客户端研发

岗位描述

1.带领云信客户端团队，负责云信IM SDK和音视频SDK的开发和维护

2.负责iOS、android、pc、web等客户端的架构演进

3.建设卓越高效的客户端团队

岗位要求

1.计算机相关专业，本科及以上学历，6年以上研发工作经验，1年以上管理经验

2.精通跨平台C/C++开发，熟练掌握Objective-C或Java

3.深刻理解面向对象设计原则，熟悉常用设计模式

4.精通tcp/ip网络，熟悉音视频技术栈，熟悉WebRTC者优先

5.有良好的的学习和沟通能力，有较强创新能力和责任感

6.熟悉团队协作模式，有良好的产品意识，能协调产品人员推进项目进程

**高级/资深Web前端开发工程师**

**岗位描述**

1、负责网易云相关web前端设计和实现；  
2、负责web前端功能开发和重构，保持良好的代码可维护性、可扩展性，高效的前端性能和良好的浏览器兼容性；  
3、负责web前端新技术调研和在产品中的推广使用, 提升web前端质量和团队的开发效率。

**岗位要求**

1、计算机及相关专业本科以上学历,1年以上web前端开发经验（资深要求3年以上工作经验）；  
2、熟练掌握前端相关技术(HTML/CSS/JavaScript/HTML5/CSS3...)，了解http协议以及相关开发调试工具；  
3、熟练掌握一个前端框架（Vue,React,Angular…）；  
4、对web服务器端开发（nodejs、java）有一定的了解和实践，对web系统安全有一定的了解；  
5、良好软件工程思想，良好的编程能力和编程习惯；  
6、具备较强的逻辑分析、解决问题能力，具备良好学习能力和对新技术的追求精神；  
7、工作认真负责，乐观开朗，善于团队合作。

**前端开发实习生**

**岗位描述**

1、负责有道数学前端开发工作；  
2、前沿技术调研，包括前端框架、h5游戏框架等；  
3、同ios/android开发人员合作，定义并实现高效的hybrid app内部接口；  
4、同后端开发人员合作，获取并展示数据；   
5、保持高效的前端性能，优秀的代码可维护性。

**岗位要求**

1、有一定的前端开发经验；   
2、熟悉前端相关技术(html/css/js)，了解http协议以及相关开发调试工具；   
3、掌握或了解一定的前端框架(react/vue/angular/...)；  
4、用过node.js；  
5、加分项：了解h5游戏框架、了解hybrid app的webview环境、有后端开发经验；  
6、工作认真负责，乐观开朗，善于团队合作；   
7、每周可实习四天及以上，共可实习三个月以上。

**网易云音乐资深前端开发工程师**

**岗位描述**

云音乐作为网易的核心产品，目前已经超4亿用户和数千万日活，还在保持继续快速增长。对于技术而言，最大的感触就是，周围有着一群同样靠谱的策划、市场等同事，让你没有「瞎忙乎的感觉」，所以真诚的邀请你加入这个团队。  
网易云音乐技术团队负责 Web端、移动webview、PC & MAC客户端以及大量中后台系统的开发。你将...  
1、负责或参与各类型项目（Web&Webview、中后台系统、混合应用、小程序、创意活动及游戏）的前端开发工作，完成系统设计、技术选型、模块开发工作。  
2、以不断提升质量、效率、体验为目的，封装组件、沉淀文档、生产工具、搭建系统, 享受工程师的日常  
3、分享自己日常的所见所想，引导同事共同成长，营造积极健康的技术氛围。

**岗位要求**

1、精通各种Web前端技术和标准(Javascript、HTML、CSS)，熟悉页面布局常用解决方案，对表现与数据分离、Web语义化等有深刻理解。  
2、熟练掌握一个数据驱动视图的框架（ React, Angular, Vue...）。  
3、熟练掌握常用的前端构建工具（webpack等），并具有成熟的模块化开发思维。  
4、熟悉常用前端数据管理的解决方案（如Redux、Mobx等），并清楚它们的优劣和应用场景。  
5、熟悉HTTP协议，并掌握相关网络调试工具，有服务端开发的背景常识。  
6、对重复性或不规范的工作容忍度低，能自发通过沟通或技术手段解决。

加分项  
1、丰富的跨端( PC & IOS & Android )开发经验，有实际跨端技术融合的实践经验。  
2、扎实的Node基础和功底，对Node异步编程、进程管理通信、Stream应用等有自己的实践经验，有大规模高可用的Node生成环境落地经验.   
3、对至少一个较大型前端框架有超越文档指南的原理性了解，能在更高层面分析框架的内在技术选型  
4、基于Web的爆款创意活动&游戏实践经验，有相关框架或解决方案的产出。  
5、主导或以核心贡献者角色参与过优质开源项目，对技术社区运营有深入的思考见解。  
6、有大厂成熟大前端团队的工作经验，参与开发或熟练应用内部沉淀的基础设施，能清楚它们的改进点更好。

**前端开发工程师**

**岗位描述**

1. 负责有道云笔记webapp前端开发  
2. 追求最佳用户体验，工作认真细致，有责任心，富激情。

**岗位要求**

1. 具有2年以上前端开发经验，大学本科及以上学历，计算机相关专业优先；精通web前端技术，熟悉W3C标准，对表现与数据分离、Web语义化（h5标签）等有深刻认识，  
2. 理解各主流浏览器间的兼容性问题，熟练使用常用前端开发与调试工具；  
3. 了解前端行业的最新资讯，至少熟悉一种以上前端开发框架或库，如：angular, react、vue、zepto等, 习惯 es6（es7）语法，熟悉基于 webpack+npm script 的开发模式；  
4. 具备较强的逻辑思维能力、沟通能力、团队合作能力  
加分项：  
给较大的开源项目贡献过代码，或发布过较大的开源项目者优先

现在是2018年7月，距离我正式写简历和找工作还有13个月左右的时间，在这段时间里一方面要正常维持实验室工作，另一方面要完成以下几点：

1. 熟练掌握前端相关技术(HTML/CSS/JavaScript/HTML5/CSS3...)，了解http协议以及相关开发调试工具，理解各主流浏览器间的兼容性问题；

2. 熟练掌握一个前端框架（Vue,React,Angular…），看要求掌握其中一个即可；这个我补充下，因为Vue和React语法很相近，我觉得还是都学习下比较好；

3. 熟悉W3C标准，对表现与数据分离、Web语义化等有深刻认识；（？）

4. 熟练掌握常用的前端构建工具（webpack等），并具有成熟的模块化开发思维；

5. 精通tcp/ip网络，掌握C/C++，熟练掌握Objective-C或Java;

6. 习惯 es6（es7）语法，熟悉基于 webpack+npm script 的开发模式；

7. 掌握一门数据库语言，我选择MySQL；

8. 掌握Node.js;

9. 熟练使用Git；

10. 熟悉常用前端数据管理的解决方案（如Redux、Mobx等）。

其中1-4为重点，我的近期目标是在9月之前看完大红书，完成并复习初中高算法题，复习html&css基础知识并完成网校的一两个小项目。

<time datatime=”2018-8-10”>2018/8/10</time>

完成了算法题，接下来准备把大红书看一遍，然后看数据结构与算法

关于项目我的想法是做一个小游戏集合的网站，网站可以登录和注册用户，用户信息保存在数据库，网站首页是一个ui界面，鼠标滑到界面左边会出来一个日历，界面上面一栏是功能栏，包括用户注册和登录，游戏选择，手气不错（任意跳出一个游戏），学习栏（我的学习博客），主界面是我的网站封面和logo，下滑是今日游戏推荐（三个一横排）

因为真的一点也不会做！所以我决定前面的小游戏先从网上学习（代码非原创会注明），然后我想自己实现换装小游戏（用像素点格子）、解密小游戏（素材网上找）、抓小人（小人会满屏幕乱跑，限定时间抓人，这个我想做一个积分榜）、敲砖块（我喜欢玩……）

也不知道能不能做出来，看看吧，先学习下登录界面的设计

今天阅读了下框架推荐，先尝试vue+bootstrap，先做多页面SPA，即上方索引栏跳转游戏，再做单页面，即单个网页的下滑和本页跳转功能，最后实现搜索功能，再进行seo优化（？），这周把页面大体结构和框架做下。

<time datatime=”2018-9-10”>2018/9/10</time>

数据结构和算法一书已经阅读完毕,算法的学习值得探究的很多，暂时不再深入了

很多算法如果能够可视化展示会更加容易理解，我想之后把排序算法和后面难度较高的算法进行可视化实现

今年12月左右进行复习，把之前freecodecamp上的算法再写一遍并放到我的github中

接下来复习下html/css，很久没学都忘得差不多了……最好能一边复习一边实践下，仿着网上的看看

<time datatime=”2018-9-18”>2018/9/18</time>

这一周把《Head First HTML&CSS》这本书复习了，这本书虽然很多点没有细挖但确实适合初学者学习，很多遗忘的点看过也能补上，但还是要以W3C标准为主，很多元素书中并没有详细说明，另外需要特别注意html5新增的元素（标签），下面是我的摘录：

**概要**

**1. HTML5 简介：** （HTML5 是目前-2017年来说，是最新的一代HTML标准）

**2. HTML5 一些新的规则：**   
新特性应该基于 HTML、CSS、DOM 以及 JavaScript。   
减少对外部插件的需求（比如 Flash）   
更优秀的错误处理   
更多取代脚本的标记（新增了一些语义化的标签： 在HTML4当中，很明显的一个形象就是用DIV来进行任何的布局。）   
HTML5 应该独立于设备 （对于各个移动设备说来，也是完美兼容的）   
开发进程应对公众透明 。

Web语义化：（分为html语义化（给机器看）和css语义化（给人看），另外还有一个url语义化）

语义化优点：

1. 去掉或者丢失样式的时候能够让页面呈现出清晰的结构

2. 有利于SEO：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：爬虫依赖于标签来确定上下文和各个关键字的权重；  
  
3. 方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）并渲染网页；  
  
4. 结构清晰，便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

<https://juejin.im/entry/5ab5f229518825558a069304>

**3. 新特性：**   
HTML5 中的一些有趣的新特性：   
新的特殊内容元素，比如 article、footer、header、nav、section   
新的表单控件，比如 calendar、date、time、email、url、search   
用于媒介回放的 video 和 audio 元素   
用于绘画的 canvas 元素   
对本地离线存储的更好的支持（我们讲JS的时候，会讲到技能点）

**4. 兼容性情况：** 针对浏览器的支持（除了IE6/IE7/IE8/IE9以为，其他的大部分主流的浏览器都是完美兼容的）

**5：HTML5新增标签（常用的）**   
**如何去理解标签：**   
（1） 基本语境（语义化），用在场景是什么？   
（2） 默认样式、是否块/行内元素、是否是闭合元素。

1. **header** 页眉（**网页（部分区域）的头部 顶部 导航区域等等**），**闭合标签**；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；
2. **nav** **导航链接部分；闭合标签**；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟 DIV标签，可以说是完全一样的特性。
3. **section** 标签定义网页中的区域（部分）。比如**章节、页眉、页脚或文档中的其他部分**。**闭合 标签**；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的 特性；
4. **footer** 页脚（网页（部分区域）的底部|版权区域等等），闭合标签；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；
5. **article** 内容是引用其他地方的。一个区域中的，另外一部分内容；闭合标签；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的特性
6. **aside** 跟 **article** 是一起使用；是辅助 **article** 区域的内容。也可以理解为整个网页的 辅助区域；（最常见的京东的右侧的工具栏） （边栏）
7. **hgroup** 给标题分组，为标题或者子标题进行分组，通常与h1-h6元素组合使用。如果文章中只有一个标题，则不使用hgroup。   
   **闭合标签；**块元素；默认的宽是：100%；高：内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；
8. **figure** 对多个元素进行组合。通常与**figcaption**联合使用。**闭合标签**；块元素；默认的宽是：100%；高：内容的高度；实质上跟DIV标签，可以说是完全一样的特性；figcaption 定义figure元素组的标题，必须写在figure元素中。一个figure元素内最多只允许放置一个figcaption元素。闭合标签；块元素；默认的宽是：100%；高：内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；
9. **audio** 播放声音文件，比如音乐或其它音频流。可以在开始标签和结束标签之间放置文本内容，这样老的浏览器就可以显示出不支持该标签的信息。闭合标签；行内元素；默认的宽:controls的宽度300px；高:controls的高度32px；autoplay autoplay 如果出现该属性，则音频在就绪后马上播放。controls controls 如果出现该属性，则向用户显示控件，比如播放按钮。preload preload 如果出现该属性，则音频在页面加载时进行加载，并预备播放。如果使用 “autoplay”，则忽略该属性。src url 要播放的音频的 URL。
10. **video** 播放视频文件，比如电影或其它视频流。可以在开始标签和结束标签之间放置文本内容，这样老的浏览器就可以显示出不支持该标签的信息。闭合标签；行内元素；默认的宽：300px 高：150px。autoplay autoplay 如果出现该属性，则视频在就绪后马上播放。controls 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。loop loop 如果出现该属性，则当媒介文件完成播放后再次开始播放。preload preload 如果出现该属性，则视频在页面加载时进行加载，并预备播放。如果使用 “autoplay”，则忽略该属性。src 要播放的视频的 URL。width 设置视频播放器的宽度。
11. **source**为媒介元素（比如 video 和 audio）指定多个播放格式与编码，浏览器会自动选择第一个可以识别的格式。非闭合标签，只有开始标签，没有结束标签。source 行内元素，默认无宽度和高度。media 定义媒介资源的类型，供浏览器决定是否下载。src 媒介的 URL。type 定义播放器在音频流中的什么位置开始播放。默认，音频从开头播放。
12. **canvas**定义图形，比如图表和其他图像。   
    闭合标签；行内元素；默认情况下，canvas创建的画布宽：300px;高：150px;
13. **datalist** 定义可选数据的列表。与 input 元素配合使用，就可以制作出输入值的下拉列表。   
    闭合标签；行内元素；默认无宽度和高度。
14. **embed** 定义嵌入的内容，比如插件。用来插入各种多媒体，格式可以是MIDI、MP3等。   
    非闭合标签，只有开始标签，没有结束标签。行内元素；默认的宽：300px；高：150px。
15. **time** 定义日期或时间，或者两者。闭合标签；行内元素，默认情况下，宽：内容的宽度；高：内容的高度；
16. **address** 为文档或 section 定义联系信息，比如：电子邮箱、地址、电话、QQ、微信号等。   
    闭合标签；块元素；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；
17. **map**定义客户端的图像映射。图像映射是带有可点击区域的图像。   
    闭合标签；行内元素；默认情况下，无宽度和高度；
18. **area** area 元素永远嵌套在 map 元素内部。area 元素可定义图像映射中的区域。   
    闭合标签，行内元素；只有结束标签，没有开始标签。默认情况下，无宽度和高度；
19. **mark**定义页面中需要突出显示或高亮显示的内容，通常在引用原文时，使用此元素，目的就是引起当前用户的注意。闭合标签；行内元素；默认情况下，宽：内容的宽度；高：内容的高度；
20. **details** 标签定义元素的细节，用户可进行查看，或通过点击进行隐藏。(备注信息)， 块元素   
    ；默认的宽是：100%；高： 内容的高度；实质上，跟DIV标签，可以说是完全一样的特性；   
    但是有一个动态的效果，点击可以显示(展开)内容，再点击可以隐藏(收起)内容；   
      
      
    **1-12 的标签一定要好好掌握； 多媒体（音乐、视频、图形）是结合 js 来做项目的。**

**支持HTML5浏览器默认的标签样式：**

article, aside, footer, header, hgroup, main, nav, section {   
display: block;   
}

但对于这些标签也不要滥用，如<section>和<div>标签不能等同，<section>元素主要适用同一主题内容的整合，一般会带有一个<header>标题，<div>则适用不同内容的整合。

关于新增标签的滥用，可以参考：

https://www.cnblogs.com/yingzi1028/p/7819099.html

<http://www.wangdahai.cn/1008.html>

<time datatime=”2018-9-28”>2018/9/28</time>

阅读学习了《图解HTTP》，很多没有记住，之后复习。

<time datatime=”2018-10-9>2018/10/9</time>

观看视频<https://www.youtube.com/watch?v=82hnvUYY6QA>，学习AJAX = Asynchronous JavaScript and XML，这里出来了一个很神奇的概念叫做Vanilla JavaScript，还有网站上骗你说这是世界上最轻量级的JS框架（没有之一），废话，原生JS能不轻量……

视频讲得不错，基本跟着做一遍就能懂AJAX的基本工作方式了，顺带学习了VSCode和WampServer的用法，复习直接看源代码和教程概念即可，这里我发现我已经把之前学习的PHP和MySQL全都还给佛祖了……

<time datatime=”2018-10-10>2018/10/10</time>

观看视频<https://www.youtube.com/watch?v=z6hQqgvGI4Y>，学习了Vue框架的基础知识，包括一点点node.js的安装和使用，用的时候才发现其实我node装得很早，但装到一半就没有然后了？？我也忘记这回事了？？这个视频我就听懂了一半吧，后面设计到Vue Router的我就真的是不懂了，我在书签里保存了一些文章可以之后看着了解下Vue Router的相关知识和整个Vue文件的架构知识，Vue全家桶是实现SPA的利器啊，up在视频中其实就是做了一个简单的单页页面实例了……之后继续学习，我先把我之前那本《JavaScript DOM 编程艺术》看完，不过几天不见我都忘了之前看的了……

<time datatime=”2018-10-12>2018/10/12</time>

感觉我和没头苍蝇一样又慌又乱的，什么都想学，什么都没耐心，很不好啊……《JavaScript DOM 编程艺术》这本我还差DOM动画和H5新增标签没有看，其实我记性还行吧也没有看了就忘……虽然还没有学习jQuery，但我个人觉得它就是DOM动画的方便版本，用更少的代码实现同样的功能，所以我准备首先把这两个看了，然后把编程艺术和红书中涉及h5新增内容的部分看完，再去学习Vue框架。准备下周之前把第一点结束。

<time datatime=”2018-10-15>2018/10/15</time>

把《JavaScript DOM 编程艺术》和jQuery相关学习结束，下周要开始转组时间可能没那么多了，现在多看点，手机里还保存了一个jQuery的面试题，现在刚学完确实能答得出，之后可能就要复习了，其中第一道问$是什么，其实就是jQuery的简写，可以详见noConflict方法，jQuery 使用 $ 符号作为 jQuery 的简写，noConflict() 方法会释放对 $ 标识符的控制。

看了网易的前端面试题，其实对jQuery要求比较深的，我掌握的还是很浅啊，很多我觉得用不到的没有看，但还是要看的。

<time datatime=”2018-10-15>2018/10/15</time>

补充了下jQuery知识，在补充过程中了解了JSONP跨域数据请求，初步了解了下**[同源策略、跨域解决方案](https://www.cnblogs.com/rockmadman/p/6836834.html)/**[**浏览器的同源策略**](https://developer.mozilla.org/zh-CN/docs/Web/Security/Same-origin_policy)，说起来后者还是一个简单的SPA应用，可以学习下……这个问题后续也要继续学习，是个比较重要的点，今天只是简单了解了下。

[**jQuery jsonp跨域请求**](https://www.cnblogs.com/chiangchou/p/jsonp.html)

这个文章讲跨域请求还挺清楚的，jQuery ajax方式以jsonp类型发起跨域请求，其原理跟<script>脚本请求一样，因此使用jsonp时也只能使用GET方式发起跨域请求，如果用POST方式会发生错误。跨域请求需要服务端配合，设置callback，才能完成跨域请求。

从jQuery 1.5.0版本开始引入的一个新功能----deferred对象。这个功能很重要，未来将成为jQuery的核心方法，它彻底改变了如何在jQuery中使用ajax。为了实现它，jQuery的全部ajax代码都被改写了。具体见jQuery的[deferred对象详解](http://www.ruanyifeng.com/blog/2011/08/a_detailed_explanation_of_jquery_deferred_object.html)，不过估计后面还会更新，一切以官网http://api.jquery.com/category/deferred-object为准。

后又学习了菜鸟教程上的jQuery 延迟对象，即Deferred 延迟对象，它是通过调用 jQuery.Deferred() 方法来创建的可链接的实用对象，对比下阮一峰那篇写得更加好懂，教程就作为知识梳理和补充吧。阮一峰文章中介绍的jQuery.when()方法不是deferred对象的方法，但可以传入一个或多个deferred延迟对象，返回绑定when()方法的promise()对象。

jquery的deferred对象的done方法和then方法都能实现链式调用，但是他们的作用是有区别的，then方法中如果你传递的方法有返回值，那么他会传递给下一个链式调用的方法。而done方法与此相反，你传递的方法就算有返回值，done方法也不会把你的返回值传给下一个链式调用的方法的，

**话不多说，直接上实例：**

[?](https://www.jb51.net/article/130911.htm)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | var defer = jQuery.Deferred();  defer.done(function(a,b){    console.log("a = " + a+"b = " + b);    return a \* b;  }).done(function( result ) {    console.log("result = " + result);  }).then(function( a, b ) {    console.log("a = " + a+"b = " + b);    return a \* b;  }).done(function( result ) {        console.log("result = " + result);  }).then(function( a, b ) {    console.log("a = " + a+"b = " + b);    return a \* b;  }).done(function( result ) {    console.log("result = " + result);  });  defer.resolve( 2, 3 ); |

**输出结果如下：**



**结果分析：**

1、第一个done和第二个done都返回了defer.resolve( 2, 3 )

2、done中callback的返回值不会被传递

3、第二个done只有一个参数，接收了defer.resolve( 2, 3 )的第一个参数2，所以result是2

4、第一个then接收defer.resolve( 2, 3 )，接收两个参数，result是6，同时新建一个deferred object，传递result给deferred object

5、第三个done接收到了这个新的deferred object和传递的result，打印结果是6，并把这个新的deferred object传递给第二个then

6、第二个then现在接收新的deferred object，它只有一个参数，是result，所以参数b没有定义，返回的结果是NaN，同时又新建一个deferred object

7、第四个done接收一个新建的deferred object，传递的参数是NaN，打印的结果自然就是NaN

以上这篇jquery中done和then的区别(详解)就是小编分享给大家的全部内容了，希望能给大家一个参考，也希望大家多多支持脚本之家。

jQuery HTML/CSS方法中的难点

1）jQuery.cssHooks方法，基本没看懂，见2018/10/19学习日记

2）jQuery中.attr()和.prop()的区别

<https://api.jquery.com/attr/#attr-attributeName>官方API

<https://www.cnblogs.com/dolphinX/p/3348582.html>这篇讲得还算清楚

我个人觉得有以下两点：

A. jQuery中attr() 对应原生JS中的 setAttribute() / getAttribute ,prop() 对应原生JS中 DOM对象.property;前者设置的“属性”会在元素标签中显示，后者设置的“属性”存在于该元素对应DOM对象下。对于build-in（内置）属性，attribute和property共享数据，attribute更改了会对property造成影响，反之亦然，但是两者的自定义属性是独立的数据，即使name一样，也互不影响。

B. 返回值

<input id="test3" type="checkbox"/>

var t=document.getElementById('test3');

console.log(t.getAttribute('checked'));//null

console.log(t.checked);//false;

t.setAttribute('checked','checked');

console.log(t.getAttribute('checked'));//checked

console.log(t.checked);//true

t.checked=false;

console.log(t.getAttribute('checked'));//checked

console.log(t.checked);//false

对于checked类型，.prop返回的是true/false的布尔值，attribute取得值是HTML文档字面量值

在JQuery的API当中提到过，checked特性(attribute)不是对应他的checked属性(property)，特性（attribute）实际对应的是defaultChecked属性（property）而且仅用于设置复选框最初的值。  
checked特性（attribute）值不会因为复选框的状态而改变，而checked属性（property）会因为复选框的状态而改变。因此， 跨浏览器兼容的方法来确定一个复选框是否被选中，是使用该属性（property）

<a id="test4" href="#">Click</a>

var t=document.getElementById('test4');

console.log(t.getAttribute('href'));//#

console.log(t.href);//file:///C:/Users/bsun/Desktop/ss/anonymous.html#

对于一些和路径相关的属性，两者取得值也不尽相同，但是同样attribute取得是字面量，property取得是计算后的完整路径

其实这些难点通过对jQuery源码分析应该都是能看懂的，官网也提供了完整源码，建议之后还是自己下载看了琢磨下

<time datatime=”2018-10-19”>2018/10/19</time>

[**js中(function(){}()),(function(){})(),jQuery中$(function(){});之间的区别**](https://blog.csdn.net/stpice/article/details/80586444)

今天终于把那篇和[钩子函数.cssHook()](https://segmentfault.com/a/1190000013274020)有关的文章读完了，这篇文章追根溯源jQuery.css、jQuery.style，从而追寻到钩子函数及其包含属性，当然这篇我一半都没看懂，可以参考[jQuery源码解析（4）—— css样式、定位属性](https://blog.csdn.net/vbdfforever/article/details/51059440)来对jQuery源码进行分析，这个等到我之后进阶一点再看吧……

一道很经典的基础题，涉及了JS异步、作用域、闭包，想想如何才能让i按顺序输出

**for** (**var** i = 0; i < 5; i ++) {

setTimeout(**function**(){

console.log(i);

}, 0);

}

console.log(i);

//5,5,5,5,5,5

setTimeout(...0)所表达的意思是：等待0秒后（这个时间由第二个参数值确定），往消息队列插入一条定时器事件消息，并将其第一个参数作为回调函数；而当执行栈内同步任务执行完毕时，线程从消息队列读取消息，将该异步任务入栈，执行；线程空闲时再次从消息队列读取消息。

详见<http://blog.codingplayboy.com/2017/04/25/js_async/>，比较清楚地讲了Javascript异步机制。

<time datatime=”2018-10-23”>2018/10/23</time>

js中json对象与字符串互相转换

除了开源库json2.js外（JSON.stringify/parse），下面提供原生js方法：

一、json对象转换json字符串：

var printObj = typeof JSON != “undefined” ? JSON.stringify : function(obj){

var arr = [] ;

$.each(obj , function(key , value){

var next = key + “:”;

next += $.isPlainObject (value) ? printObj(val) : val ;

arr.push (next);

});

return “{“ + arr.join(“,”) + “}” ;

};

二、json字符串转换json对象：

1、

function strToJson(str){

var json = eval(“(“ + str + “)”);

return json;

}

2、

function strToJson(str){

var json = ( new Function("return " + str) )();

return json;

}

<time datatime=”2018-10-25”>2018/10/25</time>

[**Position属性四个值：static、fixed、absolute和relative的区别和用法**](https://www.cnblogs.com/theWayToAce/p/5264436.html)

定位为relative的元素脱离正常的文本流中，但其在文本流中的位置依然存在。无论父级存在不存在，无论有没有TRBL，均是以父级的左上角进行定位，但是父级的Padding属性会对其影响。

定位为absolute的层脱离正常文本流，但与relative的区别是其在正常流中的位置不再存在。若想把一个定位属性为absolute的元素定位于其父级元素内，只有满足两个条件：

　 第一：设定TRBL

 　第二：父级设定Position属性

学习了CSS实现居中

<https://css-tricks.com/centering-css-complete-guide/> ，这个网站讲得很好，分类细，跟着做一遍就能上手，值得多看几遍，算一个比较重要的CSS考点了

<https://segmentfault.com/a/1190000016389031> ，这个可以作为补充

flex布局学习，看阮一峰两篇文就行。

<http://www.ruanyifeng.com/blog/2015/07/flex-grammar.html>

<http://www.ruanyifeng.com/blog/2015/07/flex-examples.html>

<https://codepen.io/thrillbunny/pen/pxqELq>（一个筛子布局）

<http://www.css88.com/book/css/properties/flex/index.htm>（使用说明快速一览）

<time datatime=”2018-11-6”>2018/11/6</time>

距离上一次写日志都好久了……中途一直在看买的那本node.js实战，老实说这本书更像一个使用简介，写的还是不错的，涵盖的点也很多，但都是浅尝辄止，没有深入探究过，加上我其实node.js基础都没有，读起来很吃力，现在也不过学了70多页。

这两天学习gulp & webpack，吐下槽webpack实在是更新太快了，webpack4和前面几代用法差别很大，学习起来可真是有够费劲的。

<time datatime=”2018-11-15”>2018/11/15</time>

[ES6 中 Arguments 和 Parameters 用法解析](https://segmentfault.com/a/1190000006057291)（包括扩展操作符…，rest参数…，默认参数等es6新增的参数特性）

事件冒泡及其具体应用事件代理

[解析Javascript事件冒泡机制](https://blog.csdn.net/luanlouis/article/details/23927347)

事件冒泡终止的方法：

1）相应的处理函数内加入event.stopPropagation()，事件停留在本节点

2）event.target 引用了产生此event对象的dom 节点，而event.currrentTarget 则引用了当前处理节点，判定event.target===event.currentTarget, 两者相等，则执行相应的处理函数

3) 父节点统一处理事件，通过判断事件的发生地（即事件产生的节点），然后做出相应的处理

switch(event.target.id) case

[preventDefault()、stopPropagation()、return false 之间的区别](https://www.cnblogs.com/dannyxie/p/5642727.html)

事件代理的优点：代码简介，给浏览器的压力小

事件代理就是在祖先级DOM元素绑定一个事件，当触发子孙级DOM元素的事件时，利用事件流（事件冒泡）的原理来触发绑定在祖先级DOM的事件。

[**简述：js事件代理的7种方法**](https://www.cnblogs.com/caizhenqin/p/5862245.html)（实际上是讲了事件绑定的方法）

[**javascript事件代理(delegate)原理解析**](https://www.cnblogs.com/webhelper/p/5616477.html)

<time datatime=”2018-11-19”>2018/11/19</time>

前端模块化

前端模块规范有三种：CommonJs,AMD和CMD。  
CommonJs用在服务器端，AMD和CMD用在浏览器环境，随着node.js推广，CommonJs也逐渐用于浏览器环境。

主要介绍AMD和CommonJs。

AMD（define和require，异步）：<http://www.ruanyifeng.com/blog/2012/11/require_js.html>

CommonJs（module.exports和require，同步）：

<https://www.cnblogs.com/huiguo/p/7967241.html>

综述：[前端模块化（CommonJs,AMD和CMD）](https://www.jianshu.com/p/d67bc79976e6)

懒加载

**懒加载技术**：先只加载可视窗口区域的图片，当用户向下拖动滚动条时再继续加载后面的图片（也是只加载目前可视窗口区域内的图片）。1.这样减少了加载时的线程数量，使可视区域内的图片也能够快速加载，优化了用户体验。2.减少了同一时间发向服务器的请求数，服务器压力剧减。

<https://blog.csdn.net/Gytha_1/article/details/77892278>（集合多篇）

<https://blog.csdn.net/bury_/article/details/79516223>（API封装）

里面有两个函数写得可以：

1）数组化

function toArray(arrlike){

if(typeof Array.from !="function"){

var result=[];

for(let i=0;i<arrlike.length;i++){

result.push(arrlike[i]);

}

return result;

}else{

return Array.from(arrlike);

}

}

2）跨浏览器获取可视窗口大小

function getWindow () {

if(typeof window.innerWidth !='undefined') {

return{

width : window.innerWidth,

height : window.innerHeight

}

} else{

return { //IE 8 及更早 IE版本

width : document.documentElement.clientWidth,

height : document.documentElement.clientHeight

}

}

}

顺便讲下网页性能优化，因为

* 修改DOM
* 修改样式表
* 用户事件（比如鼠标悬停、页面滚动、输入框键入文字、改变窗口大小等等）

会导致网页重新渲染，需要重新生成布局和重新绘制，所以主要从DOM和CSS两方面进行性能优化

[网页性能管理详解](http://www.ruanyifeng.com/blog/2015/09/web-page-performance-in-depth.html)（阮一峰）

<time datatime=”2018-12-11”>2018/12/11</time>

设计模式

1）创建型设计模式

单例模式

<https://www.cnblogs.com/yonglin/p/8080836.html>

工厂模式

<https://segmentfault.com/a/1190000007790683>

<https://segmentfault.com/a/1190000009383232>

2）结构型设计模式

3）行为设计模式

<time datatime=”2018-12-12”>2018/12/12</time>

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title>$.extend()</title>

<script src="https://cdn.staticfile.org/jquery/1.10.2/jquery.min.js"></script>

</head>

<body>

<div id="log"></div>

<script>

$(function () {

var defaults = { validate: false, limit: 5, name: "foo" };

var options = { validate: true, name: "bar" };

/\*合并默认值和选项，不修改默认对象。\*/

var settings = $.extend({}, options, defaults);

var printObj = typeof JSON != "undefined" ? JSON.stringify : function(object){

var arr = [];

$.each(object, function(key, val){

var next = key + " : " + $.isPlainObject(val) ? printObj(val) : val;

arr.push(next);

});

return "{ " + arr.join(",") + " }";

};

$("#log").append( "<div><b>defaults -- </b>" + printObj(defaults) + "</div>" );

$("#log").append( "<div><b>options -- </b>" + printObj(options) + "</div>" );

$("#log").append( "<div><b>settings -- </b>" + printObj(settings) + "</div>" );

})

</script>

</body>

</html>

$.extend()的JS实现

<https://www.cnblogs.com/zfc2201/archive/2012/04/09/2439419.html>

从我的理解来看，浅复制是覆盖，深复制是合并，但这篇文章中对深复制的模拟让我觉得有点奇怪，感觉并没有做到同名属性的合并

Javascript对象的Get/Set方法

先看MDN上的解释：

When using **get** the property will be defined on the prototype of the object while using [Object.defineProperty()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty" \o "The static method Object.defineProperty() defines a new property directly on an object, or modifies an existing property on an object, and returns the object.) the property will be defined on the instance it is applied to.

这一点后面的几个链接都没有提到，这么重要的区别

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/get>

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Functions/set>

var expr = 'foo';

var obj = {

baz: 'bar',

set [expr](v) { this.baz = v; }//set foo(v){ this.baz = v; }

};

console.log(obj.baz); // "bar"

obj.foo = 'baz'; // run the setter

console.log(obj.baz); // "baz"

[JavaScript对象深入理解（defineProperty、get/set）](https://blog.csdn.net/doy_doy/article/details/80513467)（defineProperty添加的属性，四大特性默认值都是false）

[JavaScript get set方法 ES5/ES6写法](https://www.cnblogs.com/ljwTiey/p/7387598.html)（这里的区别是因为es6中新增了类似类与构造器）

[js对象，set和get方法 的三种实现形式](https://www.cnblogs.com/wangEddy/p/7827277.html)

**下面三种的完全一样的：**

1 var obj1 = {};  
2 var obj2 = new Object();  
3 var obj3 = Object.create(Object.prototype);

为了解释为啥这三种方式是完全一样的，我们先来解释下JavaScript中的原型对象（哎，让客官久等了！），记得一位大神说过：

Javascript是一种基于对象（object-based）的语言，你遇到的所有东西几乎都是对象。但是，它又不是一种真正的面向对象编程（OOP）语言，因为它的语法中没有class（类）。

面向对象的编程语言JavaScript，没有类！！！那么，它是怎么实现继承的呢？没错，就是通过原型对象。基本上每一个JavaScript对 象（null除外）都和另一个对象相关联，“另一个”对象就是所谓的原型对象（原型对象也可以简称为原型，并没有想象的那么复杂，它也只是一个对象而 已）。每一个对象都从原型对象继承属性，并且一个对象的prototype属性的值（这个属性在对象创建时默认自动生成，并不需要显示的自定义）就是这个 对象的原型对象，即obj.prototype就是对象obj的原型对象。

原型对象先说到这，回到上面的问题，有了对原型对象的认识，下面就是不需要过多解释的JavaScript语言规定了：

①所有通过对象直接量创建的对象的原型对象就是Object.prototype对象；

②通过关键字new和构造函数创建的对象的原型对象就是构造函数prototype属性的值，所以通过构造函数Object创建的对象的原型就是Object.prototype了；

现在也补充了第三种创建对象的方法Object.create()第一个参数的含义。

【通过关键字new和构造函数创建的对象的原型对象就是构造函数prototype属性的值，所以通过构造函数Object创建的对象的原型就是Object.prototype了；】这句话让我想起了之前背过的一张表，想不起来哪本书的了……要复习啊

[深入学习JavaScript对象](file:///C:\Users\lenovo\AppData\Roaming\Microsoft\Word\深入学习JavaScript对象%20-%20clearbug%20-%20博客园.pdf)

这篇文章写得很全很好，解释了很多容易混淆的概念

<time datatime=”2018-12-22”>2018/12/20</time>

js函数调用场合及this关键字指向（直接调用、对象方法调用、构造函数调用、apply/call改变函数的调用对象）

<https://www.cnblogs.com/xiangxiao/p/6822997.html>

执行上下文

<https://www.cnblogs.com/MinLee/p/5862271.html>

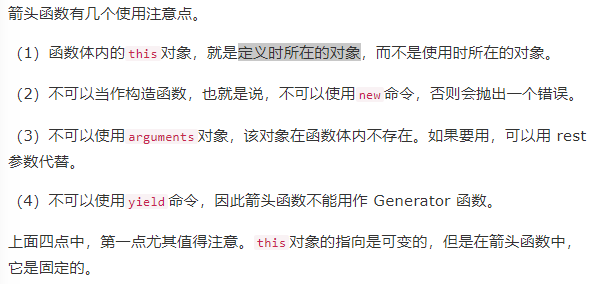
es6

箭头函数没有自己的this，它的this是继承而来，默认指向在定义它时所处的对象(宿主对象)

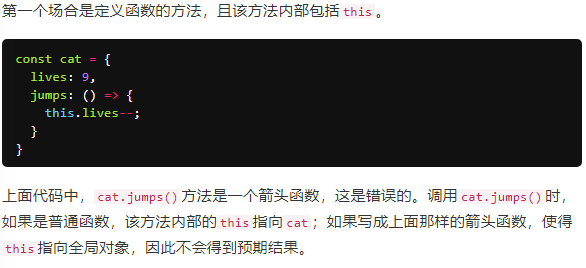
<https://www.cnblogs.com/fanzhanxiang/p/8888963.html>

[http://es6.ruanyifeng.com/#docs/function#%E7%AE%AD%E5%A4%B4%E5%87%BD%E6%95%B0（阮一峰es6](http://es6.ruanyifeng.com/#docs/function)箭头函数）

阮一峰文章摘录

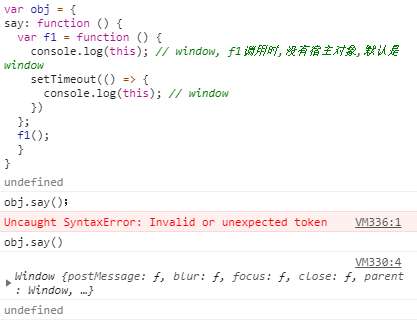


上面这张第一点：函数体内的this对象，是定义时所在的对象，牢记是函数体内，所以箭头函数在下面这种场合是不适用的



由此也能看出以下两张图的区别：





另一篇中写的：

　　箭头函数中没有this绑定，必须通过查找作用域链来决定其值。

　　箭头函数缺少正常函数所拥有的prototype属性，它的设计初衷是“即用即弃”，所以不能用它来定义新的类型。

　　箭头函数中的this值取决于该函数外部非箭头函数的this值，且不能通过call（）、apply（）或bind（）方法来改变this的值。

上面也是我个人的理解，感觉this要搞清楚还需要好好把对象、函数以及它们的关系搞清楚。

<time datatime=”2018-12-22”>2018/12/22</time>

MCV、MVP、MVVM图示

<http://www.ruanyifeng.com/blog/2015/02/mvcmvp_mvvm.html>

从MVC到MVVM（初识Vue）【不错的文章，实现起来有出入】

<https://www.jianshu.com/p/63fe81bd9671>

<time datatime=”2018-12-25”>2018/12/25</time>

[window.location.hash 使用说明](https://www.cnblogs.com/nifengs/p/5104763.html)

【Vue学习】todoList

[Vue之ToDoList实战](https://blog.csdn.net/wu__di/article/details/54918454)（简单介绍，没啥价值）

**重点**

<https://cn.vuejs.org/v2/examples/todomvc.html>（官网，写得很好，不过没讲css相关））

<https://blog.csdn.net/lhjuejiang/article/details/81040236>（写得也很好，和官网架构相同）

【Vue学习】Vue视频学习

1.基础概念

1）构建用户界面的框架，只关注视图层

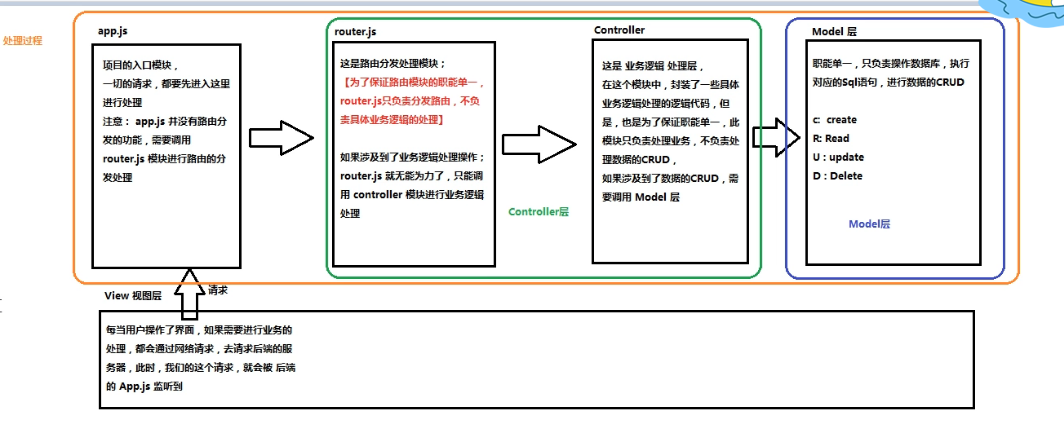
2）可借助Weex开发手机App

3）减少不必要的DOM操作，提高渲染效率

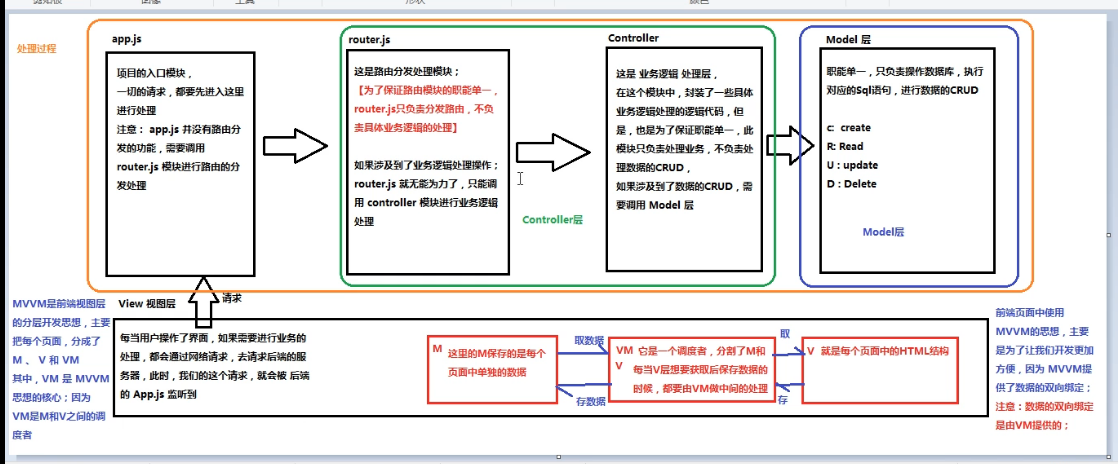
4）双向数据绑定

2.Node中的MVC和前端中的MVVM（这个说法我没在其他地方看到过）

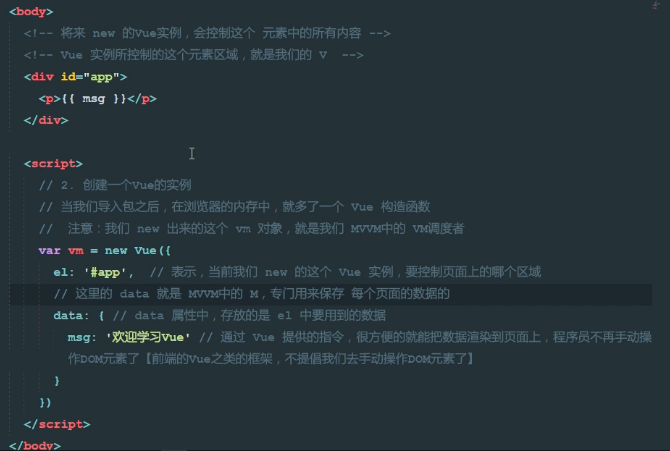
MVC一般是后端分层开发的概念



MVVM一般是前端视图层概念，主要关注于视图层分离



Vue代码与MVVM的对应关系



3.v-\*基本指令

1）v-text，v-html，v-cloak

v-cloak：防止页面加载时出现 vuejs 的变量名

html：

<div v-cloak>

{{ message }}

</div>

css:

[v-cloak] {

display: none;

}

2）v-bind/: Vue提供的用于绑定属性的指令

3）v-on/@：绑定事件

4）v-model：表单元素（input/select/checkbox/textarea等）数据的双向绑定（唯一可以实现双向数据绑定的指令）

实际上是个语法糖，综合了v-bind和v-on的功能

5）v-for

首先<https://www.jianshu.com/p/58a8866776bd>

这篇对js中可枚举属性作了简要介绍

[JS中的可枚举属性与不可枚举属性以及扩展](https://www.cnblogs.com/moqiutao/p/7389146.html)（主要讲了可枚举和不可枚举属性的修改和遍历）

A. v-for可以遍历数组、对象数组（v-for=”(item,index) in list”）、对象和数字（从1开始）；

B. 组件中使用v-for必须有key，原因摘自官网：

当 Vue.js 用 v-for 正在更新已渲染过的元素列表时，它默认用“就地复用”策略。如果数据项的顺序被改变，Vue 将不会移动 DOM 元素来匹配数据项的顺序， 而是简单复用此处每个元素，并且确保它在特定索引下显示已被渲染过的每个元素。这个类似 Vue 1.x 的 track-by="$index" 。

这个默认的模式是高效的，但是只适用于**不依赖子组件状态或临时 DOM 状态 (例如：表单输入值) 的列表渲染输出**。

为了给 Vue 一个提示，以便它能跟踪每个节点的身份，从而重用和重新排序现有元素，你需要为每项提供一个唯一 key 属性。理想的 key 值是每项都有的唯一 id。这个特殊的属性相当于 Vue 1.x 的 track-by ，但它的工作方式类似于一个属性，所以你需要用 v-bind 来绑定动态值 (在这里使用简写)：

|  |
| --- |
| <div v-for="item in items" :key="item.id">  <!-- 内容 --> </div> |

建议尽可能在使用 v-for 时提供 key，除非遍历输出的 DOM 内容非常简单，或者是刻意依赖默认行为以获取性能上的提升。

因为它是 Vue 识别节点的一个通用机制，key 并不与 v-for 特别关联，key 还具有其他用途，我们将在后面的指南中看到其他用途。

两个注意点：

key在使用时必须用v-bind绑定的形式，:key=””；key属性只能是数字或字符串。

6）v-if和v-show



7）样式和内联样式





4.Vue自定义过滤器

Vue.js允许自定义过滤器，过滤器可用在mustache插值和v-bind表达式。

全局过滤器

//语法，function的arg1永远是管道符传来的数据

Vue.filter(“过滤器的名称”，function(data){});

//调用格式

{{name | 过滤器的名称}}

私有过滤器

<time datatime=”2019-03-08”>2019/03/08</time>

[CSS3伪类和伪元素的特性和区别](https://www.cnblogs.com/ihardcoder/p/5294927.html)

伪类和伪元素在CSS2里面还没有分得太清，直到CSS3才正式区分，从我的理解看，伪类是为了增加选择器功能，很多DOM树中没有的元素可以通过伪类进行选择，伪元素则是主要进行修饰作用，功能都很强大

[::before和::after伪元素的用法](https://www.cnblogs.com/starof/p/4459991.html)（未看完，比较进阶）

[【CSS3】background-origin和background-clip的区别](https://www.cnblogs.com/shytong/p/5077129.html)

<time datatime=”2019-03-14”>2019/03/14</time>

[**关于margin、padding 对内联元素的影响**](https://www.cnblogs.com/enmeen/p/5286564.html)

**inline：**

width和height 可以看成物理属性，对内联元素没有影响。  能够影响它宽高的为其本身的内容。

margin-top:10px;margin-bottom: 10px;无效果

margin-left和margin-right;起作用。

padding属性就比较特别了。

padding-left;padding-right同样有效果,与块元素效果相同。

padding-top和padding-bottom不会影响它的高度，但是会影响他的背景高度。

<time datatime=”2019-03-15”>2019/03/15</time>

判断对象是否为数组？

* [JavaScript 判断对象是否为数组](http://www.runoob.com/note/26685)
* [JavaScript 判断该对象是否为数组](http://www.runoob.com/w3cnote/javascript-check-arrayisobject.html)

<time datatime=”2019-03-17”>2019/03/17</time>

**JavaScript 允许重复声明变量，后声明的覆盖之前的**

var a = 1;

var a = 'x';

console.log(a);

// 输出 'x'

**JavaScript 允许重复定义函数**

JavaScript 没有重载这个概念，它仅依据函数名来区分函数。

后定义的同名函数覆盖之前的，与参数无关。

function test() {

console.log("test");

}

test(); //输出 "test arg0 + undefined"

function test(arg1) {

console.log("test arg" + arguments.length + " + " + arg1);

}

test(1,2); //输出 "test arg2 + 1"

实参个数如果比形参少，那么剩下的默认赋值为 **undefined**，如果实参传的比形参数量多，那么是全部都会被传进去的，只不过没有对应的形参可以引用（但可以用 arguments 来获取剩下的参数）。

function test(arg1) {

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

console.log(arguments[i]);

}

}

test(1,2); //输出 1 2

**变量与函数重名的时候，变量生效**

这涉及到了变量和函数的预解析：

* 变量声明会被顶置，函数声明也会被顶置且比变量更先声明。
* 变量的声明和赋值语句一起写时，JS引擎在解析时，会将其拆成声明和赋值2部分，声明置顶，赋值保留在原来位置。
* 声明过的变量不会再重复声明。

var a = 100;

function a() {

return "function";

}

console.log(a); //输出 100

console.log(a());

/\*

报错

Uncaught TypeError: a is not a function

(anonymous function) @test.html:9

\*/

JS 中有两种函数，一种是普通函数，一种是函数对象。下面的这种就是“函数对象”，它实际上是声明一个匿名函数，然后将该函数的 init 方法赋值给该变量。

var a = 100;

var a = function() {

return "function";

}

console.log(a);

/\*

输出

function() {

return "function";

}

\*/

console.log(a()); //输出 "function"

**函数与内部变量重名**

定义普通函数，即在 window 变量下，定义一个 key，它的名字为该函数名，值为该函数的地址。函数内部的 this 指向 window 对象。

function a() {

console.log(this); //输出 window{...}

this.a = 1; //即 window.a = 1，此时window下的function a已经被该变量覆盖了。

var a = 5; //下面的这几个变量都是局部变量，仅在花括号范围内有效。

a = 10;

var v = "value"

return "function";

}

console.log(a); //输出 function a {...}

console.log(a()); //输出 "function"

console.log(a); //输出 1

console.log(v);

/\*

输出

Uncaught ReferenceError: v is not defined

(anonymous function) @ mycolor.html:15

\*/