1 四种编写js的方法 42](#_Toc32483579)

[2.2 输出方式 43](#_Toc32483580)

[2.3 注释 44](#_Toc32483581)

[2.4 变量 44](#_Toc32483582)

[2.5 标识符命名 45](#_Toc32483583)

[3. 数据类型 46](#_Toc32483584)

[3.1 基本类型 46](#_Toc32483585)

[3.2 引用类型 47](#_Toc32483586)

[3.3 类型转换 47](#_Toc32483587)

[4. 运算符 48](#_Toc32483588)

[4.1 算术运算符 48](#_Toc32483589)

[4.2 比较运算符 48](#_Toc32483590)

[4.3 三元运算符 48](#_Toc32483591)

[4.4 逻辑运算符 49](#_Toc32483592)

[4.5 字符串拼接符 49](#_Toc32483593)

[4.6 赋值运算符 49](#_Toc32483594)

[5. 流程控制语句 49](#_Toc32483595)

[5.1 条件分支 49](#_Toc32483596)

[5.2 状态分支 50](#_Toc32483597)

[5.3 for循环(常用) 51](#_Toc32483598)

[5.4 while循环 51](#_Toc32483599)

[5.5 中止循环 52](#_Toc32483600)

[6. 数组 52](#_Toc32483601)

[6.1 什么是数组 52](#_Toc32483602)

[6.2 数组的定义 52](#_Toc32483603)

[6.3 数组的使用 52](#_Toc32483604)

[6.4 新增和修改数组元素 52](#_Toc32483605)

[6.5 删除数组元素 52](#_Toc32483606)

[6.6 数组的方法 53](#_Toc32483607)

[7. 函数 54](#_Toc32483608)

[7.1 函数的定义 54](#_Toc32483609)

[7.2 函数调用 54](#_Toc32483610)

[7.3 函数参数 54](#_Toc32483611)

[7.4 函数返回值 55](#_Toc32483612)

[7.5 变量的作用域 55](#_Toc32483613)

[7.6 递归函数 56](#_Toc32483614)

[8. 对象 56](#_Toc32483615)

[8.1 什么是对象 56](#_Toc32483616)

[8.2 定义对象 56](#_Toc32483617)

[8.3 使用对象 57](#_Toc32483618)

[8.4 对象方法中的this 57](#_Toc32483619)

[8.5 遍历对象 57](#_Toc32483620)

[8.6 内置对象 58](#_Toc32483621)

[9. Bom 60](#_Toc32483622)

[9.1 什么是BOM 60](#_Toc32483623)

[9.2 window对象 60](#_Toc32483624)

[10. DOM 61](#_Toc32483625)

[10.1 什么是DOM 61](#_Toc32483626)

[10.2 节点树 61](#_Toc32483627)

[10.3 节点操作 62](#_Toc32483628)

[10.4 事件 64](#_Toc32483629)

[四. JAVA Query 67](#_Toc32483630)

[1. Java query概念 67](#_Toc32483631)

[2. Java query版本 68](#_Toc32483632)

[3. jquery的引入及使用说明: 68](#_Toc32483633)

[4. 选择器 69](#_Toc32483634)

[4.1 基本选择器===重点 69](#_Toc32483635)

[4.2 过滤选择器 69](#_Toc32483636)

[4.3 表单选择器 71](#_Toc32483637)

[4.4 CSS样式操作 71](#_Toc32483638)

[4.5 属性操作 72](#_Toc32483639)

[五. Ajax 73](#_Toc32483640)

[1. Ajax基础 73](#_Toc32483641)

[1.1 Ajax基本概念： 73](#_Toc32483642)

[1.2 ajax优缺点 73](#_Toc32483643)

[1.3 ajax工作原理(通过一个注册流程剖析) 73](#_Toc32483644)

[1.4 使用ajax发送get请求 74](#_Toc32483645)

[1.5 服务器 74](#_Toc32483646)

[1.6 如何查看ajax请求 75](#_Toc32483647)

[1.7 使用post发送ajax请求 75](#_Toc32483648)

[1.8 phpStudy设置 76](#_Toc32483649)

[1.9 使用jQuery发送ajax的三种不同方式 76](#_Toc32483650)

[1.10 Json 77](#_Toc32483651)

[2. Ajax与前端模板 78](#_Toc32483652)

[2.1 MockJS实现前后端分离 78](#_Toc32483653)

[2.2 官方网站和实用实例 78](#_Toc32483654)

[2.3 Mockjs的安装和实用 78](#_Toc32483655)

[2.4 模板的概念 79](#_Toc32483656)

[六. html5 79](#_Toc32483657)

[1. html5的新特性 79](#_Toc32483658)

[2. Html5兼容性 79](#_Toc32483659)

[2.1 html4和html5的文档结构区别（掌握） 80](#_Toc32483660)

[2.2 html5语义标签 80](#_Toc32483661)

[2.3 .H5 form表单(了解) 80](#_Toc32483662)

[2.4 自定义属性（重点） 81](#_Toc32483663)

[2.5 其他属性（了解） 81](#_Toc32483664)

[2.6 本地存储（重点） 81](#_Toc32483665)

[七. Css3 82](#_Toc32483666)

[3. Css标准 82](#_Toc32483667)

[4. 与HTML5有什么关系 82](#_Toc32483668)

[5. css兼容性写法 82](#_Toc32483669)

[6. css3新特性 83](#_Toc32483670)

[7. 选择器（重点） 83](#_Toc32483671)

[8. 文本效果 84](#_Toc32483672)

[9. 阴影和边框 85](#_Toc32483673)

[10. css3flex布局 85](#_Toc32483674)

[10.1 felx布局 85](#_Toc32483675)

[10.2 如何设置为flex布局 85](#_Toc32483676)

[10.3 flex布局的几个概念 85](#_Toc32483677)

[10.4 flex属性 86](#_Toc32483678)

[10.5 web字体 87](#_Toc32483679)

[10.6 过渡 87](#_Toc32483680)

[11. 变形动画 88](#_Toc32483681)

[11.1 变形函数 88](#_Toc32483682)

[11.2 设置对象原点 88](#_Toc32483683)

[11.3 定义动画 88](#_Toc32483684)

[11.4 执行动画 89](#_Toc32483685)

[11.5 动画属性 89](#_Toc32483686)

[11.6 监听动画事件 90](#_Toc32483687)

[12. animate.css动画库 90](#_Toc32483688)

[12.1 animate.css使用 90](#_Toc32483689)

[12.2 swiper 90](#_Toc32483690)

[13. 响应式网站开发原理 93](#_Toc32483691)

[13.1 媒体类型 93](#_Toc32483692)

[13.2 引入方式 93](#_Toc32483693)

[13.3 dpr设备像素比 94](#_Toc32483694)

[13.4 视口 94](#_Toc32483695)

[13.5 媒体查询 94](#_Toc32483696)

[八. Canvas 94](#_Toc32483697)

[1. Canvas简介 94](#_Toc32483698)

[1.1 canvas中的属性 95](#_Toc32483699)

[九. echarts 报表 95](#_Toc32483700)

[1. 百度eCharts的强大 95](#_Toc32483701)

[2. 五分钟上手 95](#_Toc32483702)

[十. Less编译 96](#_Toc32483703)

[2.1 编译less 环境安装有2种方式 96](#_Toc32483704)

[2.2 less 用法 97](#_Toc32483705)

[十一. nodeJs基础 98](#_Toc32483706)

[1. nodeJs定义及安装 98](#_Toc32483707)

[1.1 安装： 98](#_Toc32483708)

[1.2 基础命令： 98](#_Toc32483709)

[2. 模块化概念 98](#_Toc32483710)

[2.1 作用： 98](#_Toc32483711)

[2.2 模块化的优势 99](#_Toc32483712)

[2.3 nodeJS和JS 区别？ 99](#_Toc32483713)

[2.4 主模块 99](#_Toc32483714)

[3. 模块组成 99](#_Toc32483715)

[3.1 3个类型： 99](#_Toc32483716)

[4. NPM 命令 100](#_Toc32483717)

[5. Cnpm 100](#_Toc32483718)

[十二. Gulp 100](#_Toc32483719)

[1. 使用步骤 101](#_Toc32483720)

[2. 插件网站 101](#_Toc32483721)

[3. 观察者模式 102](#_Toc32483722)

[十三. Php 102](#_Toc32483723)

[1. 前后端交互流程 102](#_Toc32483724)

[2. Php运行环境 103](#_Toc32483725)

[3. Php语法 103](#_Toc32483726)

[4. Php变量 103](#_Toc32483727)

[5. if条件判定语句和for循环语句 103](#_Toc32483728)

[6. 定义函数 104](#_Toc32483729)

[7. 连接数据库 104](#_Toc32483730)

[8. Php登录功能 104](#_Toc32483731)

[十四. Mysql 106](#_Toc32483732)

[1. 概念 106](#_Toc32483733)

[2. Mysql接口 106](#_Toc32483734)

[3. 安装mysql 106](#_Toc32483735)

[4. 配置环境 107](#_Toc32483736)

[5. 常用数据库命令 107](#_Toc32483737)

[6. 常用建表语句 107](#_Toc32483738)

[7. 操作数据常用命令 108](#_Toc32483739)

[8. 条件语句 108](#_Toc32483740)

[9. 数据类型 109](#_Toc32483741)

[十五. Js高级（重点） 109](#_Toc32483742)

[1. 高级js基础 109](#_Toc32483743)

[1.1 Json回顾 109](#_Toc32483744)

[1.2 Debugger 109](#_Toc32483745)

[1.3 变量划分 110](#_Toc32483746)

[1.4 变量内存分配 110](#_Toc32483747)

[1.5 变量复制 110](#_Toc32483748)

[1.6 参数传递 111](#_Toc32483749)

[1.7 变量类型检测 111](#_Toc32483750)

[1.8 对象获取或设置属性的两种方式 112](#_Toc32483751)

[2. 对象的浅拷贝与深拷贝 112](#_Toc32483752)

[2.1 作用域进阶 112](#_Toc32483753)

[3. Js编译原理与B包 113](#_Toc32483754)

[3.1 编译原理 113](#_Toc32483755)

[3.2 函数 113](#_Toc32483756)

[3.3 Iife立即执行函数 113](#_Toc32483757)

[3.4 Arguments 113](#_Toc32483758)

[3.5 B包 114](#_Toc32483759)

[4. 箭头函数 114](#_Toc32483760)

[5. 构造函数原型this 114](#_Toc32483761)

[5.1 编程思想 114](#_Toc32483762)

[5.2 类与实例对象 115](#_Toc32483763)

[5.3 构造函数创建对象 115](#_Toc32483764)

[5.4 方法过载 116](#_Toc32483765)

[5.5 原型 116](#_Toc32483766)

[5.6 This 119](#_Toc32483767)

[6. 继承 120](#_Toc32483768)

[6.1 概念 120](#_Toc32483769)

[6.2 Es5继承 120](#_Toc32483770)

[6.3 Es6继承 121](#_Toc32483771)

[7. 插件开发 121](#_Toc32483772)

[7.1 万能$ 121](#_Toc32483773)

[7.2 Jq与原生Dom的转化 122](#_Toc32483774)

[7.3 Jq事件 122](#_Toc32483775)

[8. Es6 122](#_Toc32483776)

[8.1 解构 122](#_Toc32483777)

[8.2 扩展运算符 123](#_Toc32483778)

[8.3 Es6关键字 123](#_Toc32483779)

[8.4 模板字符串 123](#_Toc32483780)

[8.5 默认参数 123](#_Toc32483781)

[8.6 对象增语法 123](#_Toc32483782)

[8.7 类class 124](#_Toc32483783)

[8.8 模块化：module 124](#_Toc32483784)

[8.9 Promise基本使用 125](#_Toc32483785)

[十六. Nodejs进阶 126](#_Toc32483786)

[1. Node回顾 126](#_Toc32483787)

[2. promise async 127](#_Toc32483788)

[2.1 Dos常用命令 127](#_Toc32483789)

[2.2 Nodejs作用域 127](#_Toc32483790)

[2.3 同步异步 127](#_Toc32483791)

[2.4 Promise 127](#_Toc32483792)

[2.5 Async和await 128](#_Toc32483793)

[3. Axios 129](#_Toc32483794)

[3.1 概念 129](#_Toc32483795)

[3.2 使用 129](#_Toc32483796)

[4. 文件、路径和http模块 130](#_Toc32483797)

[4.1 Fs模块 130](#_Toc32483798)

[4.2 Path模块 131](#_Toc32483799)

[4.3 Url模块 132](#_Toc32483800)

[4.4 http模块 132](#_Toc32483801)

[5. Express框架 133](#_Toc32483802)

[5.1 概念 133](#_Toc32483803)

[5.2 脚手架 133](#_Toc32483804)

[5.3 脚手架安装 133](#_Toc32483805)

[5.4 使用脚手架创建项目步骤 134](#_Toc32483806)

[5.5 项目目录 134](#_Toc32483807)

[5.6 相应对象res和请求对象req 135](#_Toc32483808)

[5.7 前后端交互流程 135](#_Toc32483809)

[6. Websocket通讯协议 135](#_Toc32483810)

[6.1 介绍 135](#_Toc32483811)

[6.2 使用 135](#_Toc32483812)

[十七. mvc与mvvm架构模式 136](#_Toc32483813)

[1. 打包上线流程 136](#_Toc32483814)

[1.1 流程： 136](#_Toc32483815)

[1.2 搭建express服务器： 137](#_Toc32483816)

[2. Mvc 137](#_Toc32483817)

[2.1 书写步骤 137](#_Toc32483818)

[2.2 三层架构图 138](#_Toc32483819)

[3. Mvvm 138](#_Toc32483820)

[3.1 概念 138](#_Toc32483821)

[3.2 Vue模式 139](#_Toc32483822)

[3.3 空电脑搭建项目 139](#_Toc32483823)

[3.4 Vue指令 140](#_Toc32483824)

[3.5 Vue生命周期 141](#_Toc32483825)

[十八. Vue 142](#_Toc32483826)

[1. Vue简介 142](#_Toc32483827)

[2. Vue上手 143](#_Toc32483828)

[2.1 安装 143](#_Toc32483829)

[2.2 Vue资源树 145](#_Toc32483830)

[2.3 结构描述 145](#_Toc32483831)

[3. 自定义组件 146](#_Toc32483832)

[4. Vue指令 146](#_Toc32483833)

[5. 生命周期： 147](#_Toc32483834)

[6. 路由 148](#_Toc32483835)

[7. 其他属性 150](#_Toc32483836)

[8. 动画 151](#_Toc32483837)

[9. 组件通信 152](#_Toc32483838)

[9.1 父传子 152](#_Toc32483839)

[9.2 子传父 154](#_Toc32483840)

[9.3 兄弟传参 154](#_Toc32483841)

[10. 插槽 155](#_Toc32483842)

[11. Element-ui 156](#_Toc32483843)

[12. betterscroll 156](#_Toc32483844)

[13. Vuex【重点】 157](#_Toc32483845)

[13.1 下载 157](#_Toc32483846)

[13.2 引入 157](#_Toc32483847)

[13.3 使用 158](#_Toc32483848)

[13.4 取值都放在computed中，提高代码性能，不会重复取值 158](#_Toc32483849)

[十九. 公司组成： 159](#_Toc32483850)

# HTML

## HTML概念

### 什么是HTML？

超文本标记语言

### HTML用来做什么

网站的骨架

### HTML的版本

HTML 4.01——1999年12月24日，W3C推荐标准

HTML 5——2014年10月29日，万维网联盟宣布，经过接近8年的艰苦努力，该标准规范终于制定完成。保留HTML4所有功能，增加很多新的功能。

广义论及 HTML5 时,实际指的是包括 HTML、CSS 和 JavaScript 在内的一套技术组合。

W3C：万维网联盟，是Web技术领域最具权威和影响力的国际中立性技术标准机构，该组织制定Web技术标准。

## HTML基本语法

### 标签语法

开始标签是被尖括号包围的元素名。

结束标签是被尖括号包围的斜杠和元素名。

例如：<h1>文本内容</h1> <b></b>

### 单标签

单标签是指没有结束标签的标签，由一个标签构成即为开始也是结束；

例如：<hr /> <br />

### 双标签

双标签是指由开始标签和结束标签组成，

例如：<b>内容</b>

### 标签属性语法

<标签名 属性名="属性值" 属性名="属性值" ....>

注意事项：

属性一般写在开始标签上

属性总是以名称/值对的形式存在（简称：名值对）

属性的值建议大家使用双引号（可以加双引号，也可以加单引号，也可以不加引号；千万不要在一个值又加单引号又加双引）

### 注释

语法：<!-- 注释内容 -->

含义：注释标签用来在源文档中插入注释。注释会被浏览器忽略。您可使用注释对您的代码进行解释，这样做有助于您在以后的时间对代码的理解。

### 基本结构

<!DOCTYPE html>

<html>

<head></head>

<body></body>

</html>

DOCTYPE: 文档声明

html: 标签告知浏览器其自身是一个 HTML 文档

head: 标签用于定义文档的头部（文档的描述信息）

body: 标签用于定义文档的主体（文档的主体内容）

### 字符集

UTF-8 国际通用编码字符集，UNICODE的灵活版

GBK/GB2312 中文编码字符集

UNICODE 大字符集，包含了地球上所有语言的编码

注意：2个地方必须保持一致（文件编码+HTML编码声明）

## 常用标签

### 标题hn标签

hn标签不是一个标签，而是一组标签的缩写：h1 h2 h3 h4 h5 h6定义和用法:

<h1> - <h6> 标签可定义标题。<h1> 定义最大的标题。<h6> 定义最小的标题

### 段落p标签

<p> 标签定义段落。

### 换行符br

<br / > 可插入一个简单的换行符。

单标签

### 加粗b和 strong标签

<b> 标签是基于内容的样式标签 <strong> 的物理版本，但它没有后者的扩展意义。

注意： HTML5中建议使用 <strong> 而不使用 <b>

### 斜体效果i和 em 标签

注意： HTML5中建议使用 <em> 而不使用 <i>

### 删除线del标签

<del> 定义文档中已被删除的文本。

### 行内标签span标签

<span> 标签被用来组合文档中的行内元素。

注释：span 没有固定的格式表现。当对它应用样式时，它才会产生视觉上的变化。

### 块标签div标签

<div> 可定义文档中的分区或节。

注意：

div span两个标签没有特殊的功能，只是作为一个内容容器，但使用频率很高，主要与CSS配合使用。

Div：块标签(它会独占一行，主要用于布局)

Span:行内标签(不会独占一行，用于勾选需要CSS修饰的内容)

块标签与行内标签的区别：

块标签独占一行，宽度默认为100%；

行内标签多个并排一行，宽高为内容大小。

### 语义标签-头部header标签

定义 section 或 page 的页眉。（头部）

### 语义标签-导航nav标签

<nav> 定义导航链接。

语义标签-主体nav标签

<main> 定义页面主要内容,它不应包含在文档中重复出现的内容，比如侧栏、导航栏、版权信息、站点标志或搜索表单。

### 语义标签-节section标签

定义文档中的节（section、区段）。

### 语义标签-边栏aside标签

定义页面的侧边栏内容。

### 语义标签-文章article标签

定义页面独立的内容区域。

### 语义标签-尾部footer标签

定义 section 或 document 的页脚。

### 超连接a标签

基本语法：

<a 属性="属性值">~标签内容~</a>

Href URL 链接的目标 URL。

Name section\_name 规定锚的名称。

target

• \_blank 从新的标签页打开

• \_self 自身标签打开（默认）

• \_parent 从父级打开

### 图片img标签

基本语法：

<img src="图片的路径" alt="图片说明文字" />

### 音频audio标签

基本语法：

<audio src="音乐文件路径url" controls="controls">

您的浏览器不支持 audio 标签。

</audio>

Src url 要播放的音频的 URL。

Autoplay 如果出现该属性，则音频在就绪后马上播放；

新版本浏览器已经禁止自动播放，解决方法如下：

获取audio的的Dom节点，调用play方法；

Controls 如果出现该属性，则向用户显示控件，比如播放按钮

### 视频video标签

基本语法：

<video src="视频文件路径" controls="controls">

您的浏览器不支持 video 标签。

</video>

Src 要播放的视频的 URL。

Controls 如果出现该属性，则向用户显示控件，比如播放按钮。

Height 设置视频播放器的高度。

Width 设置视频播放器的宽度。

Autoplay 如果出现该属性，则视频在就绪后马上播放。

Loop 循环播放。

新版本浏览器已经禁止自动播放，解决方法如下：

<video

src="xxxx/xxx.mp4"

autoplay="autoplay"

loop="loop"

muted="muted"

controls="controls"

controlsList="nodownload"

height="100%" width="100%">

</video>

## 引入外部CSS、js

### 引入外部CSS

1） 将样式写在独立的 .css 文件中

2） 在网页中使用 link标签 引入到当前页面来

特征：对所有引入的网页都有效，需要写选择器

示例：

-----index.css

h1{

color:red;

}

-----index.html

<link rel="stylesheet" type="text/css" href="index.css"/>

### 声明内部CSS

将样式写在 style标签 内

特征：对当前网页的所有标签都有效， 需要有选择器

示例：

<style>

h1{color:red;}

</style>

### 引入外部JS

<script src="js文件"></script>

### 声明内部JS

<script> js代码</script>

## 列表

### 无序列表ul

基本语法：

<ul >

<li>文本内容</li>

</ul>

注意事项

1） <ul>标签对里面添加<li>标签，<ul> <li> 是组合使用的，li代表列表里面的每一个列表项。

2） li标签对里面可以添加任何内容（可以嵌套标签）

3） 无序列表通常会自动生成一个小圆点效果，在实际应用中通常要去除。

### 有序列表ol

基本语法：

<ol>

<li>小苹果</li>

</ol>

注意事项：

有序列表ol上的type属性有几个值：

1）A 代表列表项前面显示的是大写字母；

2）a表示显示小写字母；

3)I代表用大写罗马数字显示；

4)i代表用小写罗马数字显示。

5)默认情况下显示数字；

### 定义列表dl

基本语法：

<dl>

<dt>计算机</dt>

<dd>用来计算的仪器 ... ...</dd>

<dt>显示器</dt>

<dd>以视觉方式显示信息的装置 ... ...</dd>

</dl>

## 表格

<table 属性="属性值">

<tr>

<td></td>

</tr>

<tr>

<td></td>

</tr>

</table>

<table> ：定义表格

<tr> ：定义表格的行

<td> ：定义表格中的单元格

<th> : 产生一个表头单元格

<caption>：定义表格标题

<thead> ：定义表格中的表头内容

<tbody> ：定义表格中的主体内容

<tfoot> ：定义表格中的表注内容（脚注）

rowspan : 跨行合并（纵向）

一个单元格占一列内多个单元格的位置 跨n行，在把它下面n-1 个 tr的td删除

colspan : 跨列合并（横向）

一个单元格占一行内多个单元格的位置 跨n列，就把下面的n-1个td删除

属性：

Border： 表格边框的宽度。

Cellpadding： px % 规定单元边沿与其内容之间的空白。

Cellspacing：px % 规定单元格之间的空白。

Width： % pixels 规定表格的宽度。

## 表单

### 提交地址、方式

基本语法：

<form action="URL" method="get|post">...</form>

action

URL 规定当提交表单时向何处发送表单数据。

enctype

上传文件时必须指定为：multipart/form-data

method

get(默认) post

规定用于发送 form-data 的 HTTP 方法。

### 输入框

#### 7.2.1 text普通文本框 默认

<input type="text" />

#### 7.2.2 password密码框 内容显示为\*号

<input type="password" />

#### 7.2.3 radio单选 一组单选必需name相同

<input type="radio" />

#### 7.2.4 checkbox多选 一组多选必需name相同

<input type="checkbox" />

#### 7.2.5 file 文件上传框 可以选择文件进行提交

<input type="file" />

#### 7.2.6 hidden隐藏域 隐藏控件，但是会被提交

<input type="hidden" />

#### 7.2.7 Date日期选择框

<input type="Date" />

 Date 日期选择框

 Datetime 日期时间选择框

 datetime-local 日期和时间 (无时区)选择框

 month 月份选择器

 time 时间选择框

 week 周和年选择框

#### 7.2.8 number 数值的输入域

<input type="number" />

#### 7.2.9 submit 提交按钮 点击后就提交表单

<input type="submit" />

#### 7.2.10 reset 重置按钮 回到最初状态（注意：不是清空）

<input type="reset" />

#### 7.2.11 button 普通按钮

<input type="button" />

### 下拉框

基本结构：

<select name=””>

<option></option>

</select>

select属性：

multiple 规定可选择多个选项。

name 规定下拉列表的名称。

option属性：

selected 规定选项（在首次显示在列表中时）表现为选中状态。

value 定义送往服务器的选项值。

注意：

当option中有value的时候，传将value传到后台

没有value的时候，将它显示的内容传到后台

### 多行文本框（文本域）

例如：<textarea rows="行数" cols="列数" name="intro">

我是一个中国人

</textarea>

属性：

rows="行数"

cols="列数

### 按钮标签

#### 7.5.1 提交按钮

<button typ="submint">按钮的内容 </button>

#### 7.5.2 普通按钮

<button typ="button">按钮的内容 </button>

#### 7.5.3 标注标签

Label的两种写法：（）防手抖

1）<label> <input type="radio" name="gender" id="male" />男 </label>

2）<input type="radio" name="gender" id="male" /><label for="male">男</label>

### 7.6 相对与绝对路径

#### 7.6.1 相对路径

./ 当前目录

../ 回到上一层目录

目录名/ 当前目录的下级目录

#### 7.6.2 绝对路径

[http://www.\*\*\*.com](http://www.***.com) 绝对路径以协议（http://、https://）或者以“/”作为前缀：

<file:///E:/> 一般网络路径（本地地址）

/news.html 基于网站根路径（/）

## 特殊符号

&lt;

&gt;

&amp;

&nbsp;空格（正常一个空格键）

&emsp;小空格（一个中文字符）

&copy;

# CSS样式表

## CSS概念

### 什么是CSS？

即层叠样式表，简称样式表

### CSS的作用

1） 遵循W3C标准，符合WEB2.0标准。

2） 减少重复格式化，减少网页体积，加快下载和访问速度。

3） 符合内容与表现形式分离的原则，方便于搜索引擎抓取到有用的内容。

4） 便于更新和维护，成千上万的网页只需要修改CSS便可以更改显示外观。

5） 浏览器干扰相对较小，实现一些HTML格式化不能实现的高级功能。

### CSS的版本

什么是CSS3?

CSS3是CSS规范的最新版，是css2.1的升级版本，弥补了CSS2很多不足之处。CSS层叠样式表告诉浏览器如何渲染你的Web页面。

## CSS基础语法

### CSS的样式属性语法

选择器名称{

属性名称1：属性的值1；

属性名称1：属性的值1；

……

}

### 引入CSS的三种方式

#### 2.2.1 行内样式

内联定义即是在对象的标记内使用对象的style属性定义适用其的样式表属性。

<p style="font-family: '微软雅黑'; font-weight: bold; font-size: 24px; ">新闻标题</p>新闻正文

#### 2.2.2 内部样式

在HTML文档的<HEAD>标记中插入一对<STYLE>...</STYLE>标签。

<p class="NewsTitle">新闻标题</p>新闻正文

#### 2.2.3 外部样式

先建立外部样式表文件（\*.css），然后在HTML中使用的link标签将其引入到网页中。

<p class="NewsTitle">新闻标题</p>新闻正文

## CSS选择器

### 什么是选择器？

选择需要添加样式的目标对象。

选择器 {属性：值}

选择器名称{

属性名称1：属性的值1；

属性名称2：属性的值2；

属性名称3：属性的值3；

……

}

### 3.2基本选择器

#### 3.2.1 ID选择器

#ID的值{ 样式表 }

所有的HTML标签都有一个id属性

我们可以为这个id设定一个值，然后再写选择器来匹配这一个值

#### 3.2.2 类选择器

. 类的值{ 样式表 }

所有的HTML标签都有class属性，为这个class设定一个值，然后就可以使用类选择器来匹配这一个值。并对其应用样式。

#### 3.2.3 标签选择器

标签名{ 样式表 }

例如：div{样式表 }，span{样式表}，ul{样式表}…

必须是HTML语法中固有的标签，而不能使自己定义的名称。

#### 3.2.4 通用选择器

\*{ 样式表 }

\* 星号选择器，也称为通用选择器

作用：选择HTML页面上所有的元素

#### 3.2.5 选择器分组

作用：将同样的样式应用于多个选择器，可以将选择符以逗号分隔的方式并为组。

语法：

E1,E2,E3

示例：

.td1,div a,body { font-size:14px; }

### 层次选择器

#### 3.3.1 后代选择器

包含选择器E1 E2（后代关系）

作用：选择所有被E1包含的E2。

语法： E1 E2 { 样式 }

#### 3.3.2 子选择器

子选择器 E1>E2（父子关系）(重点)

作用：选择所有作为E1子对象的E2。

语法： E1 > E2 { 样式 }

### 伪类

#### 3.4.1 未访问的连接

E : link 未访问的链接（a标签）

#### 3.4.2 已访问的连接

E : visited 已访问的链接

#### 3.4.3 鼠标移上去的连接

E : hover 鼠标悬停状态

#### 鼠标按住不动的连接

E : active 选定的激活状态

#### 获取到焦点的状态

E : focus

设置元素在成为输入焦点（该元素的onfocus事件发生）时的样式。

#### 获取第一个子元素

E : first-child { 样式 }

#### 获取最后一个子元素

E : last-child { 样式 }

#### 获取任意位置的子元素

E : nth-child (数值第几个或者是函数 ) { 样式 }

偶数：2n

奇数：2n-1

注意：索引是从0开始

### 伪元素

#### 3.5.1 在标签内容前添加内容

::before {

content :具体添加内容

}

#### 3.5.2 在标签内容后添加内容

::after {

content :具体添加内容

}

### 属性选择器

#### 3.6 1 通过属性名选择

E[ 属性名 ] { 样式 }

选择具有‘属性名’属性的E元素。

#### 3.6.2 通过属性名和属性值选择

E[ 属性名=属性值 ] { 样式 }

选择具有属性且属性值相等的E元素。

### 选择器优先级

!important > 行内样式 > #id > .class > tag > \* > 继承 > 默认

在选择器优先级相同的情况下，我们参照就近原则

## 文本字体

### 字体大小

语法： font-size : absolute-size | relative-size | length

absolute-size : 绝对值字体。如： 50px

relative-size : 相对于父对象中字体尺寸进行调节。如：1em

### 字符粗细

语法： font-weight :normal | bold | bolder | lighter | number

normal : 正常的字体。

相当于number为400。声明此值将取消之前任何设置

bold : 粗体。

相当于number为700。也相当于b标签的作用

### 字体样式

语法： font-style :normal | italic | oblique

参数：

normal : 正常的字体

italic : 斜体。对于没有斜体变量的特殊字体，将应用oblique

oblique : 倾斜的字体

### 行高

语法： line-height :normal | length

注： 值可以是固定值，也可以是相对值。

### 字体名称

语法： font-family : name

注意：

1）字体名称可以是英文也可以是中文。

2）英文字体如果出来多个单词需要用双引号包括。

### 字体复合样式五合一

语法： font :font-style font-weight font-size/line-height font-family

/\*复合写法\*/

p { font:italic bold 12px/30px arial,sans-serif,宋体;}

### 字体颜色

语法： color :color

CSS中颜色的三种表现方式：

1）英文名称

红色：red 绿色：green 蓝色：blue 黄色：yellow ...

2）16进制颜色值（#RRGGBB）

0-9 10-15（A-F或a-f）

红色：#ff0000 绿色：#00ff00 蓝色：#0000ff 黄色：#ffff00 ...

这种模式可以简写，简写原则为： RGB三段分别相同时可以简写成3位。

正确的： #ff0000 => #f00 #00ff00 => #0f0 #228833 => #283

错误的： #f10000 => #f100 #00ab11 => #0ab1 #228e36 => #28e36

3）rgb或 rgba

红色： Rgb(255,0,0) 绿色：Rgb(0,255,0) 蓝色：Rgb(0,0,255) 黄色：Rgb(255,255,0)

Rgba(255,0,0,1) 最后一位是alpha透明通道。

### 下划线

语法： text-decoration :none | underline | blink | overline | line-through

none : 　无划线

underline : 　下划线

blink : 　闪烁

line-through : 　贯穿线（删除线）

overline : 　上划线

### 字间距

语法： letter-spacing :normal | length

### 水平对齐

语法： text-align :left | right | center | justify

参数：

left：　左对齐

right： 右对齐

center：居中

Justify：两端对齐

### 文本缩进

语法： text-indent : length

注意：缩进常用 em

### 文字阴影

text-shadow属性：文本添加阴影效果；

语法：

text-shadow: X轴偏移值 Y轴偏移值 阴影颜色;

text-shadow: X轴偏移值 Y轴偏移值 阴影羽化半径值 阴影颜色;

## 背景

### 背景颜色

语法：

background-color : transparent | color

参数：

transparent : 　背景色透明

color : 　指定颜色。

### 设置一个背景图

语法：

background-image : none | ur（）

参数：

none : 无背景图

url : 使用绝对或相对地址指定背景图像

### 背景图平铺

语法：

background-repeat : repeat | no-repeat | repeat-x | repeat-y

参数：

repeat : 背景图像在纵向和横向上平铺

no-repeat : 背景图像不平铺

repeat-x : 背景图像在横向上平铺

repeat-y : 背景图像在纵向平铺

### 背景图位置

语法：

background-position : position | position

length : 百分比 | 由浮点数字和单位标识符组成的长度值。

参数：

length : 百分比 | 由浮点数字和单位标识符组成的长度值。

position : top | center | bottom | left | center | right

### 背景图是否滚动

语法：

background-attachment: fixed| scroll

参数：

scroll 默认。背景图像会随着页面其余部分的滚动而移动。 （绑定到body上）

fixed 当页面的其余部分滚动时，背景图像不会移动。（绑定到HTML标签上）

### 背景复合五合一

语法：

Background： background-color（颜色） | background-image（背景图片） | background-repeat （平铺） | background-position （起始位置）

/\* 复合完整写法：\*/

p{ backgrouond-color:red; background-image:url(aaa.jpg); background-position: left 30px; background-repeat: repeat-x; }

/\* 常见写法（简写）\*/

p { background: red url(aaa.jpg) repeat-x left 30px fixed; }

### 背景图大小

background-size ：设置背景图片的大小。

auto： 背景图像的真实大小。

cover： 将背景图像等比缩放到完全覆盖容器，背景图像有可能超出容器。

contain： 将背景图像等比缩放到宽度或高度与容器的宽度或高度相等，背景图像始终被包含在容器内。

### CSS精灵图

X轴一般为负，表示向左偏移多少

Y轴一般为负，表示向上偏移多少

## 盒子模型

### 尺寸

高度 宽度

最小高度 min-height

最大高度max-height

### 内边距

上边内边距 padding-top

语法： padding-top : length、

右边内边距 padding-right

语法： padding-top : length

下边内边距 padding-bottom

语法： padding-bottom : length

左边内边距 padding-left

语法： padding-left : length

复合样式：

语法： padding : length

示例：

p { padding: 20px; }

如果只提供一个，将用于全部的四条边。

如果提供两个，第一个用于上－下，第二个用于左－右。

如果提供三个，第一个用于上，第二个用于左－右，第三个用于下。

如果提供全部四个参数值，将按上－右－下－左的顺序作用于四边。

### 外边距

上边外边距 margin-top

语法： margin-top : length

右边外边距 margin-right

语法： margin-right : length

下边外边距 margin-bottom

语法： margin-bottom : length

左边外边距 margin-left

语法： margin-left : length

复合样式：

语法： margin : length

常用：

p { margin: 0 auto; } （ 板块居中 ）

如果只提供一个，将用于全部的四条边。

如果提供两个，第一个用于上－下，第二个用于左－右。

如果提供三个，第一个用于上，第二个用于左－右，第三个用于下。

如果提供全部四个参数值，将按上－右－下－左的顺序作用于四边。

外边距在不同浏览器的效果会有很大不同，建议尽量少用。

技巧：版块居中可以使用左右自动外边距。

### 边框

#### 6.4.1边框颜色

边框颜色 border-color

语法： border-color : color

#### 6.4.2 边框线大小

边框宽度 border-width

语法： border-width : medium | thin | thick | length

如果只提供一个，将用于全部的四条边。

如果提供两个，第一个用于上－下，第二个用于左－右。

如果提供三个，第一个用于上，第二个用于左－右，第三个用于下。

如果提供全部四个参数值，将按上－右－下－左的顺序作用于四个边框。

要有边框线型的情况下，才能设置边框的宽度。

通常边框宽度都会设置成一个实际的象素值。

#### 6.4.3 边框线型

边框线型 border-style

语法： border-style : none | dotted | dashed | solid | double

参数：

none（无边框）

dotted（点线）

dashed（虚线）

solid（实线）

double（双线）

groove（凹槽）

ridge（凸槽）

inset（凹边）

outset（凸边）

#### 6.4.4边框复合样式

上边框 border-top

右边框 border-right

下边框 border-bottom

左边框 border-left

border

语法： border : border-width | border-style | border-color

### 阴影和圆角边框

#### 6.5.1 阴影

box-shadow属性：设置盒子的阴影。

语法：

box-shadow : X轴偏移值 Y轴偏移值 模糊值（不为负） 阴影颜色;

取值：

none： 无阴影

①： 第1个长度值用来设置对象的阴影水平偏移值。可以为负值

②： 第2个长度值用来设置对象的阴影垂直偏移值。可以为负值

③： 如果提供了第3个长度值则用来设置对象的阴影模糊值。不允许负值

④： 如果提供了第4个长度值则用来设置对象的阴影外延值。不允许负值

color： 设置对象的阴影的颜色。

inset： 设置对象的阴影类型为内阴影。该值为空时，则对象的阴影类型为外阴影

#### 6.5.2 圆角边框

border-radius：设置边框为圆角。

border-radius：数值；

默认值：0

取值：

length： 用长度值设置对象的圆角半径长度。不允许负值

percentage： 用百分比设置对象的圆角半径长度。不允许负值

### 盒子模式

#### 6.6.1 标准盒子

标准模式的盒子模型：

padding和border不被包含在定义的width和height之内。

对象的实际宽度等于设置的width值和border、padding之和，即 ( Element width = width + border + padding ) 。

#### 6.6.2 怪异盒子

怪异模式的盒子模型：

padding和border被包含在定义的width和height之内。

对象的实际宽度就等于设置的width值，即使定义有border和padding也不会改变对象的实际宽度，即 ( Element width = width ) 。

#### 6.6.3 转换盒子模式的属性

语法：box-sizing：content-box | border-box

## 布局定位

### 浮动

#### 7.1.1 什么是浮动？

float是css样式中的定位属性的浮动布局；

浮动的框可以向左或向右移动，直到它的外边缘碰到父级盒子（父容器）或另一个浮动框（子元素）的边框为止。

只要设置了浮动，就将盒子脱离了文本流。

#### 7.1.2 左右浮动用法

左浮动：div { float : left ; }

右浮动：div { float : right ; }

#### 7.1.3 浮动造成什么影响

只要设置了浮动，盒子将脱离普通文本流，下方的内容会自动上移。解决的办法是：使用clear属性清除浮动带来的效果。

### 清除浮动造成的影响

#### 7.2.1 父元素加overflow

overflow : hidden ; 超出部分隐藏

#### 7.2.2 后面元素加 clear

div { clear : left ; } 清除左浮动

div { clear : right ; } 清除右浮动

div { clear : both ; } 清除左右浮动

#### 7.2.3 父元素尾部加::after

after伪元素实现元素末尾添加一个看不见的块元素（Block）

### 超出处理

#### 7.3.1 超出部分隐藏

语法：

overflow : visible | auto | hidden | scroll

overflow : hidden ; 超出部分隐藏

#### 7.3.2 超出部分出现滚动条

设置textarea对象为hidden值将隐藏其滚动条。

### 显示与隐藏

#### 7.4.1 隐藏元素

语法： display : none

#### 7.4.2 以块元素显示

语法： display : block

#### 7.4.3 以行内元素显示

display : inline

以行内元素的方式显示DIV标签

#### 7.4.4 以行内块显示

display : inline-block;

以行内块元素的方式显示DIV标签

### 定位

#### 7.5.1 定位方式

1）相对定位 position : relative

根据父辈定位进行偏移，一直找到浏览器为止

2）绝对定位 position : absolute;

根据当前的位置进行偏移

3）固定定位 position : fixed;

注意：成固定定位不随页面内容滚动

常用：广告位 回到顶部

#### 7.5.2 位置控制

上 top : auto | number

右 right : auto | number

下 bottom : auto | number

左 left : auto | number

1）当对象的定位方式是相对定位的时，以当前对象为基准，表示偏移量。

2）当对象的定位方式是固定定位时，以浏览器为基准，表示与浏览器边框之间的距离。

3）当对象的父对象有设置定位，以父对象为不是static基准,，表示与父对象之间的距离。

#### 7.5.3 层次

层次 z-index

作用： 检索或设置对象的层叠顺序。

语法： z-index :auto| number

与 position搭配使用。只有当DIV定位方式设置成relative或 absolute 或 fixed时，该属性才起作用，层次值高的会遮住层次值低的对象。越大越在上面

示例：

div { position: absolute; z-index:999; }

#### 7.5.4 子绝父相

父元素用相对定位

子代用绝对对位

## 表格列表手型

### 细边框

语法： border-collapse : collapse

参数：

collapse如果可能，边框会被合并为一个单一的边框。

separate默认。边框会被分开。

### 清除引导符

list-style : none ;

### 手型

cursor : pointer ;

注： 常在非超链接的对象上设置成手型，配合JavaScript效果，让用户认为是可以点击的模块。

# JavaScript交互

## Js概念

### 什么是js

基于对象和事件驱动，并且具有相对安全性的解释型的客户端脚本语言

### Js的作用

网页的交互(验证，动态，幻灯片，动效)操作都是由JavaScript完成

### Js的版本

Es5与es6的区别：

ES5作为ECMAScript第五个版本（第四版因为过于复杂废弃了），浏览器支持情况可看第一副图。

什么是ES6在保证向下兼容的前提下，提供大量新特性：

1. Symbol是一种基本类型
2. Promises是处理异步操作的对象
3. 箭头函数 Arrow functions
4. 块级作用域 关键字let, 常量const
5. 对象字面量的属性赋值简写

## 基本语法

### 四种编写js的方法

script标签的规范位置二种：（常用2）

1) 所有的<script>标签都应该放在页面的<head>标签中；

2) 为了让内容优先加载显示，提高用户体验，也可以把<script>标签放在</body>结束标签之前。

#### 2.1.1内部JS

Javascript可以写在script标签内（内嵌方式）

如：

<script type="text/javascript">

alert('这是Javascript程序');

</script>

#### 2.1.2 外部JS

Javascript可以写在单独的文件中（外联方式）

1.创建以.js结尾的文件并且写入Javascript代码

2.hello.js(文件名任意):

alert('这是Javascript程序');

3.在html文件中使用Javascript代码文件引入

<script type="text/javascript" src=’hello.js’>

</script>

#### 2.1.3 行内脚本

JavaScript可以写在HTML标签的事件属性中

onclick点击事件（用户点击），属性值：点击时，执行的js代码。

<button onclick=” alert(250); ”>点击我</button>

#### a标签href属性

Javascript可以A标签的href属性中

a标签：超链接，主要用于跳转到href属性设置连接地址；

<a href=”http://www.itsource.cn”>源码时代官网</a>

a可以设置href值前缀以“javascript:”（前缀其实叫协议,例如：thunder://xxxfasdfasd,迅雷的协议，点击这样的连接打开的不是页面，而是迅雷）开头，当用户点击的时候就会告诉浏览器这不是一个连接地址，而是需要执行一段javascript代码。

如：

<a href=”javascript: alert(‘搞啥子？？？’);”>点我</a>

### 输出方式

#### 2.2.1 网页输出

①　输出纯文本：

document.write("hello world");

②　输出html代码：

document.write("<h1>hello world</h1>");

#### 2.2.2 弹出框

alert ('hello word ' );

#### 2.2.3 控制台输出

console.log ('hello word');

### 注释

注释是不需要执行，主要是对程序进行文字说明和解释。

#### 2.3.1 单行

// 注释的内容

#### 2.3.2 多行

/\*（快捷键：ctrl+shift+/）

注释的内容。

注释的内容

\*/

### 变量

#### 2.4.1 定义变量

变量是：用于存储在内存中中变化的数据；

ES5：var 变量名 = “变量值”;

ES6：let 变量名 = “变量值”

const 变量名 = “变量值”

**ES5 ES6 区别**

1. var 可以重复定义，后面定义时var没有意义

2. let 不能重复定义，否则报错，有块级作用域，没有变量提升

3. const 定义常量，不能重复定义，不能再次赋值，否则报错

#### 2.4.2 命名规则

1)变量由字母、数字、字符组成（\_，$）

2)首字母不能以数字开头；

3)变量一般首字母小写；

4)变量名尽量使用有意义的单词；

5)变量如果是多个单词，一般采用驼峰命名法或使用\_间隔命名法

HelloWord 大驼峰

helloWord 小驼峰

hello\_word 间隔命名法

#### 2.4.3 使用变量

1)变量可以在内存中通过名字保存一个数据，然后通过变量的名称来多次使用该数据。

2也可以通过变量的名字让变量的值传递给其他的程序使用。

#### 检测变量类型

typeof （变量名）

### 标识符命名

#### 2.5.1 可用字符

标识符可以由：

任意顺序的大小写字母、数字、下划线(\_)和美元符号($)组成，

注意：

1.但标识符不能以数字开头，

2.不能是JavaScript中的保留字或关键字。

3.区分大小写

4.见名知义

5.注意关键字与保留字

#### 2.5.2 关键字与保留字

关键字：

编程语言中有特殊含义的单词

保留字：

现在JavaScript版本中没有使用这些单词，供以后版本使用，但是也不准我们使用。

提示：关键字和保留字不需要特别记忆，慢慢来积累和记忆。

## 数据类型

### 基本类型

#### 3.1.1 数字

Number--(数字)



整数（int）

小数（float）

特殊数值：NaN(计算错误，类型转换失败)

#### 3.1.2 字符串

字符串(string)

使用 “ ” 或者是 ‘ ’ 表示的数据是字符串类型的数据。

转义字符（\）：

\’ 表示单引号 \” 表示双引号

\n 表示换行符 \\ 表示反斜线 \ 号本身

#### 3.1.3 布尔值

布尔类型（boolean）

boolena类型的值只有两个：

true 真

false 假

#### 未定义

未定义(undefined)

①　注意：undefined代表，一个变量已经定义但是未赋值，默认值是undefined.

②　如果我们直接使用未定义的变量将会报错！（控制台查看）

#### 空

空(null)

数据类型null只有一个值null。

可以通过给一个变量赋 null 值来清除变量的内容。

#### symbol()（es6新增)

### 引用类型

Array 数组

Function 函数

Object 对象（一切皆对象）

### 类型转换

#### 3.3.1 自动转换（隐式转换）

在运行过程中根据编程语言的运算的语意环境,Javascript会自动进行类型转换.

#### 3.3.2 强制转换

(1)parseInt（变量名）：将变量转换为整数

把小数和不是数字的部分去掉；

注意：1)只有首字符开始的连续数字转换后得到值，其他得到NaN

2)除了字符串的其他类型转换后都得到NaN

(2)parseFloat(变量名)：将字符串转为小数

注意：1)只有首字符开始的连续数字转换后得到具体值，其他得到NaN

2)除了字符串的其他类型转换后都得到NaN

3）如果小数后面的值是有效值，转换后就保留，否则就删除

(3)Boolean(变量名)：转换为布尔值，true/false;

注意：除了0 "" '' undefined NaN null 得到fasle ， 其他的都是true

(4)String(变量名):转换为字符串；

## 运算符

### 算术运算符

+ - \* / %（取余）

++ --：

++,-- 分为前,后.

++放在前面: 先自身+1运算,再和其他的运算数计算.

++放在后面: 先和其他的运算数计算,再自身+1运算

### 比较运算符

> < >= <=

==（等于）: 比较值的大小是否相等，不比较值的类型 （比较大小）

===(全等于): 既比较值的大小是否相等，又比较值的类型.（比较大小，类型）

!== 不全等

!= 不等

### 三元运算符

条件运算符：

(expr1) ? (true执行语句1) : (flase执行语句2)，

a>b? a : b

### 逻辑运算符

&&（与） 同时满足

||（或） 其一满足

! （非） 非（取反）

### 字符串拼接符

+

### 赋值运算符

=

+=： a+=b  a=a+b;

-=: a-=b  a=a-b;

\*=: a\*=b  a=a\*b;

/=： a/=b  a=a/b;

%=: a%=b  a=a%b;

## 流程控制语句

### 条件分支

#### 5.1.1 单分支

语法结构：

if(条件表达式){

//语句1

}

#### 5.1.2 双分支（常用）

语法结构：

if(条件表达式){

//条件成立的需要执行的语句

}else{

//不成立的需要执行的语句

}

#### 5.1.3 多分支

if(条件1){

//语句...

//语句...

}else if(条件2){

//语句...

//语句...

}else if(条件N){

//语句...

//语句...

}else{

//条件都不满足的语句

}

注意：

如果条件有重合，最苛刻的条件放在最前面。

### 状态分支

switch(表达式){

case 表达式1:

语句1;

break;

case 表达式2:

语句2; ...

break;

case 表达式n:

语句n;

break;

default:

语句n+1;

break;

}

switch语句：“如果switch中的表达式等于case 后面的个值(value)，则执行后面的语句，截止break；或switch的右花括号（}）”。

default:默认语句，如果switch中的表达式与case后面的值都不同等，那么默认执行default后面的语句，default不是必须的。

break：关键字会导致代码执行流跳出switch语句。

switch与if类似，但是switch比较适合用于对某个状态值的判断，而if则是针对稍复杂的判断，但是其实switch与if可以相互转换。

### for循环(常用)

for ( 初始化循环变量表达式 ; 循环条件判定表达式 ; 循环变量更新表达式 ) {

循环体。。。

}

初始化循环变量表达式：记录循环的次数；

循环条件判定表达式：判断是否执行循环，只有条件满足才进行循环；

循环变量更新表达式：修改条件表达式中的变量；

循环体：需要多次执行的语句

### while循环

while ( 条件表达式 ) {

循环体语句

}

while：循环执行代码，如果while()条件表达式为真，将执行while花括号{}中的语句，直到条件不为真时，才会终止while运行。

while：循环执行代码，如果while()条件表达式为真，将执行while花括号{}中的语句，直到条件不为真时，才会终止while运行。

### 中止循环

break: 跳出当前循环

continue: 终止本次循环,继续下次循环.

## 数组

### 什么是数组

数组是一组数据有序排列的集合。

### 数组的定义

字面量： var arr3 = [1,3,5,7,9];

实例化： var arr2 = new Array(1,25,37,46,95);//定义了一个具有5个元素的数组

### 数组的使用

语法：

数组变量名 [ 索引值 ]

例如：arr2 [ 1 ] 获取数组的指定取值

console.log ( arr2 [ 1 ] );

### 新增和修改数组元素

语法：

数组变量名称 [ 索引 ] = 值

### 删除数组元素

语法：

delete 数组变量名 [ 索引 ]

如：

delete arr[2];

清空数组：

arr = [ ];

或

arr.length=0;

### 数组的方法

#### 6.6.1 连接多个数组

. concat (arrayX,arrayX,......,arrayX)

#### 6.6.2 元素拼接成字符串

数组变量 . join( ' 用什么符号分割 ' )

把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。

#### 6.6.3 弹出最后一个元素

数组变量 . pop( ) （出栈）

删除并返回数组的最后一个元素

#### 6.6.4 在最后新增一个元素

. push（新元素1，新元素2 ） （入栈）

向数组的末尾添加一个或更多元素，并返回新的长度。

#### 6.6.5 删除、新增、替换元素

.splice(index,howmany,element1,.....,elementX)

index 必需。规定从何处添加/删除元素。

该参数是开始插入和（或）删除的数组元素的下标，必须是数字。

howmany 必需。规定应该删除多少元素。必须是数字，但可以是 "0"。

如果未规定此参数，则删除从 index 开始到原数组结尾的所有元素。

element1 可选。规定要添加到数组的新元素。从 index 所指的下标处开始插入。

elementX 可选。可向数组添加若干元素。

删除元素，并向数组添加新元素。

#### 6.6.6 删除第一个元素

. shift ( ) 删除第一个元素，将数组当成队列进行操作，出队

#### 6.6.7 在最前新增一个元素

. unshift ( ) 在第一个位置添加一个元素，插队

#### 6.6.8 反转数组

. reverse ( )

颠倒数组中元素的顺序。 但该方法不会创建新的数组

## 函数

### 函数的定义

#### 7.1.1普通函数与匿名函数

声明式与表达式：

（普通函数）声明式：全局可调用

（匿名函数）表达式：在函数之后才能使用

### 函数调用

函数名 （ 函数值 ）；

### 函数参数

#### 7.3.1 形式参数

写在函数定义中,出现在函数名的 ( ) 里面

function 函数名字(形式参数1,形式参数2){

功能代码…

}

#### 7.3.2 实际参数

调用函数时传入的参数数据叫做实际参数(实参)

function 函数名( ) {

函数体...

}

函数名（ 实参 ）；

执行过程: 实际参数分别赋值后形式参数后,方法才执行.

#### 7.3.3 参数默认值

function 函数名( 形式参数=默认参数 ) {

函数体...

}

### 函数返回值

#### 7.4.1 return 值;

如果函数执行参结果的数据需要被其他程序所使用. 可以通过return关键字标记函数执行的结果(返回结果).

function 函数名(形参1,形参2){

//功能代码.

return 返回的值;

}

### 变量的作用域

#### 7.5.1 全局作用域

全局变量：函数外定义的变量，整个程序中都可以使用。

#### 7.5.2 局部作用域

局部变量：在某个特定的函数中定义的变量，称之为局部变量，只能够在函数内部使用

如果不加 var 自动提升为全局变量

### 递归函数

#### 7.6.1 什么是递归函数

在函数体内对函数自身调用！

#### 7.6.2 递归的特点

入口 : 什么时候再次调用当前函数

出口： 什么时候不再调用当前函数

## 对象

### 什么是对象

class：类就是指一类事物的共有特性和行为的抽象化描述，比如：人类，狗类，动物类，车类

对象就是类实例化出来的实体，具备类描述的特征（属性）和行为（方法）

注意：一切皆对象

### 定义对象

#### 8.2.1 字面量

字面量方式创建对象：

var 对象名={属性名 : 属性值, 方法名 : function(){

方法的功能代码

}};

#### 8.2.2 实例化

实例化方式创建对象

var 对象名=new Object(); //创建空对象

对象.属性名=值; //新增属性

对象.方法名=function(){ //新增方法

}

### 使用对象

#### 8.3.1 属性

对象.属性

对象['属性']

#### 8.3.2 方法

对象.方法名()

对象['方法名']()

### 对象方法中的this

this关键词：

1. 什么是this？

this是JS中的关键词，在不同的环境中代表不同的对象

2. 方法中的this

原则：

谁调用函数，函数中的this就指向谁

方法中的this指向对象本身

### 遍历对象

for in

遍历对象：

for(var i in 对象){

属性名：i

属性值：对象[ i ]

方法：对象 [ i ] ( )

}

### 内置对象

#### 8.6.1 字符串对象

获取字符串长度

. length 【重点】长度属性

查找子字符串位置

.charAt（ 索引 ） 取出指定下标的字符

替换

. replace（ ‘ 要替换字符 ’ ） 【重点】替换字符串，

默认只能找到一个，如果要找到全部就必须用正则表达式

如：

a.replace(/l/g,'A')

截取

. substr (下标 , 长度)

从指定的位置开始截取指定长度的字符串

拆分成数组

数组变量 . join( ' 用什么符号分割 ' )

转大小写

. toLowerCase 变小写

. toUpperCase 变大写

#### 8.6.2数字对象

保留小数位数

Math . toFixed(小数点位数)

设置数字的小数点个数，如果没有小数就用0填充

#### 8.6.3 数学对象

向上取整

Math . ceil 【重点】向上取整，通常用于实现客户端分页

向下取整

Math . floor 【重点】向下取整

随机数

Math . random 【重点】随机数，得到一个0-1之间随机小数，包0不包1

#### 日期对象

创建日期对象【重点】

1) 当前日期时间对象

var d= new Date();

2) 按指定时间创建日期对象

var d=new Date(指定日期时间); //必须要符合日期时间格式才能识别

日期时间格式： '年/月/日 小时:分钟:秒'

3) 按指定的参数创建日期对象

var d=new Date(年,月,日,时,分,秒);

注意：时,分,秒可以省略， 月份从0开始计算

获取时间戳

. getTime ( ) 获取指定日期的时间戳

返回本地时间格式

. toLocaleString ( )

如：

var d=new Date();

console.log( d . toLocaleString() );

## Bom

### 什么是BOM

浏览器对象模型，让JS可以操作浏览器，也不是语法，是一套接口

### window对象

#### 9.2.1 属性

. name 窗口的名字

. innerHeight 窗口的内部高度

. innerWidth 窗口的内部宽度

. scrollY 【重点】浏览器纵向滚动条的滚动位置

. onscroll 【重点】浏览器滚动事件

#### 9.2.1 方法

. alert ( ) 【重点】弹出一个警告框

. confirm ( ) 弹出一个确认框，返回用户选择的值

. prompt ( )弹出一个输入框，返回用户输入的值，返回的是一个字符串

. scroll ( x , y ) 【重点】让浏览器滚动条滚动的方法

. setInterval ( ) 【重点】设置一个周期定时器，到时间就自动执行，可以执行多次。 返回一个定时器序号作为定时器名称，可以被清除

. setTimeout ( )设置一个超时定时器，到时间执行一次就自动结束，可以执行1次

. clearInterval ( 定时器变量名 ) 【重点】清除周期定时器

. clearTimeout ( ) 清除超时定时器

#### 9.2.1 子对象

文档对象 document

屏幕对象 screen对象（屏幕对象）

width 屏幕宽度（分辨率x轴）

height 屏幕高度（分辨率y轴）

screen . width

screen . height

地址栏对象

location对象【重点】

. hash URL地址后面 #后的部分

. href 当前页面完整的URL地址，赋值给它就能实现网页的跳转

Location .hash

location .href

历史记录对象：

history对象（历史记录对象）

. back ( ) 返回上一次访问过的网页

. forward ( ) 进入后一次访问的网页

. go ( n ) n可以是正数（前进forward），也可以是负数（倒退back）

history . back ( )

history . forward ( )

浏览器信息

navigator对象

. userAgent 返回浏览器的基本信息，可以判断用户的浏览器和操作系统

navigator . userAgent

## DOM

### 什么是DOM

DOM：文档对象模型，让JS可以操作HTML和CSS， 不是语法，是一套接口

### 节点树

#### 10.2.1 根节点

根节点：html标签

#### 10.2.2 元素节点

元素节点：每一个HTML标签都是一个元素节点 element

#### 10.2.3 文本节点

文档节点：一个文档就是一个节点，document

#### 10.2.4属性节点

属性节点：HTML标签的属性 attribute

### 节点操作

#### 10.3.1 获取节点

通过ID获取【重点】. document.getElementById ( ' id值 ' );

注意：在高级浏览器中，ID可以不需要获取，直接当对象使用（不建议在正式项目中使用，测试时可以使用）

通过标签名获取【重点】. document.getElementsByTagName( ' 标签 ' );注意：返回数组

通过class类名获取 . document.getElementsByClassName ( ' 类名 ' ); 注意：找到所有符合要求的元素，放在一个数组中，数组是一个伪数组

通过name获取 document.getElementsByName ( ' 名称 ' );

注意：返回数组

Html5新方法：

【重点】document.querySelector ( ' 选择器 ' );

选择满足选择器的标签，只能匹配到第一个

选择器里就是CSS的选择器

【重点】document.querySelectorAll ( ' 选择器 ' );

选择所有符合要求的标签，返回一个数组

获取节点名称 .nodeName

获取body

document . body => body标签

document . documentElement => html标签

#### 10.3.2 新增子节点

创建节点：

在节点树中添加一个新节点

1. innerHTML新增节点【重点】

节点 . innerHTML+= '<strong>LOGO</strong> ' ;

2. 创建和添加节点

1) 创建节点

document . createElement ( ' 标签名 ' );

注意：创建节点是在内存中创建一个新标签

2) 添加节点

父标签对象.appendChild ( 节点 );

#### 10.3.3 删除节点

父标签对象 . removeChild ( 节点 ); 删除子节点

节点 . remove ( ); 删除自己，删除当前节点

#### 10.3.4 属性操作

节点 .属性

如：

var s=document.createElement('strong');

s . innerHTML='LOGO';

s . id='myid';

s . className='myclass';

#### 10.3.5 值操作

. innerHTML属性

获取或设置标签中的【HTML内容】

.innerText属性

获取或设置标签中的【普通文本】

.value属性

表单元素才有value值，.value属性只针对表单元素

所有HTML标签上的属性，都可以当前JS元素对象的属性使用

#### 10.3.6 样式操作

标签对象.style.样式属性 =

标签对象.className=

### 事件

#### 10.4.1 常用事件

常用事件【重点】：

1. 鼠标类

onclick 点击事件

onmouseover 鼠标移过

onmouseout 鼠标移出

2. 键盘类

onkeydown 按键按下

onkeyup 按键弹起

3. 表单类

onfocus 获取焦点

onblur 失去焦点

onchange 表单元素值改变事件

oninput 输入值时触发

onsubmit form表单提交事件

4. 窗口类

onload 窗口加载成功时

onscroll 窗口滚动时触发

#### 10.4.2 事件四要素

事件的四要素【重点】

事件源：事件发生的标签

事件的名称：也称为事件类型，是指发生了一件什么样的事件

事件处理函数：事件发生是要执行的操作，一般来说就是一个函数

事件对象：事件对象就是指事件发生时自动创建的对象，包含事件本身具备的信息

#### 10.4.3 注册事件的三种方式

在标签上绑定事件

<标签名 事件属性= " 事件处理函数 ( ) "></标签名>

事件属性绑定事件【最常用】

事件源.事件类型=事件处理函数;

h2.onclick=function(){

}

事W3C标准方法绑定事件

事件源.addEventListener(事件类型, 事件处理函数);

h3.addEventListener ( ' click ' , function ( ) { } ) ;

#### 10.4.4 事件对象

1）获取事件对象：

事件处理函数第一个参数（e）

事件源 . onclick=function ( e ){

e就是事件对象

}

注意：在火狐浏览器中，只支持这种方式获取事件对象

event对象

window . event

注意，只有在事件处理函数中才有事件对象

2）事件对象属性

事件目标

.target 事件发生的目标，就是事件源

按键键位

.which 鼠标或键盘的按键键位，鼠标是 1左,2中,3右。键盘事件时与keyCode相同

.keyCode 键盘键位

点击位置

x 鼠标点击时的x位置，相对于浏览器内容的左上角

y 鼠标点击时的y位置，相对于浏览器内容的左上角

#### 10.4.5 阻止浏览器默认行为

return false

事件处理函数的返回值：

目的：阻止或允许浏览器默认行为

return false; //阻止

return true; //允许，不返回也是允许

event.preventDefault ( )

e.preventDefault ( ) 阻止浏览器的默认行为，与 return false 效果相同

#### 10.4.6 事件冒泡机制

冒泡：冒泡是事件的一种默认机制，事件在子元素触发时，父元素也会被触发，一直向上直到根节点

捕获：捕获也是事件的一种机制，事件触发时首先触发在根元素，再逐层向下捕捉，直到事件源

#### 10.4.7 事件委派

事件委派：

1. 为什么要用事件委派？

因为网页在与用户交互过程中可以会产生新元素，会错过加载页面的事件绑定，所以需要事件委派

2. 什么是事件委派？

事件委派是将事件委托给网页加载时就存在的父元素，

事件触发时，由于事件的冒泡机制，点击子元素父元素也会触发，触发委派的事件

委派的事件触发时，通过事件对象反查事件源，实现目标

# JAVA Query

## Java query概念

概念： jquery是一个快速比较小的一个js库

库： 别人已经封装好了的方法或功能，后期不需要自己来写，直接使用即可

JQuery的特点: (写得少，做得多)

1.选择器比较丰富

2.链式操作DOM

3.事件绑定机制

4.兼容性比较好

链式操作: jq对象.事件1(事件处理函数).事件2(事件处理函数).事件3(事件处理函数)

## Java query版本

jquery的版本说明：

1.x 支持IE6~8 兼容性比较好

2.x 不支持IE6~8 体积小

3.x 支技IE9以上版本 炫酷的网站

实际中：

学校，医院，政府===》使用 1.x

其它只关注炫酷效果的网站，可以在2.x或3.x选用

学习：主要是讲1.x的版本

引入检测：

如果在引入后通过console.log($) 有内容，表示引入成功

如果 提示 $ is not defined表示没有引入成功

## jquery的引入及使用说明:

$和jQuery是相同的一个东西,都表示jquery的标识

如: $('div') === jQuery('div')

使用大前提:

1. jquery.js必需引入===>检测环境使用是引入成功

2. 选中的元素必需在库后面或 元素加载成功后在执行相应的jq代码

3.如果jq没有生效.需要通过下列的方法来检测

方式一: 把jq代码放在body结束标签之间,内容之后

方式二: 使用jquery的页面加载事件: 相当于 window.onload=function(){}（原生js）

1. $(function(){

执行的jq代码;

});

2. $(document).ready(function(){

执行的jq代码;

});

3. jQuery(function(){

执行的jq代码;

})

## 选择器

### 基本选择器===重点

1. id选择器

作用: 选中指定id值的标签

语法:

$('#ID的值') 或 jQuery('#ID的值');

2) class选择器

作用: 选中指定class值的标签

语法:

$('.class的值') 或 jQuery('.class的值');

3) 元素选择器:

作用: 选中网页中,所有指定标签名的标签

语法:

$('标签名') 或 jQuery('标签名');

4) 分组选择器(掌握))

作用: 把指定的效果附加多个选择器上

语法: $('#id,.class,标签名,.....')

或jQuery('#id,.class,标签名,.....')

### 过滤选择器

概念: 从获取到的元素集合中进行筛选操作(获取需要的, 删除不需要)

#### 4.2.1 基本过滤选择器

:first: 匹配第一个元素

:last 匹配最后一个元素

:even 匹配索引为偶数的元素,索引从0开始

:odd 匹配索引为奇数的元素,索引从0开始

:eq 匹配索引值等于index的元素,索引从0开始

:gt 匹配索引值大于指定索引

:lt; 匹配索引值小于指定索引

:header 匹配h1,h2,....

:animated 匹配所有在执行动画的元素

#### 4.2.2 内容过滤选择器

:contains(文本) 匹配包含指定文本的元素

:empty 匹配没有任何元素的标签(空格或回车也不算)

:has(选择器) 匹配含有指定选择器的元素

#### 4.2.3 可见性过滤选择器

隐藏元素: :hidden

display:none;

visibility:hidden; ---->不能使用此属性

input type="hidden"

显示元素: :visible

#### 4.2.4 属性过滤选择器(了解)

[attribute] 通过指定标签上的属性名来进行匹配

[attribute=value] 通过指定标签上的属性名=指定的值来进行匹配

#### 4.2.5 子元素过滤选择器

:first

:first-child

:first-of-type 同类型

:nth-child(值) 值:可以是数字,表达式,英文中的偶数或奇数

### 表单选择器

:checked 表示选中单选或多选,选中的状态

:selected 表示下拉框中默认选中的标签

区别:

$('input'): 选中表单中所有的input标签

$(':input') 选中表单中的所有控件(input button select textarea)

### CSS样式操作

js:

获取样式: getCss()

设置样式: js对象.style.属性名=值;

jq:

获取样式: jq对象.css('属性名');

设置样式:

单个属性: jq对象.css('属性名','属性的值');

多个属性: jq对象.css({'属性名1':'属性的值1','属性名2':'属性的值2',....})

### 属性操作

js:

标准属性:

获取属性: js对象.属性名

设置属性: js对象.属性名=值;

自定义属性:

获取属性: js对象.getAttribute('属性名');

设置属性: js对象.setAttribute('属性名','属性的值');

jq:

标准属性: prop()

获取属性: jq对象.prop('属性名')

设置属性: jq对象.prop('属性名','属性的值')

自定义属性:attr()

获取属性: jq对象.attr('属性名')

设置属性: jq对象.attr('属性名','属性的值')

注意:

1.attr()可以获取或设置标准属性和自定义属性

2. 在项目中一般使用attr()---除boolean属性以外

3.boolean属性一般常用prop()来操作

4.boolean属性: 属性名等于属性值

如: checked selected readyOnly disabled ....

# Ajax

## Ajax基础

### Ajax基本概念：

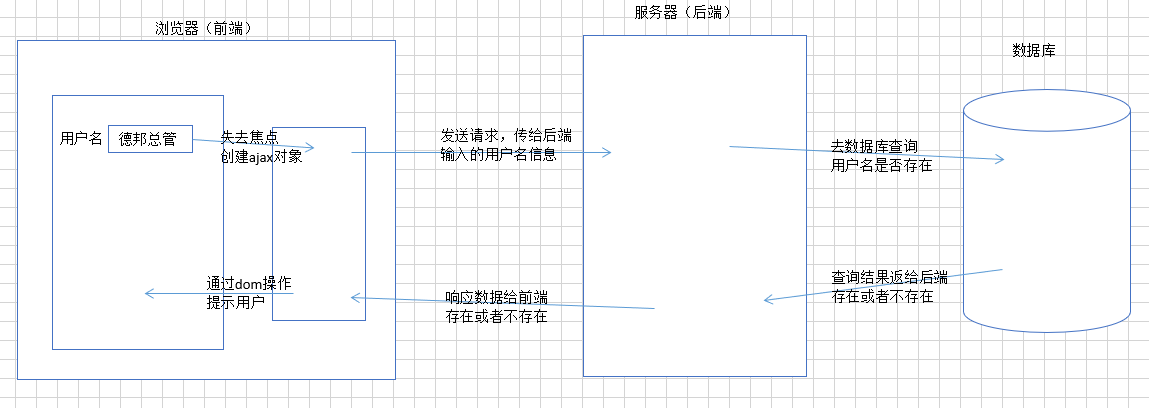
浏览器创建ajax对象，ajax对象向后端发送请求，后端接收请求响应数据给前端，ajax对象接收响应数据通过dom操作更新视图，整个过程浏览器“不刷新”。

### ajax优缺点

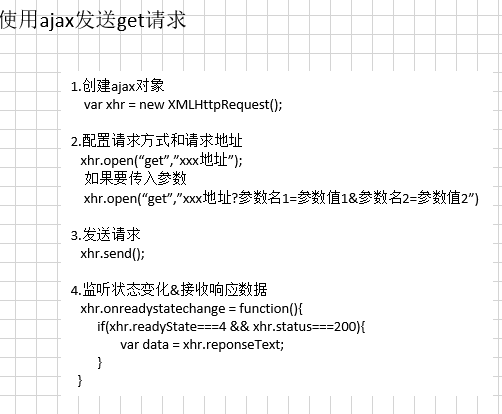
优点: 在不刷新页面的情况下，进行局部的数据交换，更新视图，用户体验好。

缺点:对SEO（搜索引擎优化）不友好.

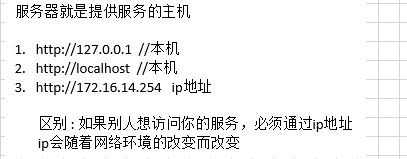
### ajax工作原理(通过一个注册流程剖析)



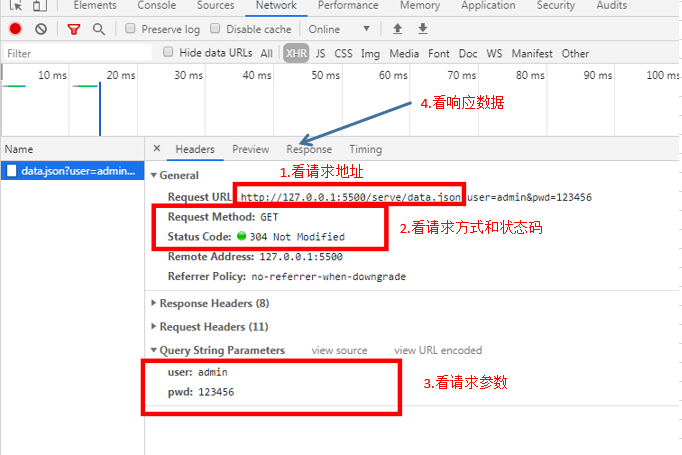
### 使用ajax发送get请求



### 服务器



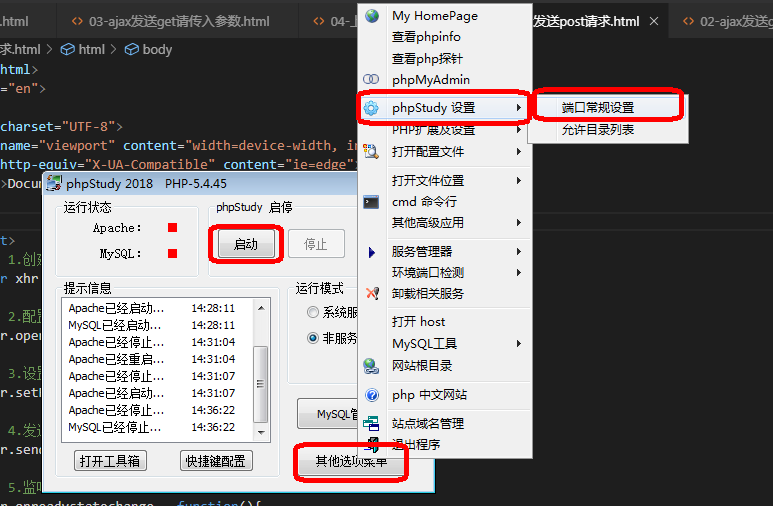
### 如何查看ajax请求



### 使用post发送ajax请求



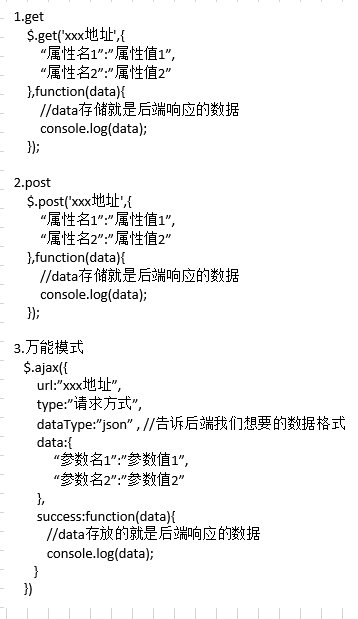
### phpStudy设置





### 使用jQuery发送ajax的三种不同方式

使用jQuery发送ajax的三种不同方式



### Json

1. json是前后端数据传输的一种格式

2. 是一个对象，以键值对的方式存在，如果是以.json结尾的文件，键(key)一定要以“双引号”括起来。

3. 大写的JSON是一个js内置对象，有两个方法:

a. parse 把字符串格式的json数据转换为对象

b. stringify 把json对象转换为字符串.

## Ajax与前端模板

### MockJS实现前后端分离

总结起来就是在后端接口没有开发完成之前，前端可以用已有的接口文档，在真实的请求上拦截ajax，并根据mockjs的mock数据的规则，模拟真实接口返回的数据，并将随机的模拟数据返回参与相应的数据交互处理，这样真正实现了前后台的分离开发。与以往的自己模拟的假数据不同，mockjs可以带给我们的是：在后台接口未开发完成之前模拟数据，并返回，完成前台的交互；在后台数据完成之后，你所做的只是去掉mockjs：停止拦截真实的ajax，仅此而已。

### 官方网站和实用实例

<https://github.com/nuysoft/Mock/wiki>

<http://mockjs.com/examples.html>

### Mockjs的安装和使用

#### 安装

bower install mockjs (库文件的使用方式)

npm install mockjs (nodejs项目中使用)

#### 使用

引入



var Mock=require('mockjs');

使用模板：

'users|10': [

{

// 属性 id 是一个自增数，起始值为 1，每次增 1

'uid|+1': 1,

'rndID':'@guid',

'sex':/[男女]/,

'age|20-30':1,

'username':'@first@last',

'truename':'@cname',

'email':/^[a-zA-Z0-9]{6,12}@(qq\.com|163\.com|sina\.com\.cn)$/

}

]

### 模板的概念

模板，通常是指将一个事物的结构规律予以固定化、标准化，它体现的是结构形式的标准化。 今天我们说的模板是指WEB前端模板，将HTML网页划分为固定结构与数据

# html5

## html5的新特性

1）语义标签header footer nav等

2）websocket 协议 双工通讯 聊天室

3）本地存储

4）多媒体

5）绘图 canvas

6）多线程 webworks

7）设备访问 媒体查询

8) flex布局 弹性盒子布局 伸缩布局

## Html5兼容性

引入html5shiv.js

### html4和html5的文档结构区别（掌握）

html5 声明 <!DOCTYPE html>

html4 声明 很长 基于strict.dtd作为文档解析

### html5语义标签

header 头部

nav 导航

section 区域

mian 主体

aside 侧边栏

article 文章

footer 页脚

### .H5 form表单(了解)

1) input type

color 颜色选择器

date 日期选择框

Datetime 日期时间选择框

datetime-local 日期和时间 (无时区)选择框

month 月份选择器

time 时间选择框

week 周和年选择框

email 电子邮件

number 数值的输入域

range 区间选择器

search 搜索框

tel 电话号码

url url地址

2） 新标签

datalist 效果相当于下拉框，可以输入，输入的时候会有个模糊匹配

output 语义化标签 输出

3） 新属性

作用于form

novalidate 取消验证

作用于input

autocomplete 自动补全

autofocus 聚焦

form 指定form外的input框跟随form一起提交

multiple 多选

pattern 正则表达验证

placeholder 提示信息

required 必填项

formnovalidate 取消验证 作用于提交按钮

### 自定义属性（重点）

html4

自定义属性名和原生属性名容易混淆

获取: el.getAttribute('属性名')

设置: el.setAttribute('属性名','属性值')

H5

自定义属性都以data- 开头，容易区分

获取：el.dataset['属性名']

设置：el.dataset['属性名'] = '属性值'

jquery

获取 $('el').data('属性名')

设置 $('el').data('属性名','属性值')

### 其他属性（了解）

contenteditable 是否可以编辑

tabIndex 聚焦顺序

### 本地存储（重点）

**cookie**

容量 4kb ，和服务端是有交互的(每次请求客户端会带上cookie信息) ，session基于cookie实现

**sessionstorage**

容量 5M

获取：sessionstorage.getItem('属性名')

设置：sessionstorage.setItem('属性名','属性值'); //属性值是string类型

删除：sessionstorage.removeItem('属性名');

清除: sessionstorage.clear();

生命周期：sessionstorage存在于页面，如果页面关闭，就会消失。

**localstorage**

容量 5M

获取：localstorage.getItem('属性名')

设置：localstorage.setItem('属性名','属性值'); //属性值是string类型

删除：localstorage.removeItem('属性名');

清除: localstorage.clear();

生命周期：永久存储，除非主动删除。

# Css3

## Css标准

最新css的标准

## 与HTML5有什么关系

html5是html的最新标准

css3是css的最新标准，两者没有关系；但可以组合使用。

## css兼容性写法

-webkit-border-radius chrome ,safari

-ms-border-radius ie

-moz-border-radius 火狐

-o-border-radius opera手机

border-radius

## css3新特性

选择器

文本颜色

背景边框

伸缩布局（flex布局）

过度

动画

2D/3D转换

## 选择器（重点）

基本选择器

id id选择器

. 类选择器

el 标签选择器

selector1,...,selectorN 群组选择器

\* 通配(通用)

层次选择器

E F E元素下面的后代元素F

E>F E元素下面的第一层子元素F

E+F 紧贴在E元素之后F元素（同级别 兄弟）

E~F E元素之后所有兄弟元素F（同级别 兄弟）

伪类选择器

动态伪类选择器

E:link a在未被访问前的样式

E:visited a在访问后的样式

E:hover 鼠标悬停样式

E:active 鼠标按下没有释放时候的样式

E:focus 聚焦

目标为类选择器

E:target 鼠标选中的文字样式

结构伪类选择器（重点）

E:first-child 父元素下的第一个子元素

E:last-child 父元素下的最后个子元素

E:nth-child(n) 父元素下的第n个子元素

E:nth-last-child(n) 父元素下倒数第n个子元素

E:first-of-type 父元素下指定类型的第一个子元素

E:last-of-type 父元素下指定类型的最后一个子元素

E:nth-of-type(n) 父元素下指定类型的第n个子元素

E:nth-last-of-type(n) 父元素下指定类型的倒数第n个子元素

E:only-of-type 父元素下唯一该类型的子元素

E:empty 没有子元素

否定伪类选择器

E:not(selector) 除了selector

伪元素

::first-letter 第一个字母

::first-line 第一行

::before ...之前

::after ...之后

::selection 选中文本

属性选择器

E[att] 包含属性名为attr的元素

E[att="val"] 属性名为attr，值为val的元素

E[att~="ccc"] 属性名为attr，值为val，空格分隔

E[att^="val"] 属性名为attr，值以val开头

E[att$="val"] 属性名为attr，值以val结尾

E[att\*="val"] 属性名为attr，值包含val

E[att|="val"] 属性名为attr，值以val开头，分隔符为“-”

## 文本效果

1）文本阴影

text-shadow:x偏移 y偏移 模糊值 颜色

2）文本溢出

text-overflow:clip(截断) | ellipsis（省略号）

3）文本换行

word-wrap:normal(不换行) | break-world(中断单词,强制换行))

## 阴影和边框

1）盒子阴影

box-shadow:x偏移 y偏移 模糊值 外延伸 颜色 insert(内阴影)

2）圆角边框

border-radius: 1~8个值

## css3 flex布局

### felx布局

弹性布局 伸缩布局

### 如何设置为flex布局

display:flex;

### flex布局的几个概念

flex布局有两根轴线，

一根是主轴(默认从左往右)

主轴起点：主轴与左边框的交叉点

主轴终点: 主轴与右边框的交叉点

一根是交叉轴(默认从上往下)

交叉轴起点：交叉轴与上边框的交叉点

交叉轴终点：交叉轴与下边框的交叉点

### flex属性

#### 父容器属性(6个)

1)flex-direction 设置主轴方向

row | row-reverse | column | column-reverse

从左往右 从右往左 从上往下 从下往上

2)flex-wrap 是否换行

nowrap | wrap | wrap-reverse

不换行 换行 换行且反转

3)flex-flow 以上两个属性的复合

flex-flow: <flex-direction> || <flex-wrap>;

主轴方向 是否换行

4)justify-content 设置主轴的对齐方式

flex-start | flex-end | center | space-between|space-around

主轴起点对齐 主轴终点对齐 居中 两端对其 间距等分

5)align-items 设置交叉轴的对齐方式

flex-start | flex-end | center | baseline | stretch

交叉轴起点对齐 交叉轴终点对齐 居中 文字基线对齐 拉伸

6)align-content 设置多根轴线的对齐方式

flex-start | flex-end |center | space-between | space-around |stretch

交叉轴起点对齐 交叉轴终点对齐 居中 两端对齐 间隔等分 拉伸

#### 项目属性(6个)

1）order 排序

值越大越靠后，值越小越靠前 默认值为0

2）flex-grow 按比例放大

正数 按比例放大 默认值为0(不放大)

3）flex-shrink 按比例缩小

正数 按比例缩小 默认值为1

4）flex-basis 基准值

配合关闭放大 和 关闭缩小使用 基准值设置的值就是所占的主轴空间的值

5）flex 以上三个属性的复合属性

flex-grow flex-shrink flex-basis

放大 缩小 基准值

6）align-self 单独的对齐方式

auto | flex-start | flex-end | center | baseline| stretch

默认 交叉轴起点对齐 交叉轴终点对齐 居中 文字基线对齐 拉伸

### web字体

1）常见的格式

.eot .ttf .woff .svg .otf

2) 介绍两个常用图标库

font Awesome

阿里巴巴矢量图标库

### 过渡

transition：样式 执行时间（多少s） 过渡函数 延迟时间（多少s）

过渡函数

linear | ease | ease-in | ease-out | ease-in-out

线性 平滑 由慢到快 由快到慢 有慢到快再到慢

## 变形动画

### 变形函数

#### 1）平移

translateX() //沿x轴平移

translateY() //沿y轴平移

translate(x,y) //沿x和y轴平移

#### 2）缩放

scaleX() //x轴缩放

scaleY() //y轴缩放

scale() //x和y轴同时缩放

#### 3）旋转

rotateX() //绕x轴旋转

rotateY() //绕Y轴旋转

rotateZ() //绕Z轴旋转

rotate() //绕Z轴旋转

#### 4）斜切

skewX() //沿x轴扭曲

skewY() //沿y轴扭曲

skew(x,y) //x和y同时扭曲

### 设置对象原点

transform-origin(x,y) //x横坐标 y纵坐标

### 定义动画

#### from-to

@keyframes 动画名{

from{

//自定义样式

}

to{

//自定义样式

}

}

#### 百分比

@keyframes 动画名{

0%{

//0%的样式

}

20%{

//20%的样式

}

40%{

//40%的样式

}

60%{

//60%的样式

}

80%{

//80%的样式

}

100%{

//100%的样式

}

}

### 执行动画

animation:动画名 执行时间 过度函数 延迟;

### 动画属性

animation-name：动画名

animation-duration：执行时间（多少s）

animation-timing-function：过度函数（linear ease ...）

animation-delay: 延迟时间（多少s）

animation-iteration-count：播放次数 infinite(无限循环)

animation-direction:播放方向（正向或者反向）

normal默认值（正向）

reverse（反向）

alternate（1，3，5...奇数次正向，偶数次反向）

alternate-reverse(奇数次反向，偶数次正向))

animation-fill-mode: 动画停留的样式

none（默认）动画在动画执行之前和之后不会应用任何样式到目标素。

forwards 动画结束后保留结束时的样式

backwards 动画开始时应用from或者0%的样式

both forwards和backwards的复合

animation-play-state：暂停或者播放

paused 暂停

running 播放

### 监听动画事件

1)animationstart 动画开始事件

2)animationend 动画结束事件

3)animationiteration 循环事件

## animate.css动画库

### animate.css使用

1)下载(https://daneden.github.io/animate.css/)和引入

2)找到想要的效果，给目标元素直接加上类名 animated 具体的效果类名

### swiper

是一个滑块效果的框架

基本使用

1)下载(https://www.swiper.com.cn/usage/index.html)和引入

2)写入html

<div class="swiper-container">

<div class="swiper-wrapper">

<div class="swiper-slide">Slide 1</div>

<div class="swiper-slide">Slide 2</div>

<div class="swiper-slide">Slide 3</div>

</div>

<!-- 如果需要分页器 -->

<div class="swiper-pagination"></div>

<!-- 如果需要导航按钮 -->

<div class="swiper-button-prev"></div>

<div class="swiper-button-next"></div>

<!-- 如果需要滚动条 -->

<div class="swiper-scrollbar"></div>

</div>

3)定义容器swiper-container样式

4）初始化swiper

<script>

var mySwiper = new Swiper ('.swiper-container', {

direction: 'vertical', // 垂直切换选项

loop: true, // 循环模式选项

// 如果需要分页器

pagination: {

el: '.swiper-pagination',

},

// 如果需要前进后退按钮

navigation: {

nextEl: '.swiper-button-next',

prevEl: '.swiper-button-prev',

},

// 如果需要滚动条

scrollbar: {

el: '.swiper-scrollbar',

},

})

</script>

swiper animate使用

1)下载（https://www.swiper.com.cn/usage/animate/index.html）和引入

2）

<script>

//Swiper5

var mySwiper = new Swiper ('.swiper-container', {

on:{

init: function(){

swiperAnimateCache(this); //隐藏动画元素

swiperAnimate(this); //初始化完成开始动画

},

slideChangeTransitionEnd: function(){

swiperAnimate(this); //每个slide切换结束时也运行当前slide动画

//this.slides.eq(this.activeIndex).find('.ani').removeClass('ani'); 动画只展现一次，去除ani类名

}

}

})

</script>

3)给目标元素添加类和三个属性

类名：ani

swiper-animate-effect 效果

swiper-animate-duration 执行时间

swiper-animate-delay 延迟时间

<div class="swiper-slide">

<p class="ani" swiper-animate-effect="fadeInUp" swiper-animate-duration="0.5s" swiper-animate-delay="0.3s">内容</p>

</div>

## 响应式网站开发原理

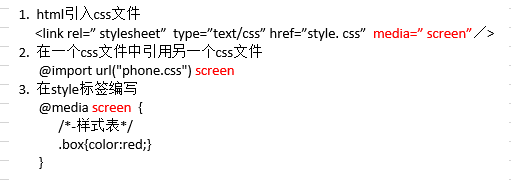
### 媒体类型

1.all 所有的

2.screen 屏幕

3.print 打印机

### 引入方式



#### link和@import区别



### dpr设备像素比

物理像素：实实在在的像素点

设备独立像素：逻辑像素，css的像素

设备像素比dpr = 物理像素/设备独立像素 (得到的值越高越清晰)

### 视口

<meta name="viewport" content="width=device-width, initial-scale=1, minimum-scale=1.0, maximum-scale=1, user-sclable=0">

width 布局视口宽度

device-width 设备宽度

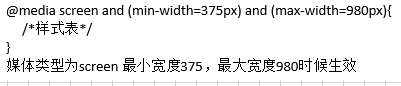
initial-scale 初始化缩放比例

minimum-scale 最小缩放比例

maximum-scale 最大缩放比例

user-sclable 是否允许缩放

### 媒体查询



# Canvas

## Canvas简介

H5的新特性，是一个画布工具。 JS就是我们手中的画笔。

html5 canvas网站

<https://www.html5tricks.com/category/html5-demo/page/11>

### canvas中的属性

设置样式（颜色）

fillStyle 填充颜色设置

strokeStyle 设置笔触的颜色

lineWidth 设置 线条宽度

矩形

rect(x,y,w,h) 创建矩形

fillRect(x,y,w,h) 绘制填充矩形

strokeRect(x,y,w,h) 绘制描边矩形

路径属性

fill() 填充

stroke() 绘制 定义好的路径

beginPath() 起始路径

moveTo() 把笔移动到指定点

lineTo() 绘制从起点到终点的线

closePath() 闭合路径

# echarts 报表

## 百度eCharts的强大

<http://echarts.baidu.com/echarts2/x/doc/example/map3d_sun.html>

## 五分钟上手

1. 在页面中，新建一个盒子（应该默认宽高），用于显示报表内容
2. 引入echart核心的js库
3. 使用javascript，显示报表
   1. 初始化报表组件。
   2. 做报表配置。
   3. 使用配置，来创建报表。

使用方式：

A 通过init方法 找到准备好的容器

B 配置核心参数 var option={核心配置}

C 把核心配置通过setOption配置在实例中

# Less编译

### 编译less 环境安装有2种方式

#### 1）node进行编译

安装node -》 直接下一步就可以了 -》 w10环境下window键+R输入cmd/ 随便在某个文件夹下输入cmd 进入dos命令 -》 npm install -g less 通过npm进行一次全局安装less

在当前文件夹下 lessc \*.less > \*.css

#### 2) 通过vscode插件

搜索easy LESS 插件名 安装

在文件中 首选项 设置 输入easyless 找到setting.json

在后面加这段代码

"less.compile": {

"compress": true, // true => 压缩编译后的css

"sourceMap": true, // true => 自动生成source maps (.css.map files)

"out": "${workspaceRoot}\\dist\\css\\",// false => 关闭less编译功能；

// 开启：'文件地址||文件夹地址' ==>自动输出到指定文件或目录

}

### less 用法

#### 注释

/\* \*/ css注释，编译后，会显示在css文件中

// 代码注释，编译后会直接忽略。

#### 变量

@变量名: 变量值;

#### 定义函数

.函数名(参数1: 默认值,参数2: 默认值){

css样式

}

#### 调用函数

.函数名(参数1,参数2);

#### 用法

.mt(@mt: 10px){

margin-top: @mt;

}

#### 嵌套

dom{

dom子元素{

dom孙子元素{

css

}

}

}

#### 颜色函数(了解)

lighten(@color, 10%); /\* 变亮 \*/

darken(@color, 10%);  /\* 变暗 \*/

saturate(@color, 10%);   /\* 加浓 \*/

desaturate(@color, 10%); /\* 变淡 \*/

mix(@color1, @color2); /\* 混合两种颜色 \*/

# nodeJs基础

## nodeJs定义及安装

可以在电脑上运行js的一个运行环境 （runtime）

### 安装：

下载node 百度node -》 下载 -》 找到电脑对应的操作系统

### 基础命令：

cmd 进入 dos

node -v 查看node版本号

cls 清屏

ctrl+c 结束node命令

node \*.js 运行文件

## 模块化概念

模块化概念 COMMONJS 经典老牌的模块化规范

### 作用：

1. node解决了浏览器 变量污染的问题

2. 每一个js都是一个独立的模块，所有变量都是私有变量,要使用需要暴露。 module.exports.变量名=变量名;

3. 接收暴露出来的模块。 require(); // 用来接收暴露的模块

4. 每一个模块都是独立，可复用性更高，减少代码冗余

### 模块化的优势

1. 彻底解决浏览器 变量污染的问题

2. 模块只管理自己内部，与外部无关

### nodeJS和JS 区别？

1. node可以书写后台，可以搭建服务器

2. nodeJS没有 window DOM BOM对象

3. 对JS进行了增强

### 主模块

你调用的模块 就是主模块

在主模块中引入的模块 统称为 依赖模块

## 模块组成

每一个模块 都包含了5个自由变量

require 引入模块

### 3个类型：

#### A 文件类型

引入方式 require('./\*') 通过require引入的文件 以'./' 或者 '../'打头的文件

#### B 系统模块

引入方式 require('fs') node自带的模块 引入方式 require("模块名");

举例： 重命名模块

fs.rename('旧的文件名', '新的文件名', function(err) {

if (err) throw err;

console.log('重命名完成');

});

#### C 第三方模块

var jq = require('jquery');

npm i 模块名 本地安装

安装在项目中会自动生成一个node\_modules的文件

npm i 模块名 -g 全局安装 安装在c盘中

module 模块中的所有属性

exports 对模块进行暴露

如：module.exports.xx=xx; == exports.xx=xx;

\_\_filename 当前模块的绝对路径

\_\_dirname 当前模块所在的文件夹路径

## NPM 命令

npm init 创建一个package.json描述文件

npm i 包名 安装在node\_modules里面的插件

npm i 包名 -g 将包安装到全局

npm i 包名 --save 安装的同时，会把包名写入 package.json 中

npm remove 包名 移除关于插件的所有依赖一并删除

npm update 包名 更新

npm view <name> versions 查看可用版本

## Cnpm

如何拥有cnpm ----执行以下代码

npm install -g cnpm --registry=https://registry.npm.taobao.org

# Gulp

gulp官网 <http://www.gulpjs.com.cn/>

## 使用步骤

1） 全局安装gulp

cnpm i gulp@3.9.1 -g

2） 生成package.json文件

npm init

2） 局部安装gulp

cnpm i gulp@3.9.1 --save-dev

在项目根目录中 创建一个 <gulpfile.js> 的文件

在文件中引入模块

var gulp = require('gulp');

## 插件网站

npm 所有插件 ：https://npmjs.org/browse/keyword/gulpplugin

gulp插件地址 ：<http://gulpjs.com/plugins/>

1） 安装插件（模块）

2） 引入模块

3） 官网示例

在默认任务里 配置一个数组 可以执行多个任务

压缩CSS gulp-clean-css

压缩图片 gulp-imagemin

编译less gulp-less

重命名 gulp-rename

压缩JS gulp-uglify

#### 执行多重任务

举个栗子 // 编译less任务+压缩+重命名

gulp.task('less', function() {

return pipeline(

gulp.src('less/\*.less'), // 文件路径

less({

paths:[path.join(\_\_dirname,'less', 'includes')]

}), // 编译less 不用改

cleanCSS({ compatibility: 'ie8' }),

rename(function(path) {

path.basename += '.min';

}), // 输出过后 更名)

gulp.dest('dist/css') // 输出路径

)

});

## 观察者模式

// 调起观察者模式

gulp.task('default', function() {

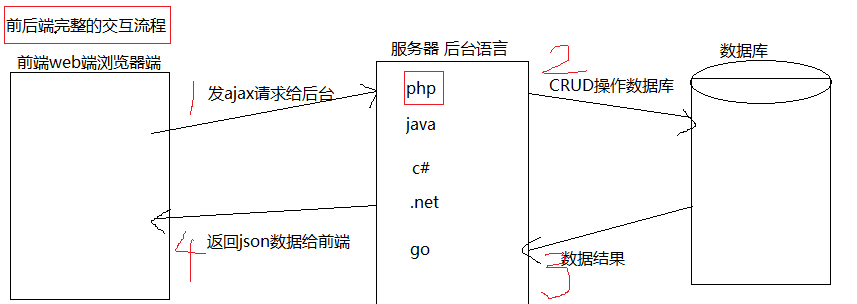
gulp.watch("less/\*.less", ["less"]);

gulp.watch("js/\*.js", ["uglify"]);

});

# Php

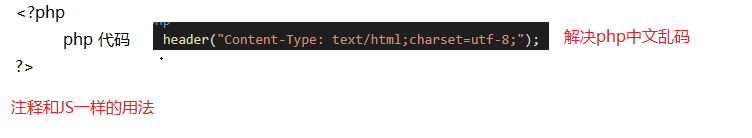
## 前后端交互流程



## Php运行环境



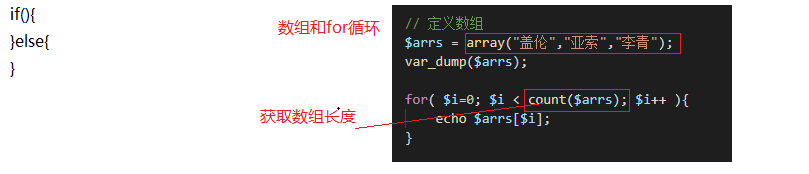
## Php语法



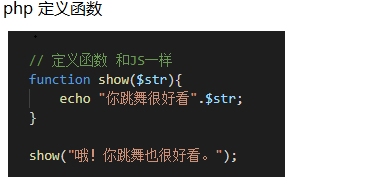
## Php变量



## if条件判定语句和for循环语句



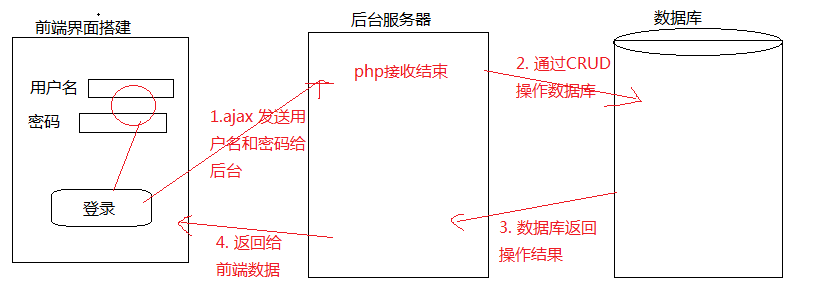
## 定义函数



## 连接数据库



## Php登录功能





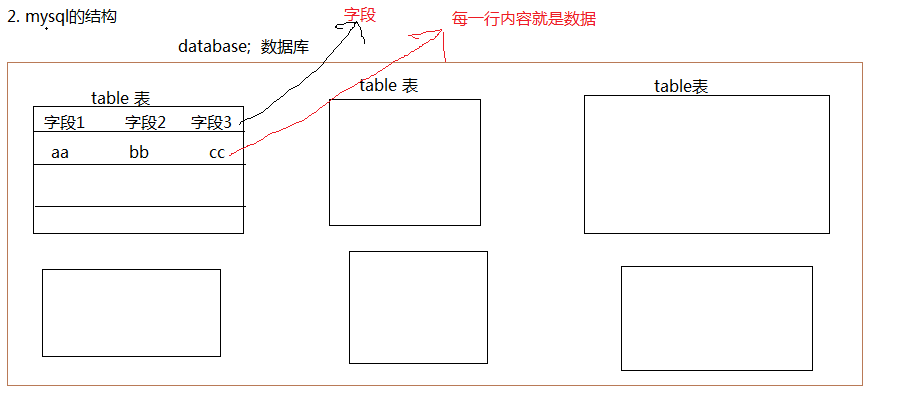


# Mysql

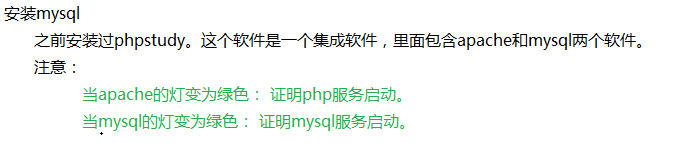
## 概念



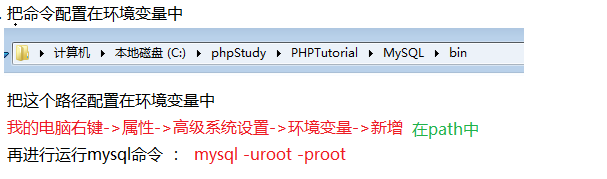
## Mysql接口



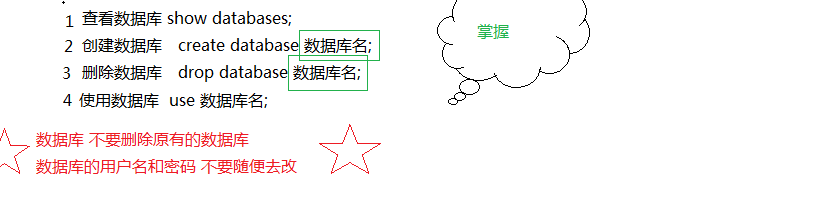
## 安装mysql



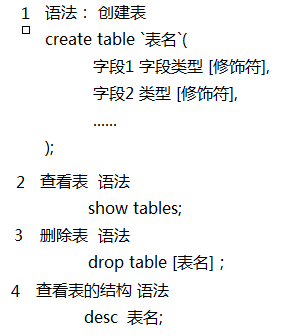
## 配置环境



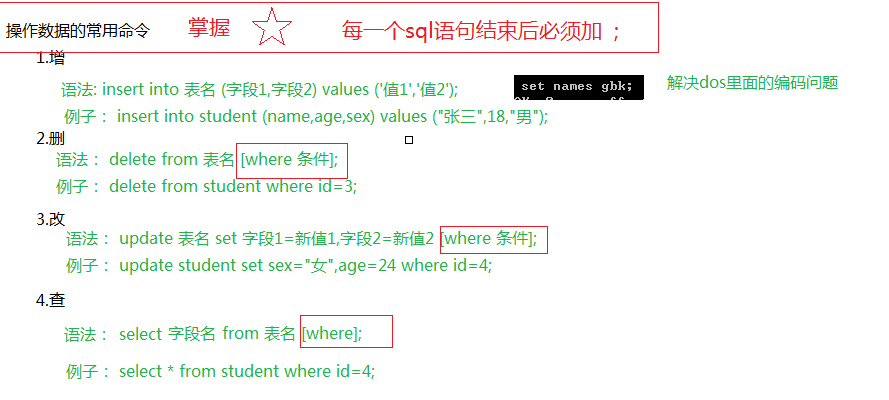
## 常用数据库命令



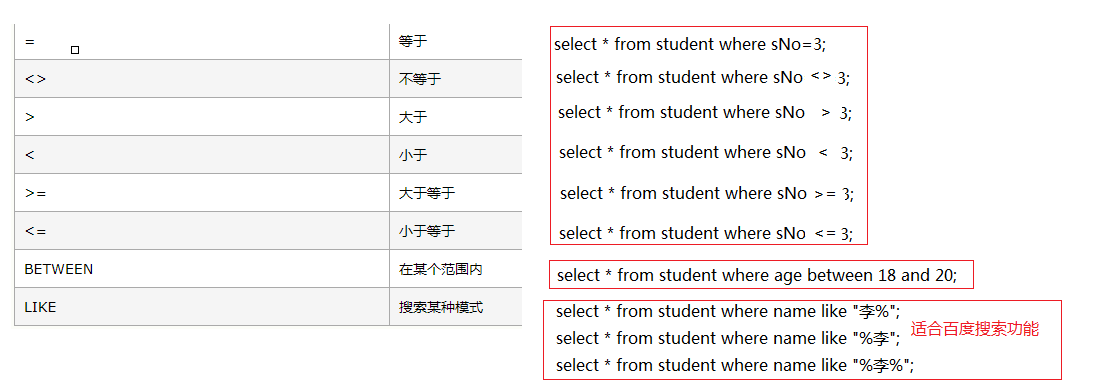
## 常用建表语句

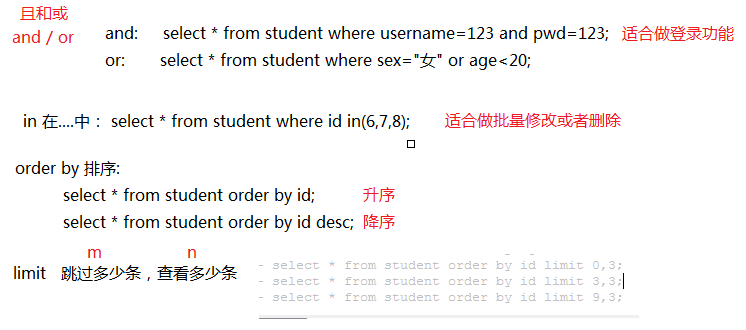


## 操作数据常用命令

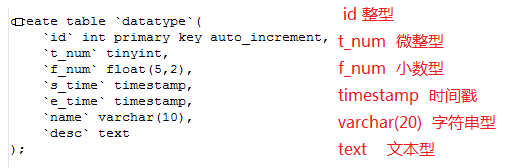


## 条件语句





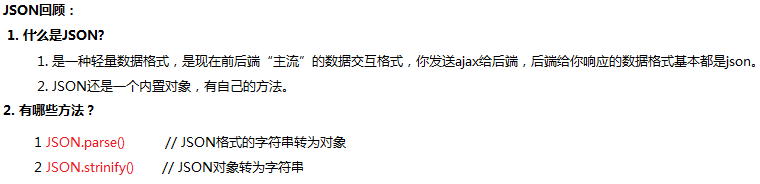
## 数据类型



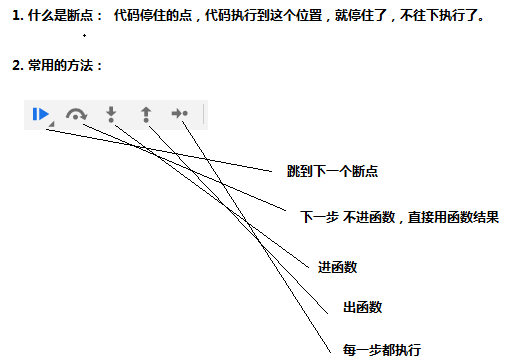
# Js高级（重点）

## 高级js基础

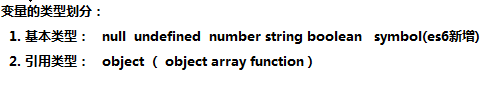
### Json回顾



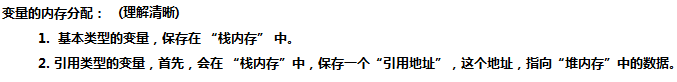
### Debugger

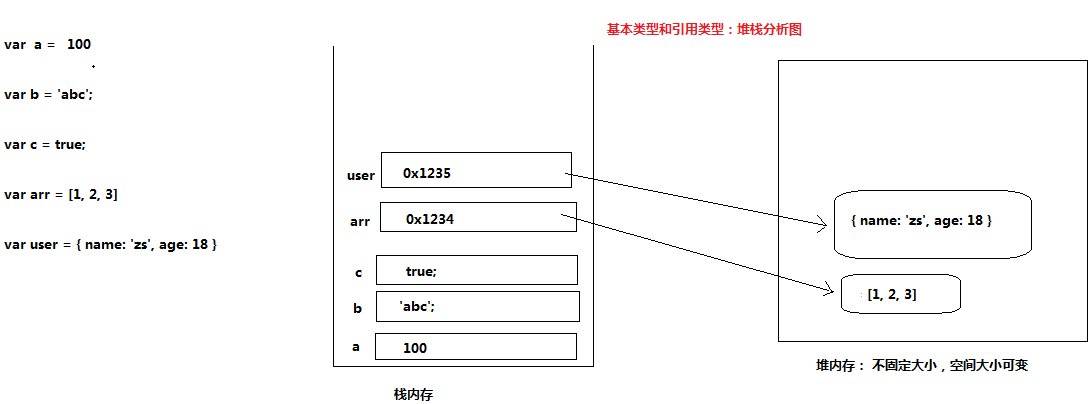


### 变量划分

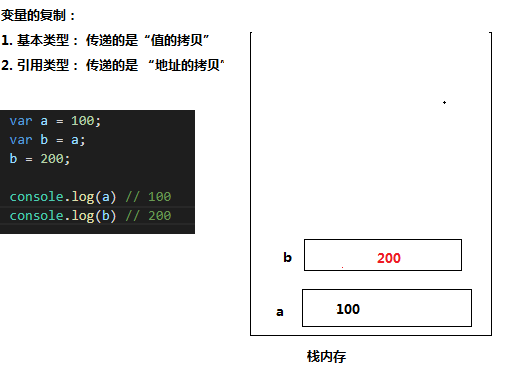


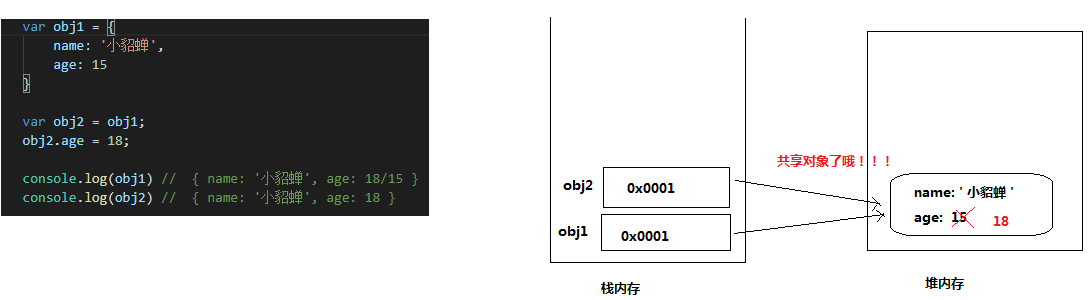
### 变量内存分配



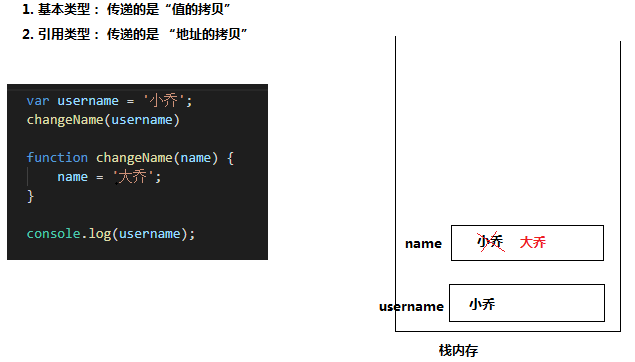


### 变量复制

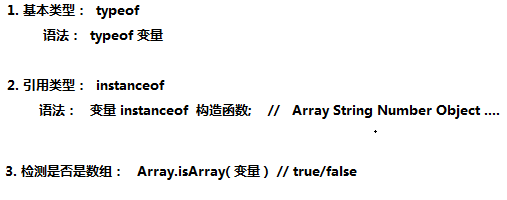




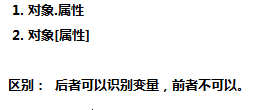
### 参数传递



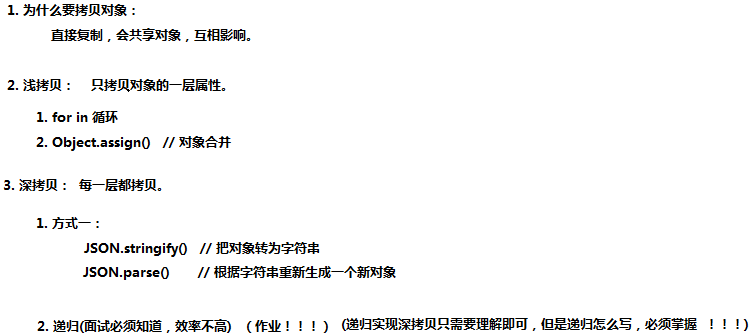
### 变量类型检测



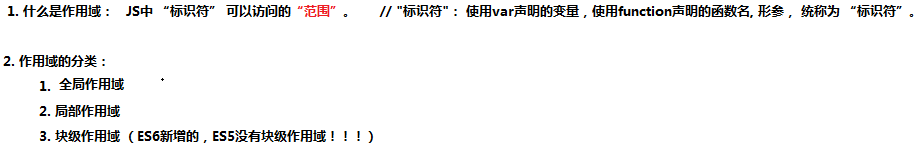
### 对象获取或设置属性的两种方式



## 对象的浅拷贝与深拷贝

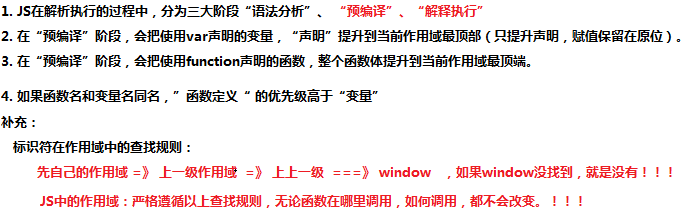


### 作用域进阶

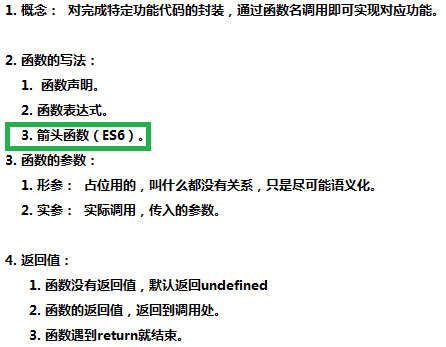


## Js编译原理与B包

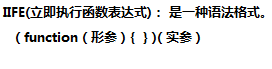
### 编译原理



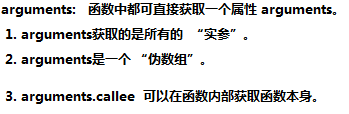
### 函数



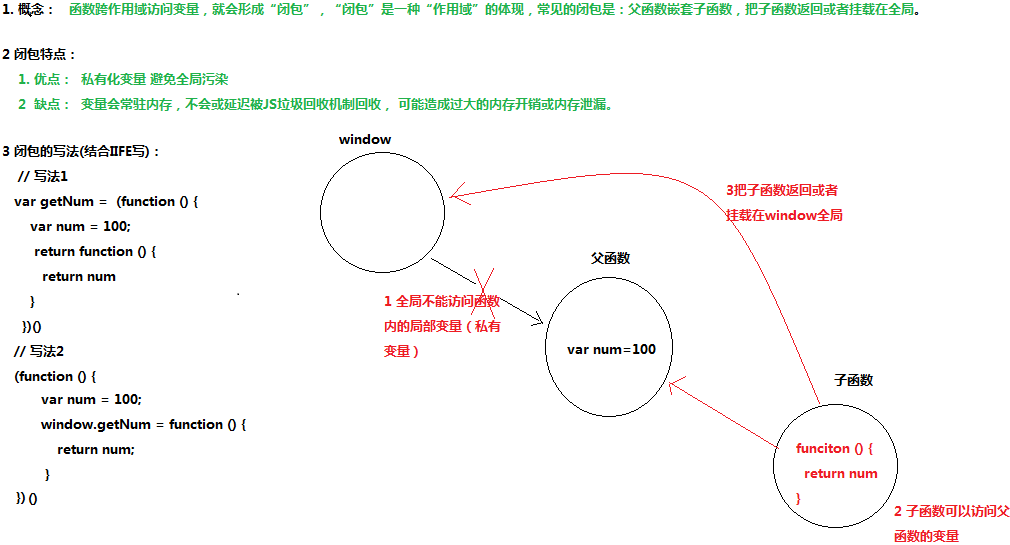
### Iife立即执行函数



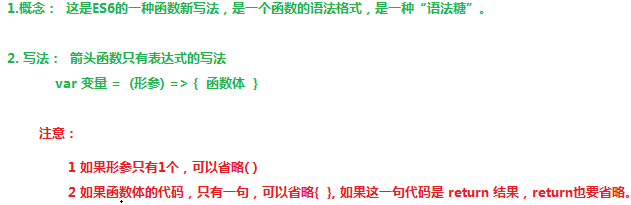
### Arguments



### B包

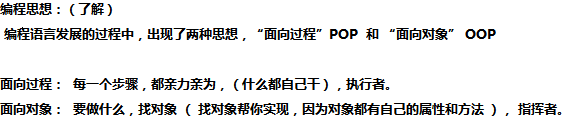


## 箭头函数

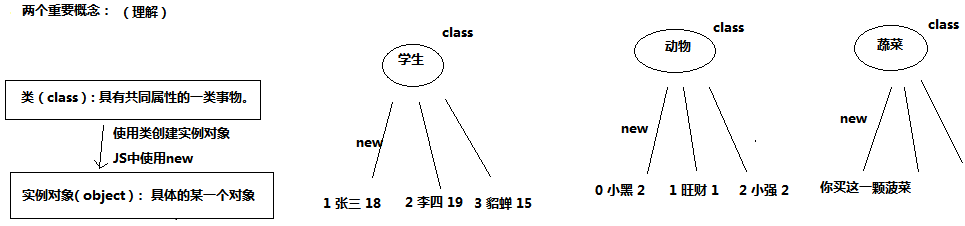


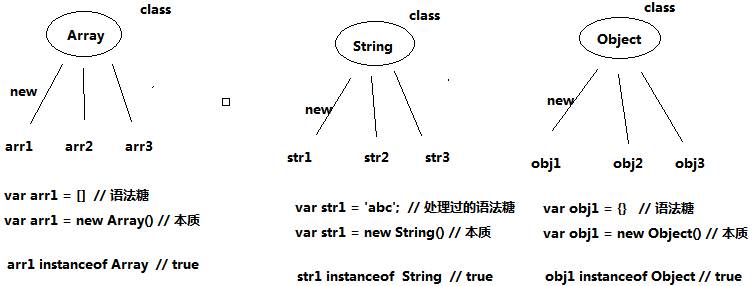
## 构造函数原型this

### 编程思想

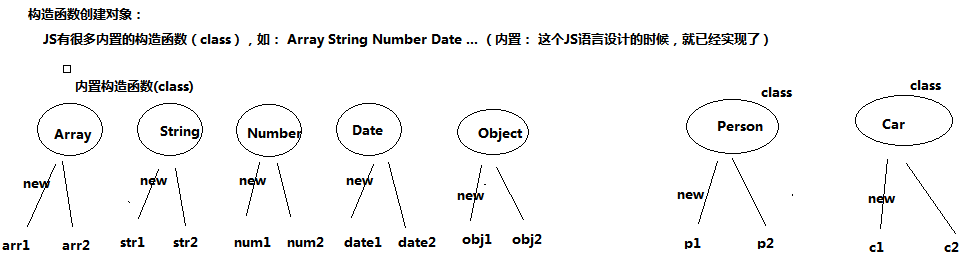


### 类与实例对象



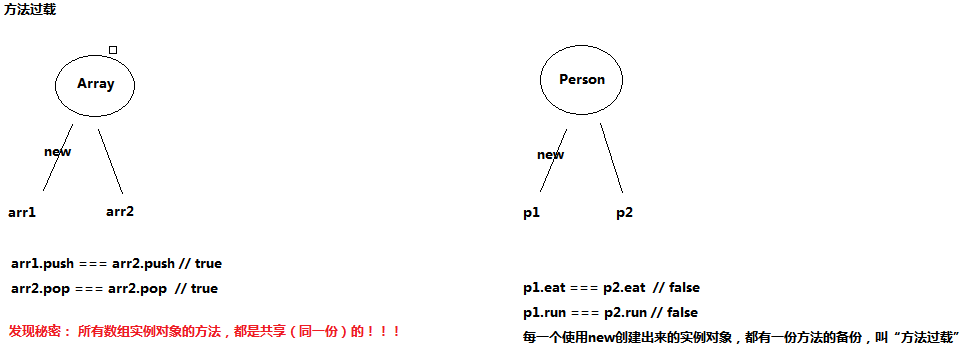


### 构造函数创建对象



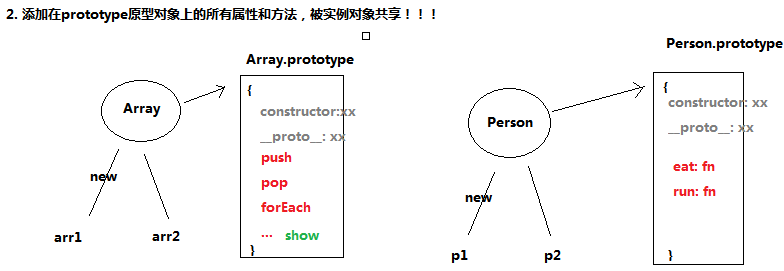


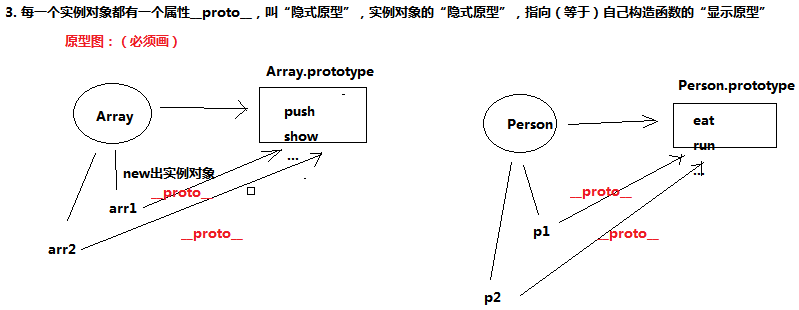
### 方法过载

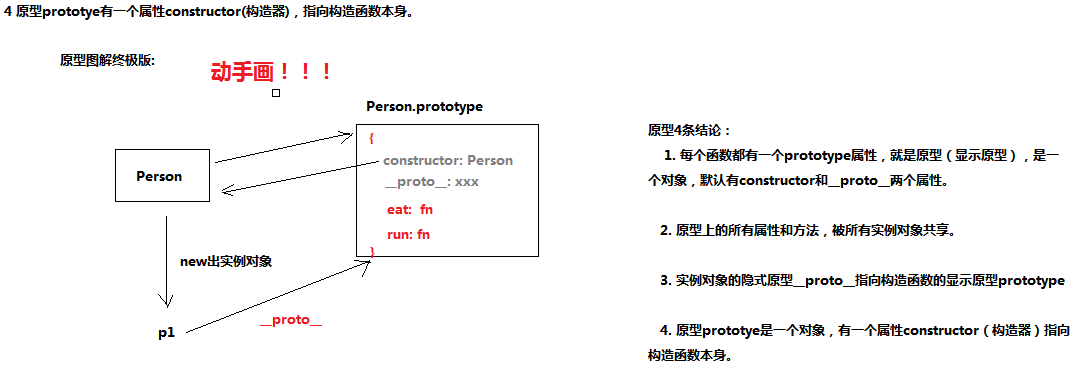


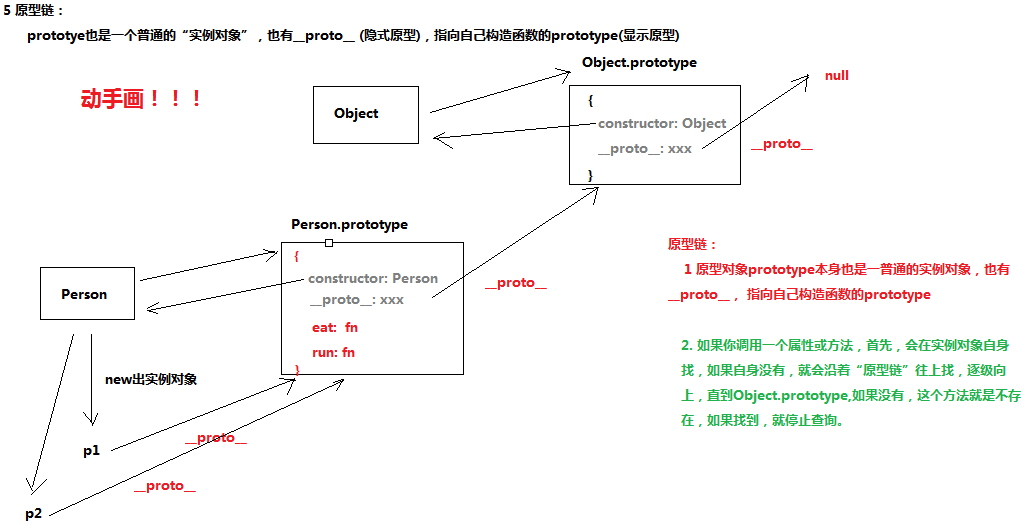
### 原型

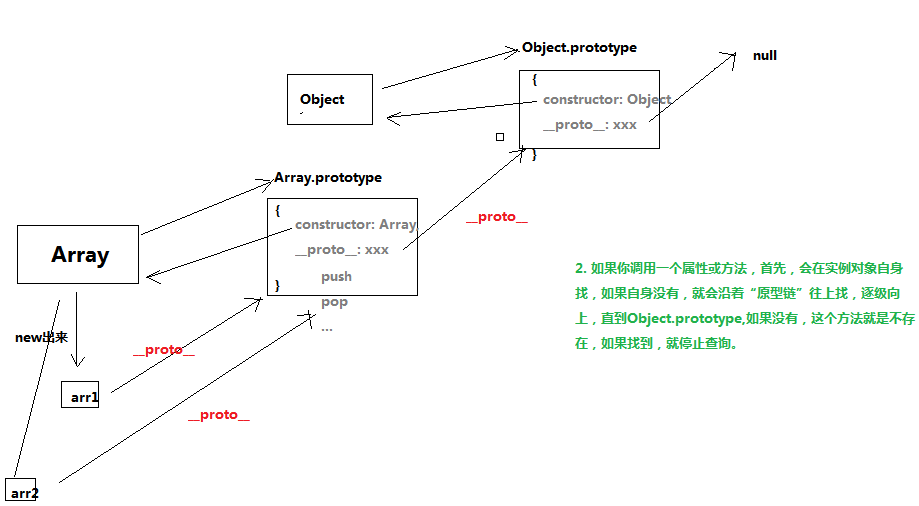




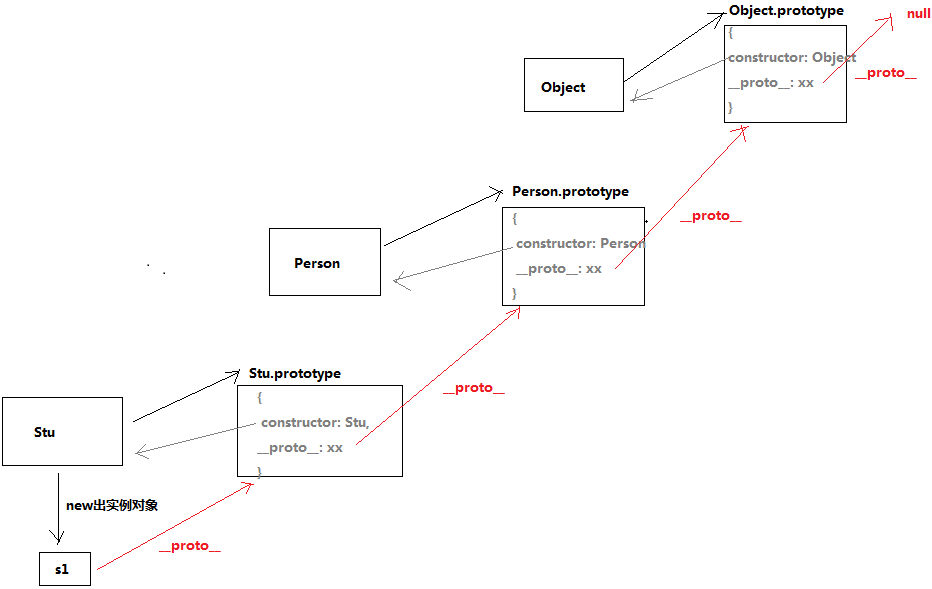




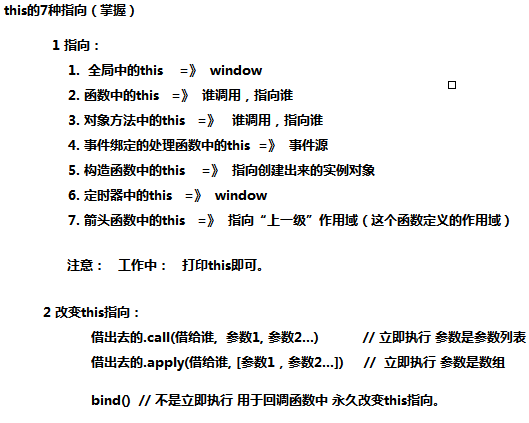




终极版原型链：



### This



## 继承

### 概念

**继承就是一种对原始类型（父类）进行细化、扩展的一个过程。**

继承的核心在于让子类拥有父类的一切特征。那么在父类之后定义子类，分别具备父类的原生和原型部分的所有特征。

这种方式虽然搞定了子类的特性问题，但是并不被JavaScript引擎推崇。

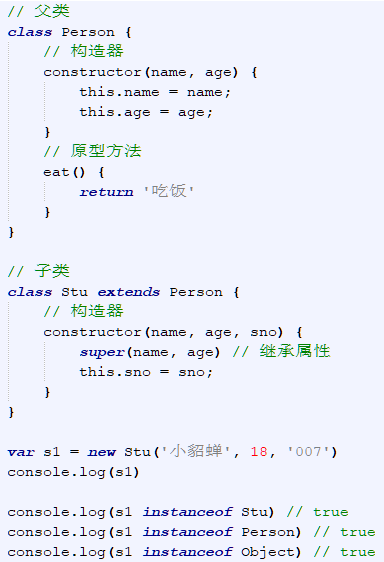
实现步骤：

1. 找到（声明）父类
2. 创建子类，构造器中继承父类所有原生属性
3. 子类原型，拷贝父类原型

### Es5继承

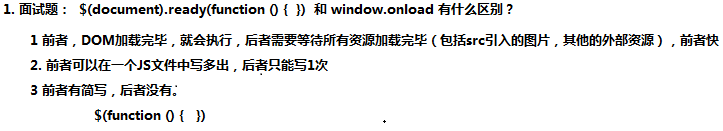


### Es6继承

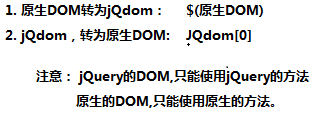


## 插件开发

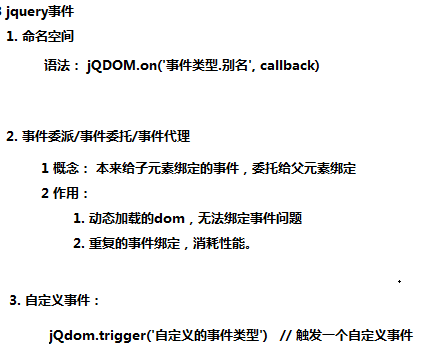
### 万能$



### Jq与原生Dom的转化

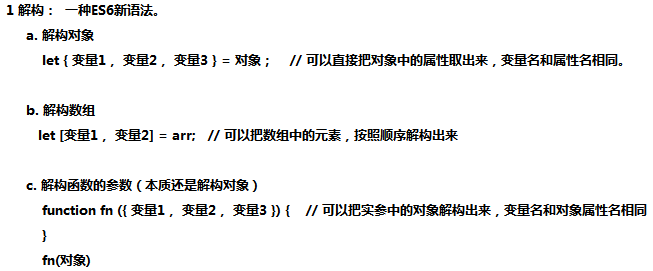


### Jq事件



## Es6

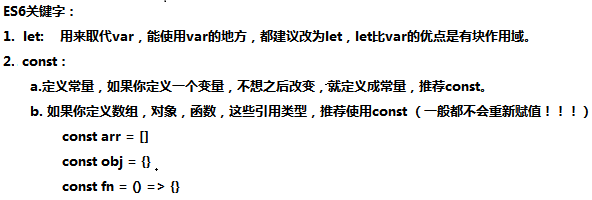
### 解构



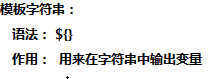
### 扩展运算符



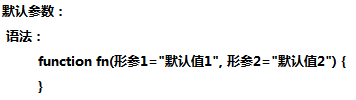
### Es6关键字



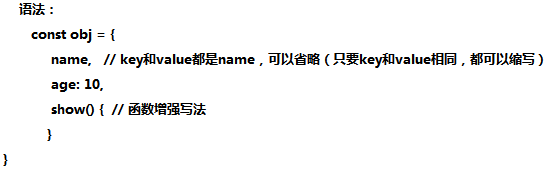
### 模板字符串



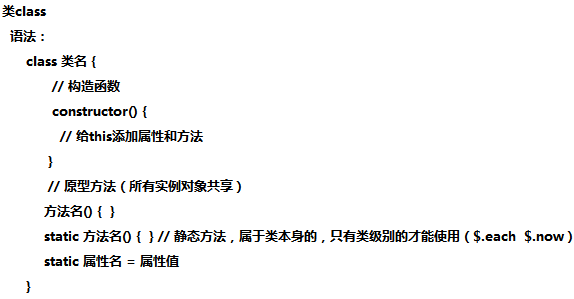
### 默认参数



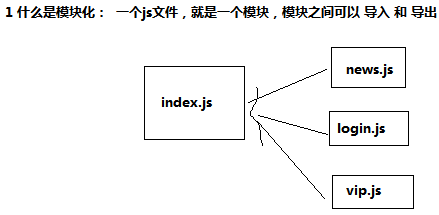
### 对象增语法

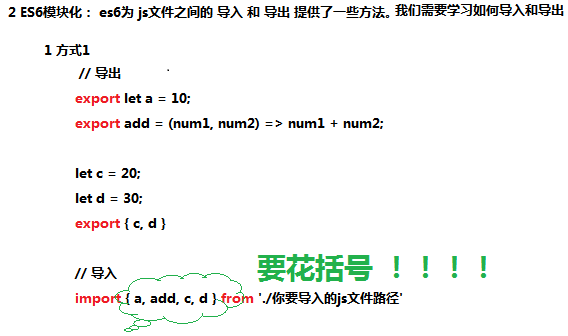


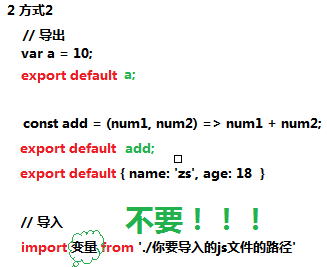
### 类class

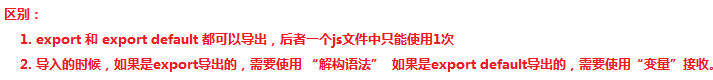


### 模块化：module

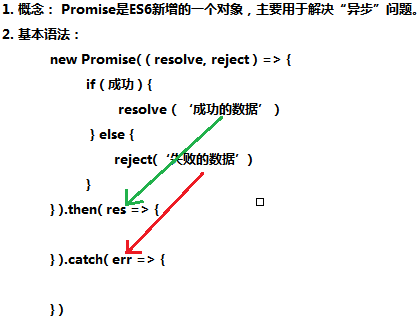






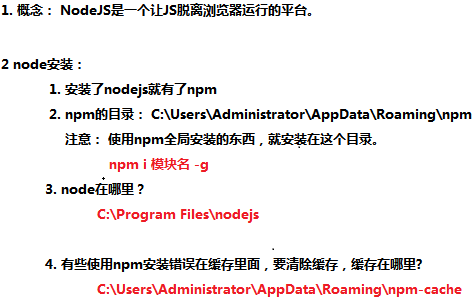


### Promise基本使用

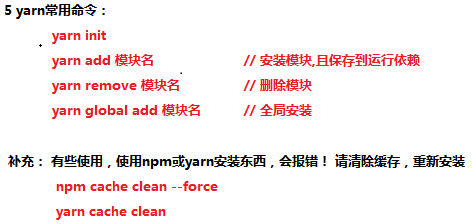


# Nodejs进阶

## Node回顾

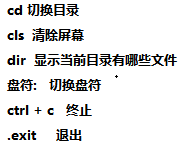




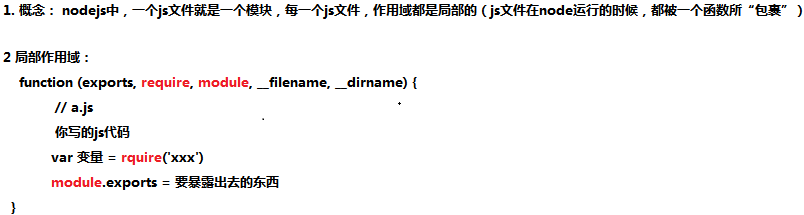


## promise async

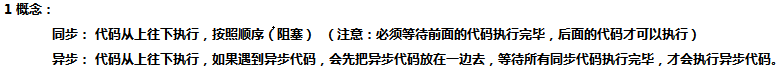
### Dos常用命令



### Nodejs作用域



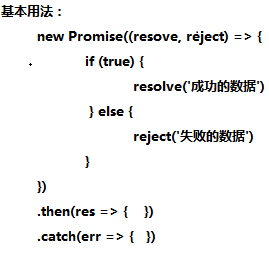
### 同步异步

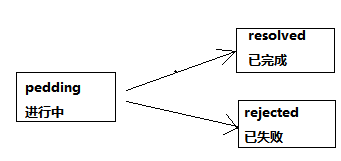


### Promise

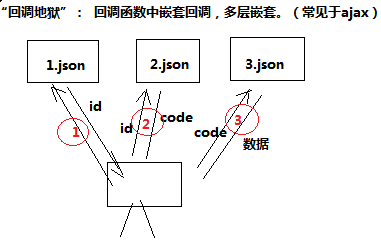


#### 基本用法

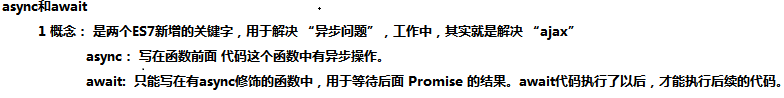


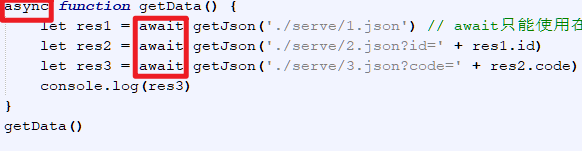


#### 回调地狱



### Async和await





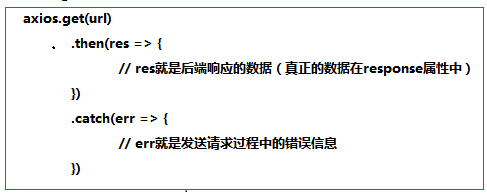
## Axios

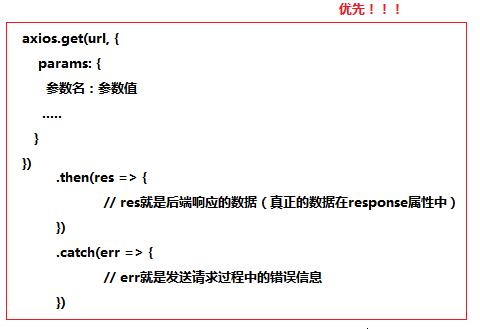
### 概念



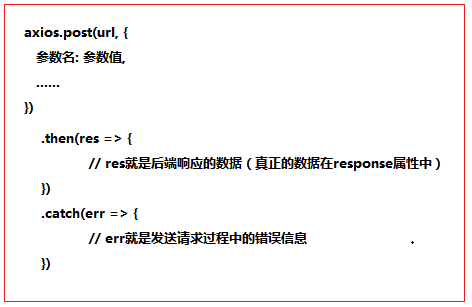
### 使用

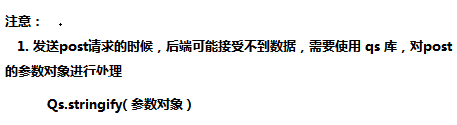
#### 发送get请求





#### 发送post请求





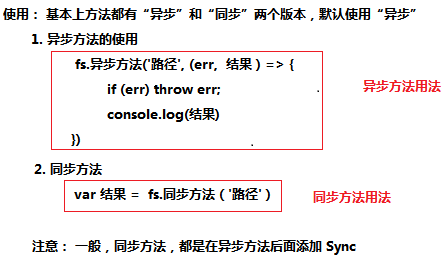
## 文件、路径和http模块

### Fs模块

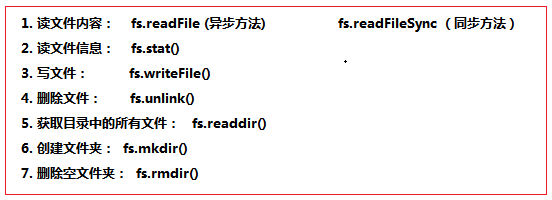
#### 概念



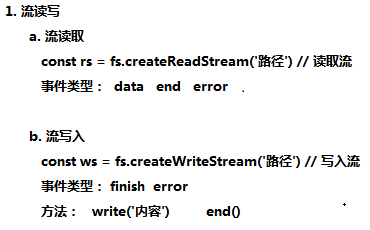
#### 使用

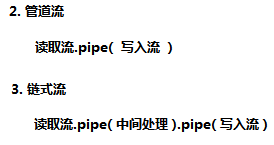


#### 常用方法



#### Fs模块的流读写

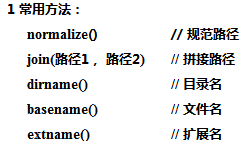




### Path模块

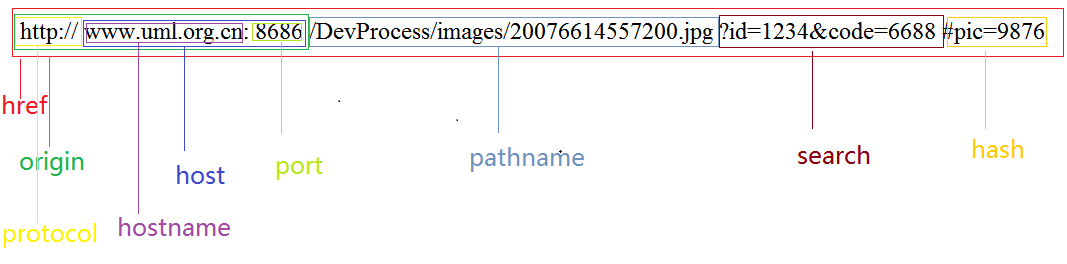


#### 常用方法



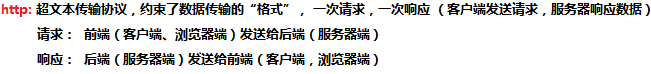
### Url模块



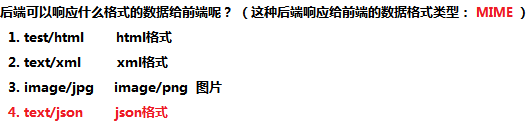


### http模块

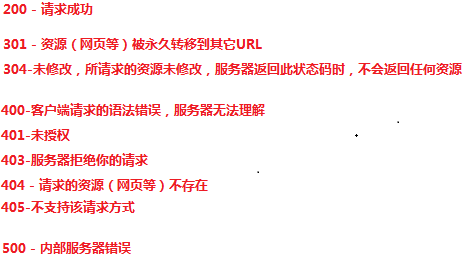
#### 前端请求



#### 后端相应数据



#### http状态码



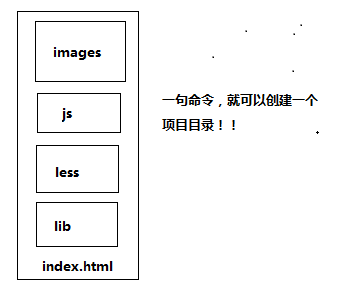
## Express框架

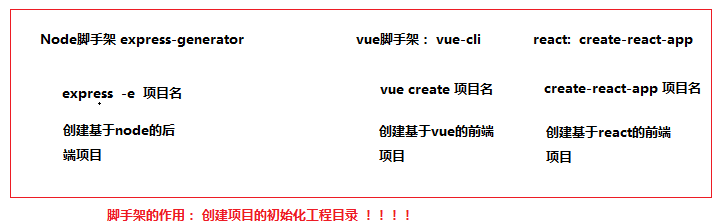
### 概念



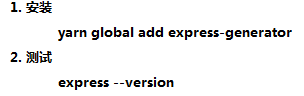
### 脚手架



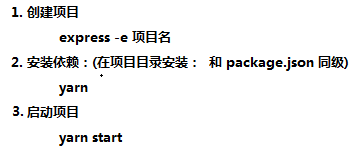




### 脚手架安装

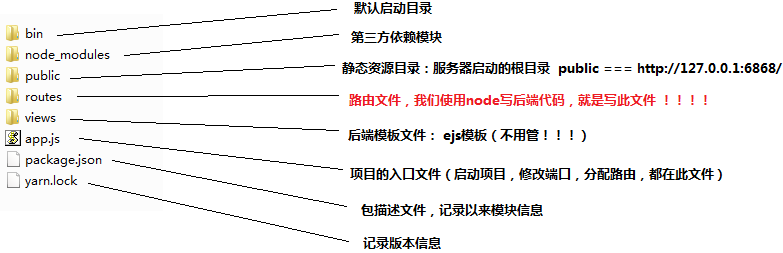


### 使用脚手架创建项目步骤

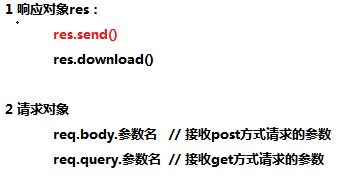




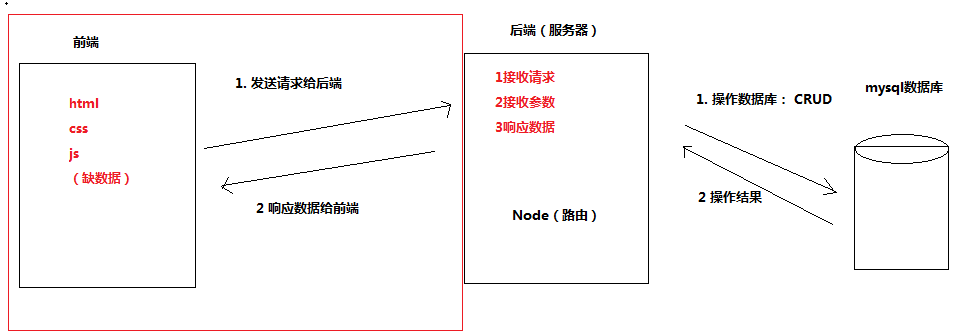
### 项目目录



### 相应对象res和请求对象req



### 前后端交互流程



## Websocket通讯协议

### 介绍

举例来说，我们想了解今天的天气，只能是客户端向服务器发出请求，服务器返回查询结果。HTTP 协议做不到服务器主动向客户端推送信息。

WebSocket是HTML5开始提供的一种在单个 TCP 连接上进行全双向通讯的协议。

### 使用

#### 安装

需要下载一个模块 socket.io (前端和后端都需要，只要引入了这个模块，前端可以发送数据给后端，后端

也可以发送数据给前端)

a. 前端安装： 引入 socket.io.js 即可

b. 后端安装：

1) 安装： yarn add socket.io

2) 引入： const io = require('socket.io')

说明： socket.io是一个js模块，集成了webscoket的功能。

#### 常用方法

A 发送数据和接收数据：无论是前端发送给后端 还是后端发送给前端 都是一样的

1. 发送数据

socket对象.emit('自定义事件类型', '数据')

2. 接收数据：

socket对象.on('自定义事件类型', (data) => {

data就是接收到的数据

})

B 广播数据：

io.sockets.emit('自定义事件类型', '数据');

# mvc与mvvm架构模式

## 打包上线流程

!!!!!!!!记住 不用管后台怎么搭建的服务器，你只需要问后台发布路径是什么即可，把压缩完的内容CV到此目录即可!!!!!!!!

### 流程：

1. 在公司电脑搭建环境

2. 根据产品文档做前端（页面/JS/CSS）

3. 测试测试你写的需求，提bug

4. 修改需求

5. ※※性能优化（懒加载，图片压缩，公共组件抽取，各种HTML文件JS文件CSS文件压缩）

6. 打包发布给服务器上线

打包给后台程序员看或者打包发布上线：

1. 进入vue项目

2. 使用yarn build进行项目打包

3. 拿到打包以后的文件夹dist

### 搭建express服务器：

1. 全局安装express脚手架（名字叫express-generator）

yarn global add express-generator

npm i express-generator -g

2. 使用express创建一个服务器（一个真正的服务器，可以提供数据接口，以及存放网页让其他用户访问）

express 项目名

后台项目创建完毕后

yarn || npm i 安装项目依赖

启动服务器

yarn start || npm

3. 把前端压缩完成的内容，放入服务器public目录中

记住，此目录固定为public!!不要放错了！！！

4. 使用IP+端口访问即可访问新页面

## Mvc

### 书写步骤

1. 分析需求，先写View（视图） 静态页面

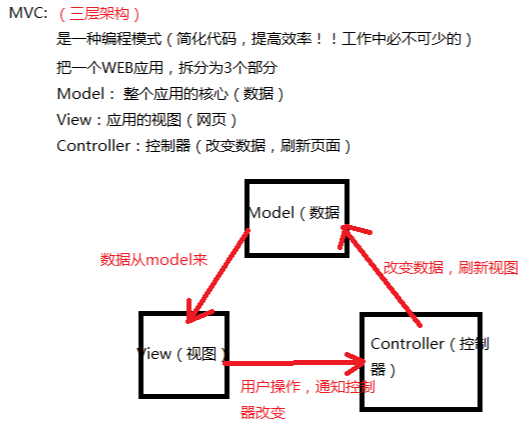
2. 根据静态页面，整理动态数据，书写Model（数据），把动态数据存放到Model中

3. 连接Model和View

4. 寻找用户操作要改变数据的地方，书写对应的Controller（控制器）

V -> M -> C

### 三层架构图



## Mvvm

### 概念

MVVM是一种架构模式（VUE基于MVVM写出来的）

它就是MVC的升级版，和MVC的唯一区别就是增加了双向绑定，数据更新自动更新视图

Model: 它指代应用的数据（核心） --> vue的data返回的对象

View: 应用的视图（用户界面） --> vue的template

ViewModel：连接View和Model，实现数据双向绑定的核心 MVVM最重要的一层

### Vue模式

vue是基于mvvm产出的数据驱动双向绑定框架

1. 组件化开发

组件可复用（减少代码冗余） 可组合（让页面更加灵活） 易维护（把一个组件的html css js放到了一起）

每个组件分为3个部分

template 模板

script JS（行为）

style 样式

2. 占位符

在template中，vue采用{{mustauche}}来占位

3. 指令

v-for: 循环

v-bind: 绑定动态属性

v-on: 绑定动态事件

v-if: 增加/删除节点（操作节点，所以性能低）

v-show: 显示/隐藏（操作CSS样式，所以性能高）

v-model: 用在表单组件上，可以双向绑定数据

### 空电脑搭建项目

安装开发软件

Chrome、FF

NodeJS

VS Code （Atom、Hbuilder）

SVN || GIT （二选一）

美术类： PS，PXCOCK等... 这个没有要求

辅助类： Postman，有道词典（不要乱取变量和文件名！！！！！）

构建VUE项目环境，迅速搭建项目

1. 安装yarn

npm i yarn -g

配置环境变量

yarn global bin 查看全局安装目录，把此路径，复制到系统环境变量的path属性中

2. 安装vue脚手架(脚手架分为2个大版本，当年换过脚手架的名字1.x 2.x vue-cli 3.x @vue/cli )

yarn global add @vue/cli

3. 创建项目

两种创建 图文化创建 vue ui（不要在公司用，low）

命令创建

vue create 项目名（项目名不能有大写）

4. 启动项目

启动： yarn serve

打包： yarn build （把dist内打包好的文件，拖入发布目录即可）

### Vue指令

vue是基于mvvm产出的数据驱动双向绑定框架

1. 组件化开发

组件可复用（减少代码冗余） 可组合（让页面更加灵活） 易维护（把一个组件的html css js放到了一起）

每个组件分为3个部分

template 模板

script JS（行为）

style 样式

2. 占位符

在template中，vue采用{{mustauche}}来占位

3. 指令

v-for: 循环

v-bind: 绑定动态属性

v-on: 绑定动态事件

v-if: 增加/删除节点（操作节点，所以性能低）

v-show: 显示/隐藏（操作CSS样式，所以性能高）

v-model: 用在表单组件上，可以双向绑定数据

### Vue生命周期

new Vue()

1. 初始化自己的内部事件

beforeCreate: 通知函数，告诉你要开始工作了

2. 初始化组件data数据，进行属性代理，转为getter/setter响应式属性

created: 通知函数，data数据转换完毕，可以开始发送初始化ajax

3. 询问，是否指定了挂载的容器？ 是：继续往下 否：等待容器挂载以后才继续

4. 是否有template模板

5. 将template的虚拟节点翻译为真实节点，保存到render函数中

beforeMount: 通知函数，template转译完毕，可以开始渲染

6. render函数把转译的节点，渲染到el容器中

mounted: 通知函数，DOM节点挂载完毕，用户可以看到真实节点 （获取DOM节点）

网页已经准备完毕并显示，用户可以操作网页

beforeUpdate： 通知，data数据发生改变，可以开始更新

7. 再次转译template到render函数中，渲染到el容器中

updated: 通知，节点更新完毕！用户可以看到新的数据

组件销毁

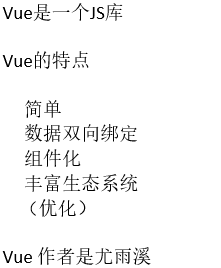
beforeDestroy: 通知，组件即将销毁

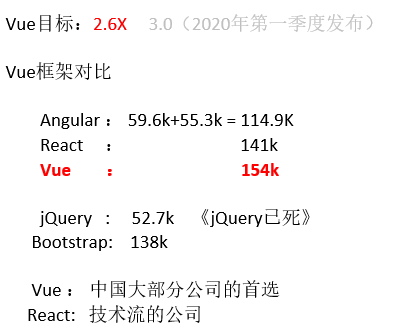
8. vue销毁此组件及此组件所有的数据，事件绑定等，释放内存

destroyed: 通知，组件say goodbye了！在此处卸载定时器等占据内存大的代码和内存泄漏挂钩

# Vue

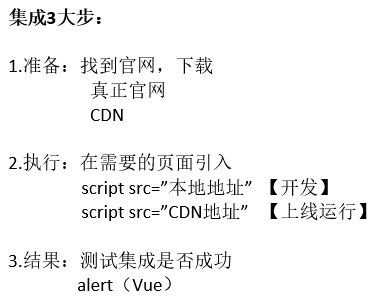
## Vue简介

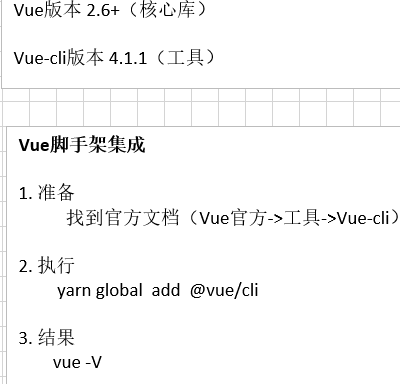




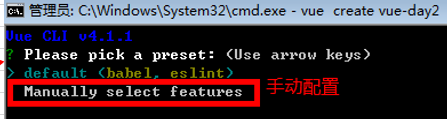
## Vue上手

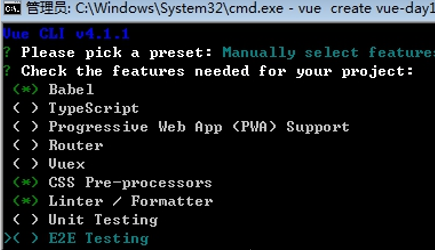
### 安装

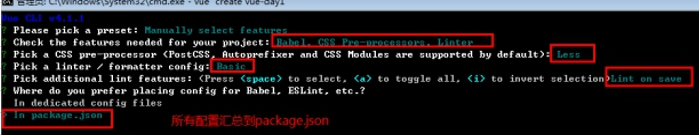


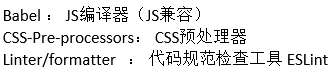




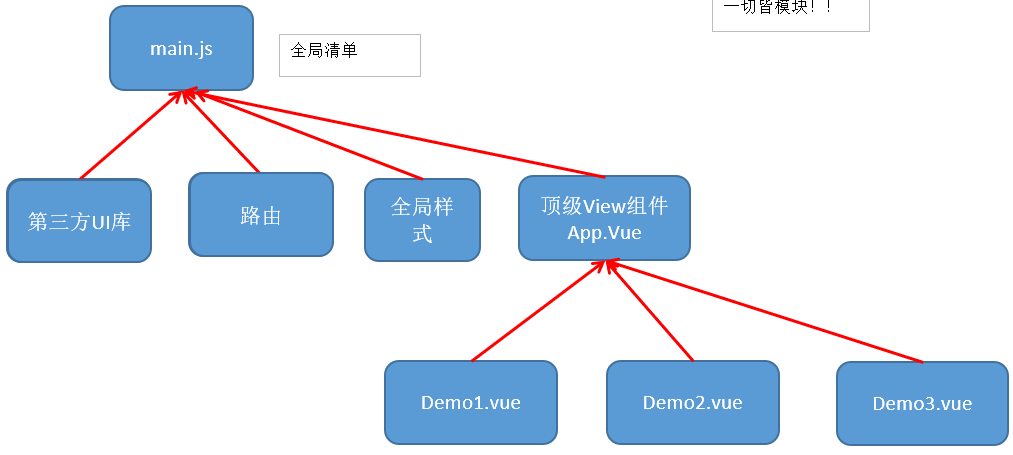




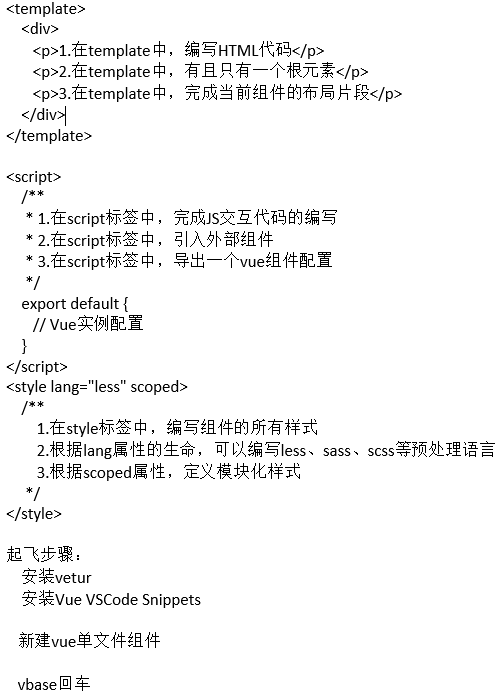




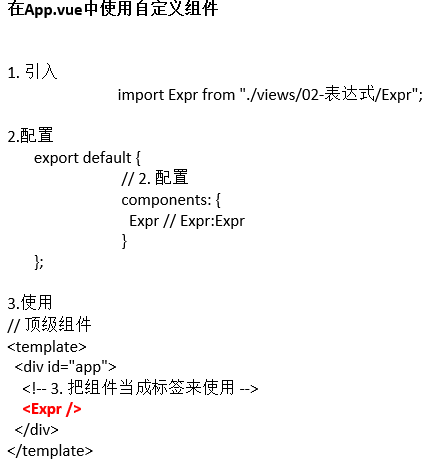
### Vue资源树



### 结构描述



## 自定义组件





## Vue指令

v-text：把data数据，输出到指定标签中。不解析HTML。

（场景： 98%场景直接使用{{表达式}}替代）

v-html：把data数据，输出到指定标签中。要解析HTML。

（场景：富文本编辑器数据回显）

v-show：值为boolean类型，为真是来显示内容，为假隐藏。

核心：使用css的display属性来实现（不安全，针对切换频繁场景）

v-if： 值为boolean类型，为真是来显示内容，为假隐藏。

核心：使用DOM添加和删除（安全，针对切换不频繁场景）

v-for： 用来循环data中的可迭代对象（数组&对象）

数组：v-for=“ value in arr” :key=”value”

对象: v-for=“ (pvalue,pname) in obj” :key=”value”

进阶指令：

v-model：针对表单项，自动绑定表单项的value属性，可以进行内容的显示，也可以修改的结果更新到data数据中（数据双向绑定）【场景：表单内容的收集和回显！】

v-model=”message”

v-model=”user.name”

v-on:事件类型=“事件函数”

Vue的事件机制非常常用，可以直接在标签上定义事件，值绑定一个事件函数

普通事件函数绑定

参数1： 事件对象

内联事件函数绑定

参数1~n-1：自定义参数

参数n ：$event手动传入事件对象（注意引用名称，一个字都不能错！）

v-bind：动态为HTML标签，绑定属性的数据。（轮播，走马灯）

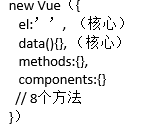
语法 ： <标签 普通属性=“值” **:**动态属性=”**表达式**”/>

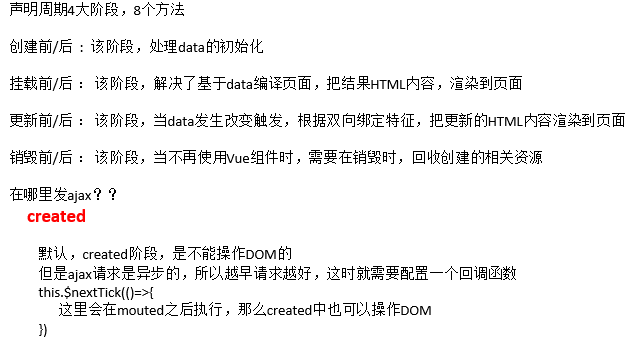
动态样式处理

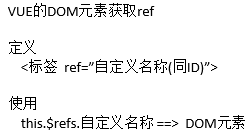
style 样式： <标签 :style=”{属性名：‘属性值’**，**属性名2：‘属性值2’}” 【了解】

class 样式类： <标签 :class=”{样式名1：boolean（是否开启）**，**样式名2：boolean（是否开启）}” 【掌握】

## 生命周期：



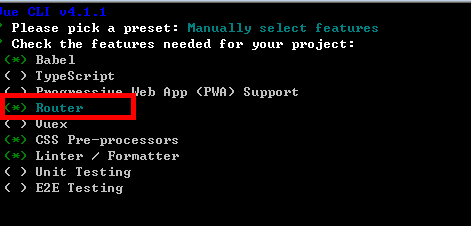


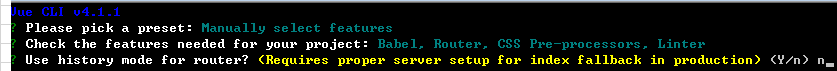


## 路由

**路由集成方式核心**

直接使用脚手架，选中router项，自动配置【掌握】

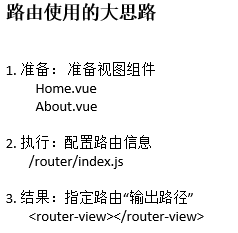


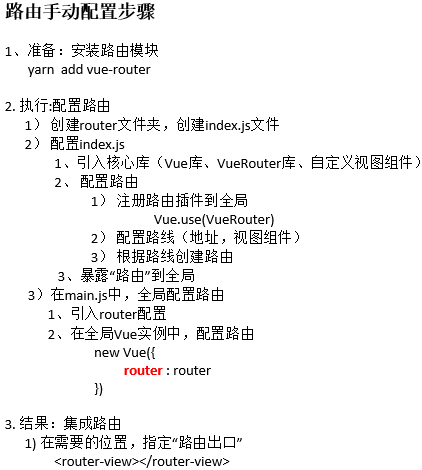


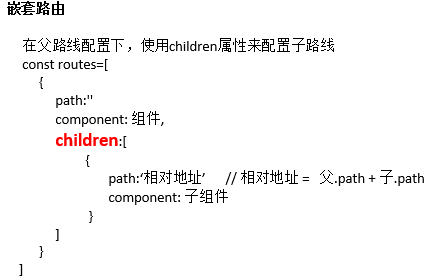
路由目前有2种模式

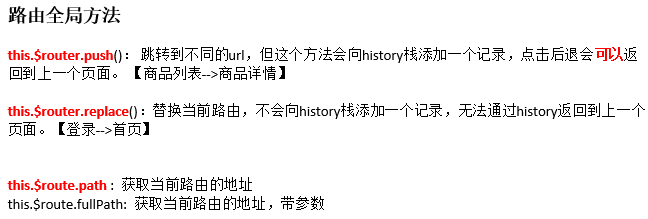
history ： Y，开启history模式需要后台支持

**hash ： N，使用hash路由方式**





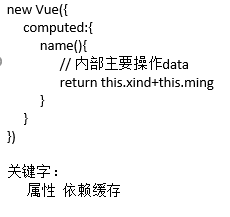




## 其他属性

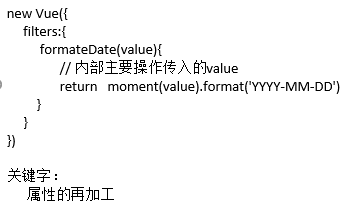
**计算属性（掌握）**

针对data中的数据，进行二次运算，把结果直接当成普通属性输出到页面。在计算依赖发生改变之前，会一直使用前面计算的结果缓存。



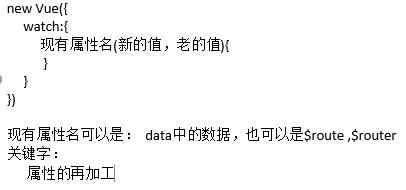
**过滤器（了解）**

针对传入的数据，进行过滤加工，返回一个加工的结果显示到页面。

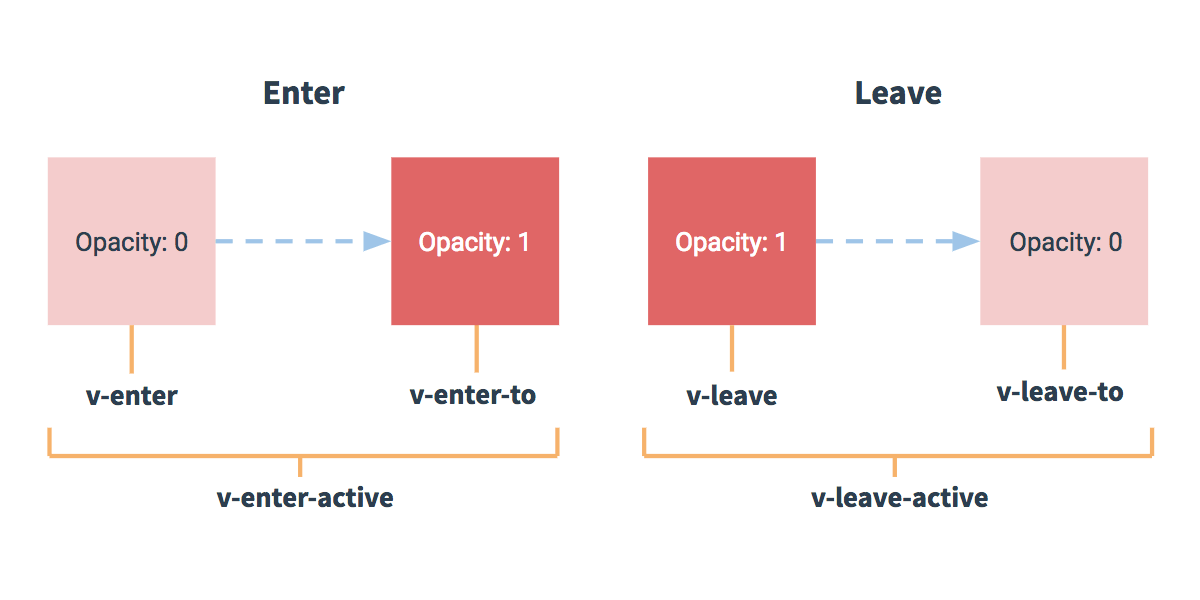


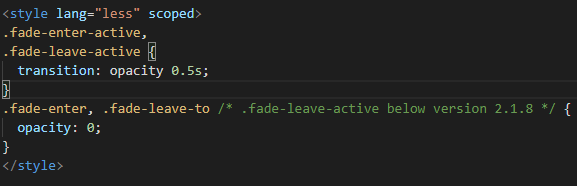
**侦听器（了解）**

针对指定的数据变化进行监听，在监听方法中，可以对数据进行一步捕获和操作



## 动画





在上面6个状态中，我们常规定义2组样式：

1、通过v-enter-active和v-leave-active定义过渡动画的配置（支持元素、时间和贝塞尔曲线）。

2、通过v-enter和v-leave-to定义初始和结束时的动画效果（如位移，旋转等）。

核心点：

**HTML**

<transition name=”动效名称”>

<唯一的一个根元素/.>

</transition>

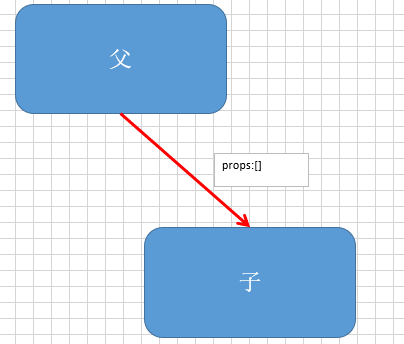
**样式**

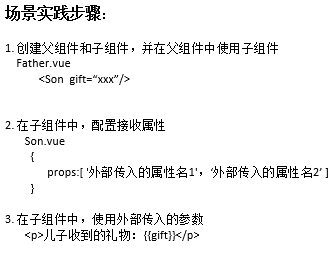
全局的进入，出去的动效配置 (过渡)

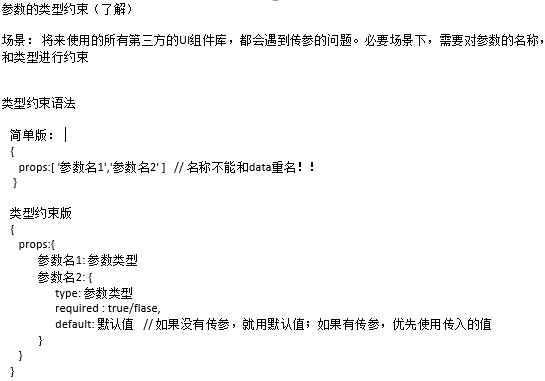
动画属性切换，定义在“动效名称-enter”和“动效名称-leave-to”

## 组件通信

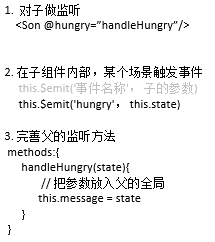
### 父传子



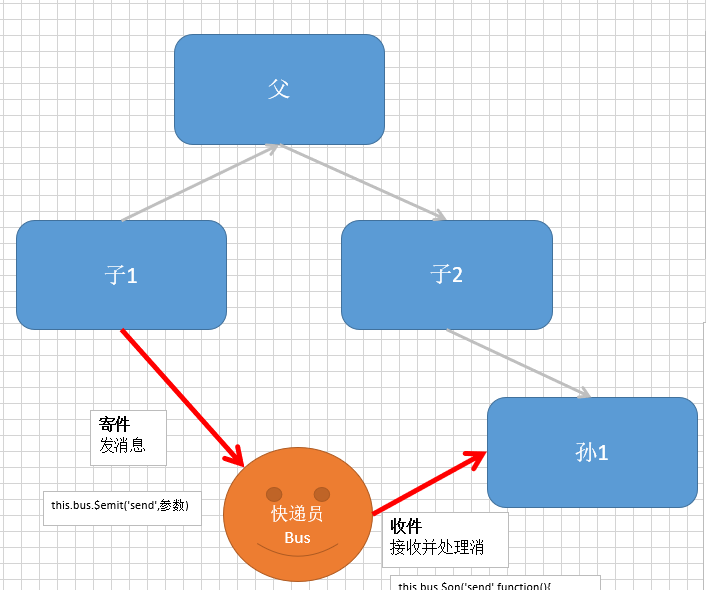


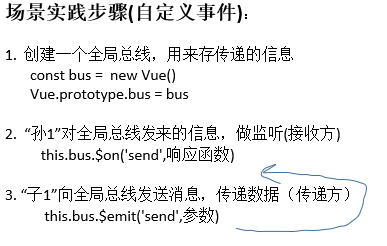


### 子传父

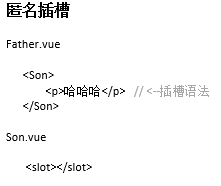


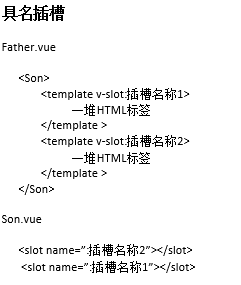
### 兄弟传参

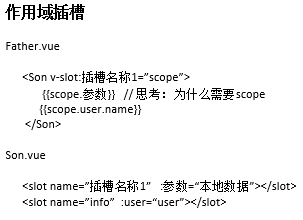




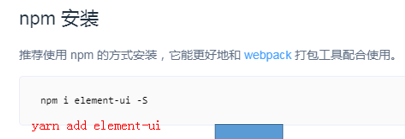
## 插槽



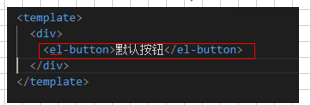




## Element-ui







## betterscroll

使用Betterscroll

1. 安装

yarn add better-scroll -S

2. 引入

在需要使用的页面

import BScroll from 'better-scroll'

3. 使用

1. template部分

<div class='xxxx'>

<ul class='content'>

//里面放要滚动的元素

</ul>

</div>

2. js部分

mounted(){

//因为要获取真实DOM节点!!!!!!!!!!只能放在这个位置，否则元素获取为null

new BScroll('DOM选择器选中真实DOM元素')

}

坑：

1. 绝对不能覆盖ul的class，而且ul的class只能是content

2. div一定一定一定一定要指定高度！！！！！！！！！！！！！！！！！！！！！！！

3. 注意div的class不要和其他界面的class重复了 这里建议 new BSCroll()时，使用id选择器#或者使用唯一的class 例如goods\_xxx

## Vuex【重点】

Vuex是什么？作用？

是一种状态机，它主要用于管理vue应用的数据交互 （主要解决跨多级组件传值，及复杂组件联动交互传值等）

\*不用考虑数据的层级关系\*

Vuex的值只有Vuex可以修改，不能再Vue组件中直接修改！！！

（谁的数据谁维护）

### 下载

yarn add vuex -S

### 引入

在项目中新建store文件夹，下面新建index.js(结构和router很像)

import Vue from 'vue'

import Vuex from 'vuex'

Vue.use(Vuex)

### 使用

#### 3.1 创建仓库store，用来存放整个web应用的交互数据

var store = new Vuex.Store()

#### 3.2 设置要交互的数据

var store = new Vuex.Store({

state: {

name: 'zs'

}

})

export default store

#### 3.3 把暴露的store和new Vue实例绑定起来

new Vue({

router,

store, //vue中所有组件都可以使用this.$store访问到此数据仓库

render: h => h(App)

}).$mount('#app')

VUEX取值（组件取store中的值）

this.$store.state.要取得属性名字

### 取值都放在computed中，提高代码性能，不会重复取值

computed:{

xxx(){

return this.$store.state.xxxx

}

}

VUEX改值（VUEX的值只有VUEX自己能修改，不能直接在VUE中使用=给它强行赋值）

改值只能使用mutation去操作自己的state

mutations: {

xxx(state,传入的参数){

state.xxxx = xxx //在mutation中就可以直接改值

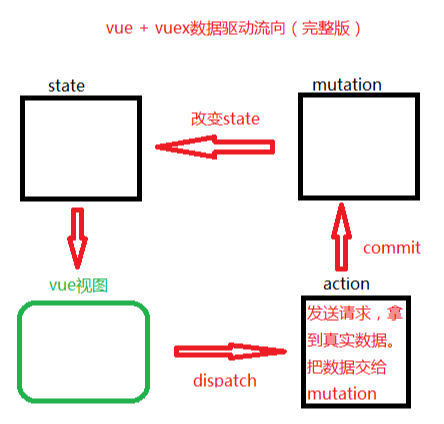
}

}

//记住mutation要手动触发

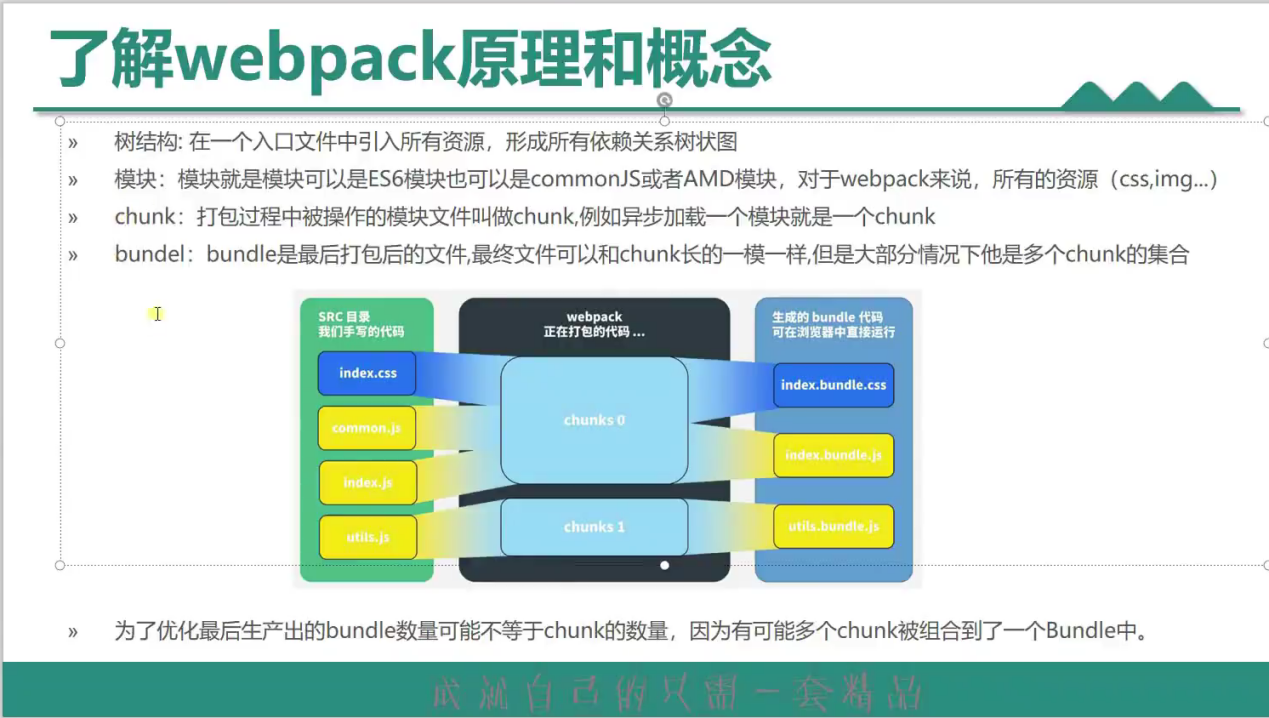
在vue中

使用 this.$store.commit('改变函数的名字',参数) //触发



# **Webpack**

## Webpack简介



## 核心概念



## Eslint



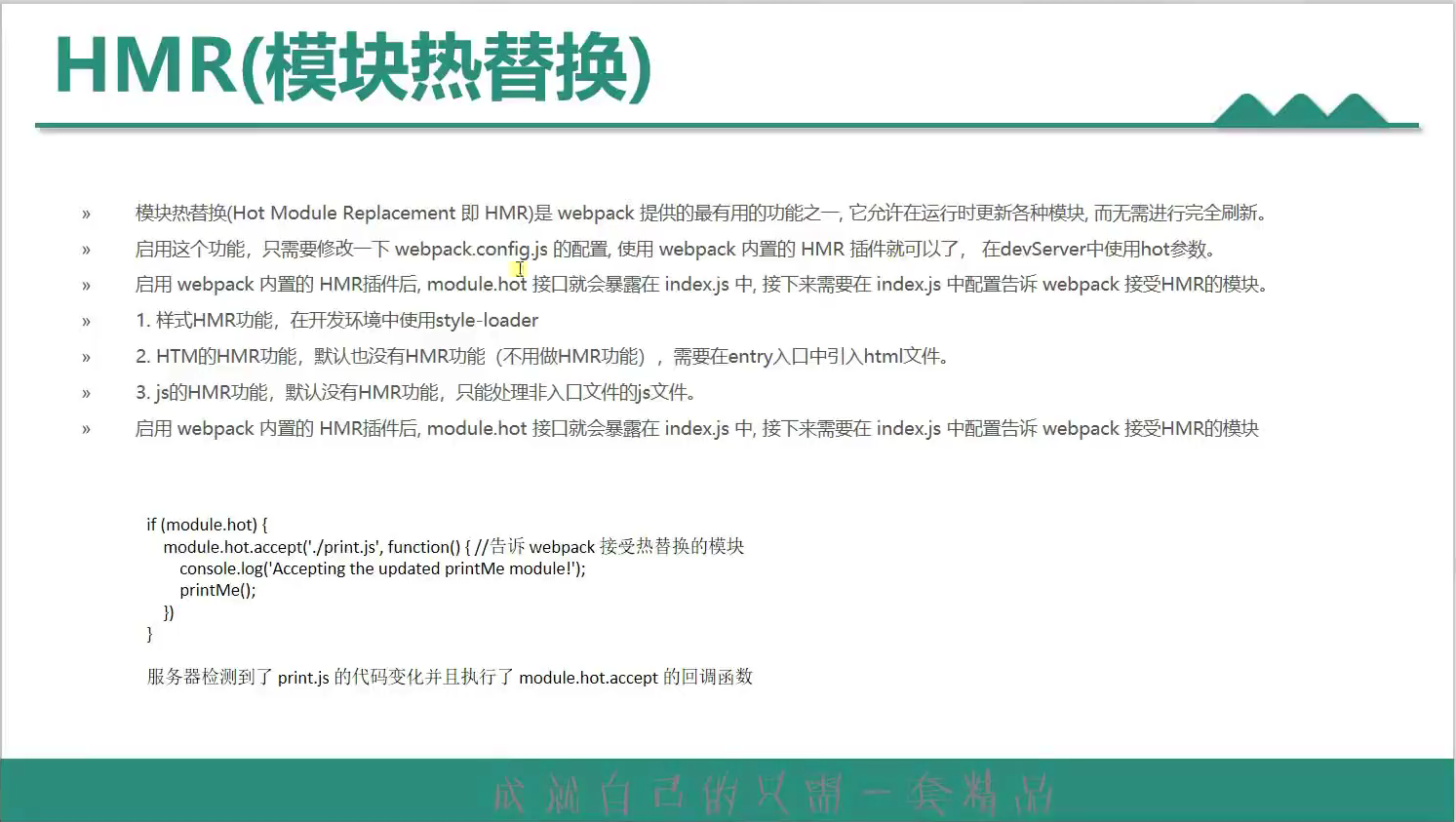
## devserve



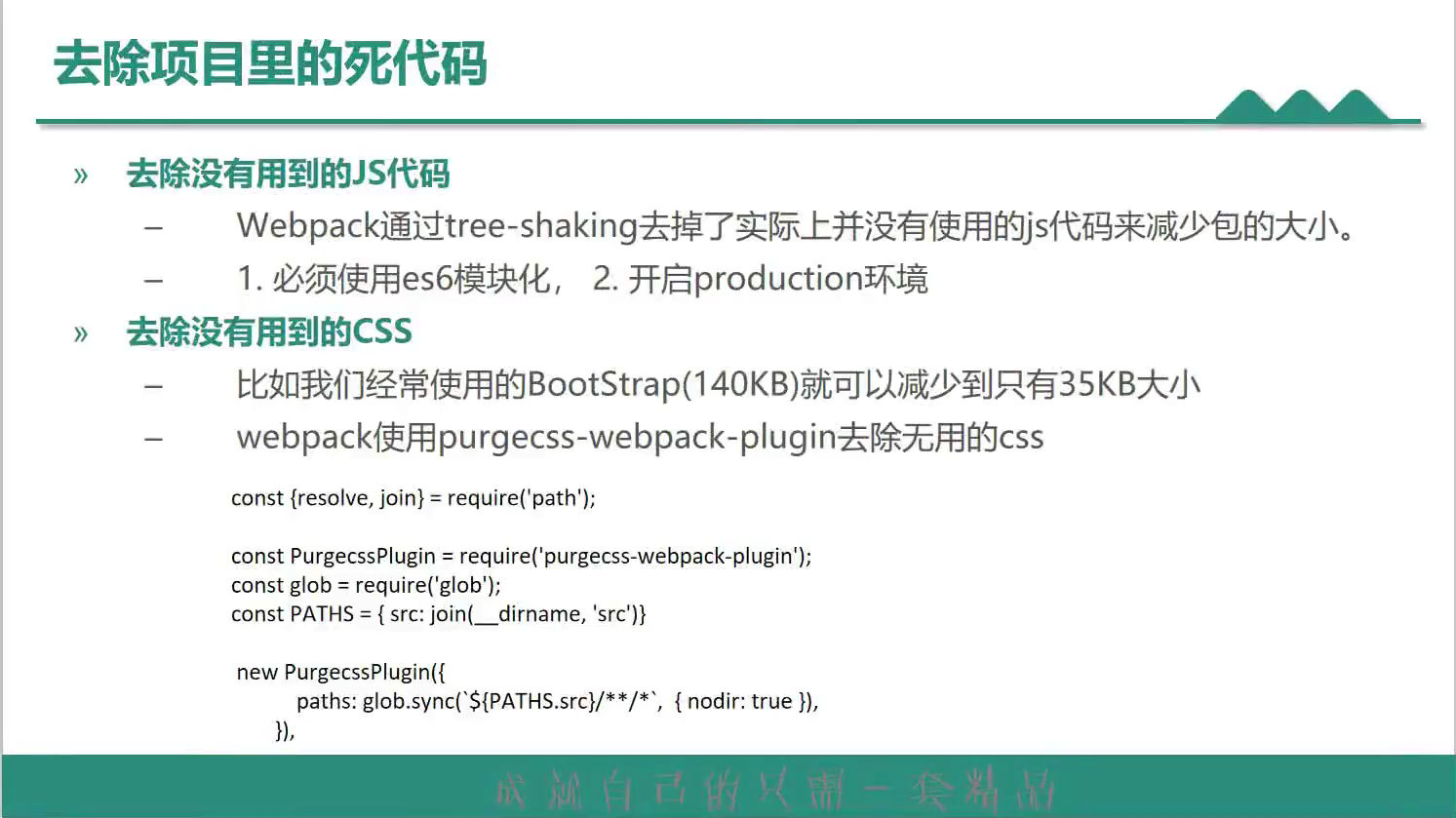
## 优化



## 热更新



## 去除死代码



# 项目指导

## 前后端解决跨域：

后台服务器： http://127.0.0.1:3000（后台服务器）

VUE脚手架前端服务器： http://127.0.0.1:8080（前端服务器）

http://123.14.66.9:34789 —> https://123.14.66.9:34789

http和https也是两个地址，会产生跨域问题

vue服务器 -> 后台服务器

**跨域： 2个不同地址的服务器相互访问，就会产生跨域问题**

**（不同地址，IP,协议，端口 只要3者有1个不一致，就会产生JS的跨域问题 ）**

浏览器施加的安全限制！ 服务器不允许其他IP地址来请求我的接口！！

前后端分离绝对会遇到的问题

每一个网站你觉得它最重要的是什么？？？用户数据（各种数据）

假设我们现在知道淘宝的获取所有用户接口是

www.taobao.com/getuserinfo

//假设没有跨域，那么只要你知道任何一个服务器的IP+接口，那我们就可以拿到此网站所有的数据！

### 如何解决（现代方案）：

#### 后台解决：

(一般在公司解决跨域都是后台来设置请求头，允许哪些IP来访问)：

设置请求头，允许指定IP或所有用户来访问自己！

//设置跨域访问(后台有安全协议)

app.all('\*', (req, res, next) => {

res.header("Access-Control-Allow-Origin", "\*"); //参数2：允许哪些IP地址来访问我，\*表示允许所有

res.header("Access-Control-Allow-Headers", "X-Requested-With");

res.header("Access-Control-Allow-Methods","PUT,POST,GET,DELETE,OPTIONS"); //允许别人给我发送什么类型的请求

res.header("X-Powered-By",' 3.2.1')

res.header("Content-Type", "application/json;charset=utf-8");

next();

});

#### 前端解决：

最早的解决方案jsonp(最早期原生js和jq的解决方案，已淘汰)

ajax有跨域，但是标签是没有跨域的，也就意味着，标签可以拿到任意网站的内容

JSONP： 用A网站的script标签去访问 B网站的js内容(利用标签没有跨域实现的)

vue项目解决方案（设置请求代理）

3.x和4.x在根目录创建vue.config.js文件，这个配置文件在运行项目的时候自动加载。

1.x和2.x项目默认有此文件

// vue.config.js

module.exports = {

devServer: {

proxy: {

'/api': {

target: 'http://22.163.72.64', //对应服务器地址

changeOrigin: true, //允许跨域

ws: true,

pathRewrite: {

'^/api': ''

}

}

}

}

}

发送请求

设置之后，发送请求时，将/api 当做域名就可以了。 例：

axios.get("api/test/login")

.then((res) => {

console.log(res)

})

这个就相当于

axios.get("http://22.163.72.64/test/login")

.then((res) => {

console.log(res)

})

## 兼容性

### Jq兼容：

jquery的版本说明：

1.x 支持IE6~8 兼容性比较好

2.x 不支持IE6~8 体积小

3.x 支技IE9以上版本 炫酷的网站

实际中：

学校，医院，政府===》使用 1.x

其它只关注炫酷效果的网站，可以在2.x或3.x选用

### 判断浏览器版本

**//通过navigator判断用户浏览器是什么类型的**

var mycss = document.getElementById('mycss')

var userAgent = navigator.userAgent;

// 获取用户的版本信息

//如果要做多浏览器，就需要每个浏览器都去判断，因为大家的版本，和不支持的版本都不同！！！

**// 火狐（怎么知道内核是不是火狐）**

if (userAgent.indexOf('Firefox') != -1) {

mycss.href = './common.css';

checkWebKitVersion(userAgent, 'Firefox', 20);

}

**// IE 5-10 是时候表演IE判断了**

else if (navigator.userAgent.indexOf('MSIE') != -1) {

mycss.href = './IE.css';

// IE 567提示IE内核过低！！！请升级！！！！！

var str = navigator.userAgent.substr(navigator.userAgent.indexOf('MSIE') + 5);

// 7.0; Windows NT 10.0; WOW64; Trident/7.0; .NET4.0C; .NET4.0E; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

var v = str.slice(0, str.indexOf('.'));

if (v < 8) {

document.getElementById('mybody').innerHTML = '您当前IE浏览器版本过低，请升级到Chrome继续访问此网站！';

}

}

**// 国标版(360 猎豹 QQ Chrome uc 等等..) 国产手机用的基本都是国际版内核 IOS用的是safiri**

// 因为平行兼容，所以可以统一获取一个版本

else {

mycss.href = './common.css';

checkWebKitVersion(userAgent, 'Chrome', 60); //现在此网页猎豹浏览器不行 Chrome可以 IE567不可以

}

**// IE 11**

// else if(userAgent.includes('Gecko') && !userAgent.includes('KHTML')){

//这个版本和国标版差别不大，一般不用特殊判断

// }

// // 如果还有公司要求兼容的特殊浏览器单独书写不同的浏览器的判断

// 检查浏览器内核版本，提示是否升级

//参数1： 当前浏览器的userAgent， webkitname： 当前判断内核的名字 version：对应的版本

function checkWebKitVersion(userAgent, webkitname, version) {

var arr = userAgent.split(' ');

// console.log(arr)

for (var i = 0 ; i < arr.length; i++) {

var str = arr[i];

//怎么知道是chrome的那一串字符串？？？

if (str.indexOf(webkitname) != -1) {

//表示str是我们想要的！ Chrome/49.0.3945.130 Firefox/72.0

//如果版本<50，直接提示用户升级浏览器，否则就正常显示

if (parseInt(str.slice(webkitname.length + 1, str.indexOf('.'))) < version) {

//直接提示用户升级浏览器

document.getElementById('mybody').innerHTML = '您当前浏览器版本过低，请升级以便继续使用！';

// return

}

}

}

}

## Git使用

GIT：版本控制工具 提交更新合并代码等...

GITHUB： 一个免费的云仓库，可以帮我们托管项目代码 （私人仓库GITHUB要收费）

### 设置自己的名字和邮箱

git config --global user.name xxxxxx

git config --global user.email xxxx@qq.com

### 在github上创建一个项目容器Create a new repository

登录以后，点击new新建一个项目库

输入项目名 选择public即可

### 生成本地SSHkey，等会和github绑定到一起

ssh-keygen -t rsa -C 邮箱地址

### 根据生存路径，打开.ssh文件夹

打开 id\_rsa.pub文件 全选复制其中所有内容

默认地址：**C:\Users\Administrator\.ssh**

### 点击GITHUB用户头像，点击下方Setting设置

点击SSH and GPG keys选项，把刚才复制的id\_rsa.pub文件内的所有内容，粘贴到SSHkey中

新建一个SSH key，Title随意， key就是刚才复制的值

为了安全性， 绑定了SSH key以后， 当前拥有key的电脑，才可以提交和修改github上的内容

### 提交项目到github仓库, 选中项目后， 右键git gui

先commit提交到本地服务器 再push推到github

复制github仓库的https地址， 在push时，把地址粘贴过去

弹框输入github用户名和密码， 就可以开始推送

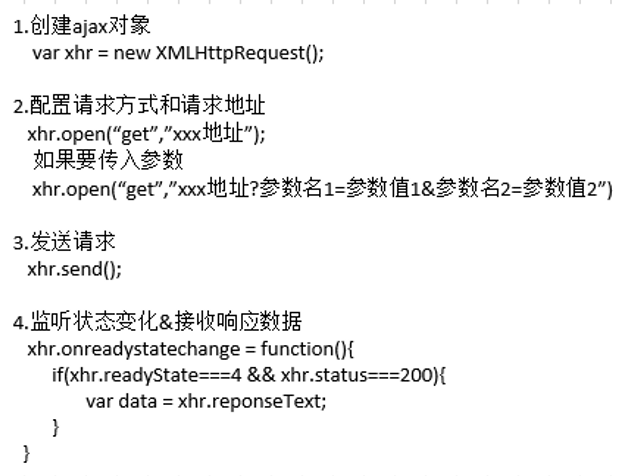
commit: 把代码提交到本地服务器

push: 把本地服务器的代码推送到远程的任何一个地址

## ajax（原生和jq）

### 原生

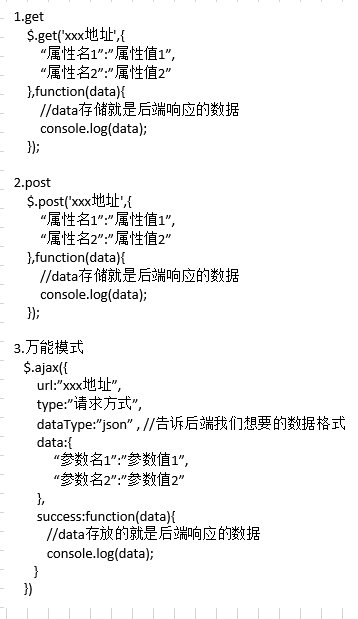
#### 原生Get发送请求



#### 原生post请求



### Jq 发送请求



## 本地存储（重点）

cookie

容量 4kb ，和服务端是有交互的(每次请求客户端会带上cookie信息) ，session基于cookie实现

sessionstorage

容量 5M

获取：sessionstorage.getItem('属性名')

设置：sessionstorage.setItem('属性名','属性值'); //属性值是string类型

删除：sessionstorage.removeItem('属性名');

清除: sessionstorage.clear();

生命周期：sessionstorage存在于页面，如果页面关闭，就会消失。

localstorage

容量 5M

获取：localstorage.getItem('属性名')

设置：localstorage.setItem('属性名','属性值'); //属性值是string类型

删除：localstorage.removeItem('属性名');

清除: localstorage.clear();

生命周期：永久存储，除非主动删除。

# 公司组成：

我们公司大概有12个人左右；

**产品部**：一个产品经理，

**技术部**（我部）：

一个项目经理

前端：3人

主程：一人（解决我们不会的问题，大哥）（1人）

小弟 前端我与另外一个人（2人）

后台：6人

主程：一人 （1人）

组长好像没有（不太清楚）

小弟 （5人）

**美术部**：1人

Ui 小哥哥（不是女的） (1人)

**测试部：**2人

小弟 (2人)

#### 个人优势：

我感觉自己体质还行，不要看着瘦，但是基本上不会感冒。我比较跟人合得来，跟什么人交流都没问题。个人不太喜欢熬夜，每天晚上玩游戏、看小说，逛论坛也会到12点睡觉（就是无聊，没有早睡的好习惯）。个人习惯逛逛我们前端的一些技术论坛：掘金，csdn，简书等等。就对新技术多了解一点。个人喜欢科技数码，看看发布会，评测，各论坛帖子。对物联网有相当高的热情。