**前端开发面试题基础篇答案**

**前言**

**前端开发知识点：**

HTML&CSS：

对Web标准的理解、浏览器内核差异、兼容性、hack、CSS基本功：布局、盒子模型、选择器优先级、

HTML5、CSS3、Flexbox

JavaScript：

数据类型、运算、对象、Function、继承、闭包、作用域、原型链、事件、RegExp、JSON、Ajax、

DOM、BOM、内存泄漏、跨域、异步装载、模板引擎、前端MVC、路由、模块化、Canvas、ECMAScript 6、Nodejs

其他：

移动端、响应式、自动化构建、HTTP、离线存储、WEB安全、优化、重构、团队协作、可维护、易用性、SEO、UED、架构、职业生涯、快速学习能力

作为一名前端工程师，**无论工作年头长短都应该掌握的知识点**：

此条由 王子墨 发表在 [攻城师的实验室](http://lab.yuanwai.wang/)

1、DOM结构 —— 两个节点之间可能存在哪些关系以及如何在节点之间任意移动。

2、DOM操作 ——如何添加、移除、移动、复制、创建和查找节点等。

3、事件 —— 如何使用事件，以及IE和标准DOM事件模型之间存在的差别。

4、XMLHttpRequest —— 这是什么、怎样完整地执行一次GET请求、怎样检测错误。

5、严格模式与混杂模式 —— 如何触发这两种模式，区分它们有何意义。

6、盒模型 —— 外边距、内边距和边框之间的关系，及IE8以下版本的浏览器中的盒模型

7、块级元素与行内元素 —— 怎么用CSS控制它们、以及如何合理的使用它们

8、浮动元素 ——怎么使用它们、它们有什么问题以及怎么解决这些问题。

9、HTML与XHTML ——二者有什么区别，你觉得应该使用哪一个并说出理由。

10、JSON —— 作用、用途、设计结构。

**备注：**

根据自己需要选择性阅读，面试题是对理论知识的总结，让自己学会应该如何表达。

**HTML**

* Doctype作用？标准模式与兼容模式各有什么区别?
* （1）、<!DOCTYPE>声明位于位于HTML文档中的第一行，处于 <html> 标签之前。告知浏览器的解析器用什么文档标准解析这个文档。DOCTYPE不存在或格式不正确会导致文档以兼容模式呈现。
* （2）、标准模式的排版 和JS运作模式都是以该浏览器支持的最高标准运行。在兼容模式中，页面以宽松的向后兼容的方式显示,模拟老式浏览器的行为以防止站点无法工作。
* HTML5 为什么只需要写 <!DOCTYPE HTML>？
* HTML5 不基于 SGML，因此不需要对DTD进行引用，但是需要doctype来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；
* 而HTML4.01基于SGML,所以需要对DTD进行引用，才能告知浏览器文档所使用的文档类型。
* 行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？
* 首先：CSS规范规定，每个元素都有display属性，确定该元素的类型，每个元素都有默认的display值，如div的display默认值为“block”，则为“块级”元素；span默认display属性值为“inline”，是“行内”元素。
* （1）行内元素有：a b span img input select strong（强调的语气）
* （2）块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p
* （3）常见的空元素：
* <br> <hr> <img> <input> <link> <meta>
* 鲜为人知的是：
* <area> <base> <col> <command> <embed> <keygen> <param> <source> <track> <wbr>
* 页面导入样式时，使用link和@import有什么区别？
* （1）link属于XHTML标签，除了加载CSS外，还能用于定义RSS, 定义rel连接属性等作用；而@import是CSS提供的，只能用于加载CSS;
* （2）页面被加载的时，link会同时被加载，而@import引用的CSS会等到页面被加载完再加载;
* （3）import是CSS2.1 提出的，只在IE5以上才能被识别，而link是XHTML标签，无兼容问题;
* 介绍一下你对浏览器内核的理解？
* 主要分成两部分：渲染引擎(layout engineer或Rendering Engine)和JS引擎。
* 渲染引擎：负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入CSS等），以及计算网页的显示方式，然后会输出至显示器或打印机。浏览器的内核的不同对于网页的语法解释会有不同，所以渲染的效果也不相同。所有网页浏览器、电子邮件客户端以及其它需要编辑、显示网络内容的应用程序都需要内核。
* JS引擎则：解析和执行javascript来实现网页的动态效果。
* 最开始渲染引擎和JS引擎并没有区分的很明确，后来JS引擎越来越独立，内核就倾向于只指渲染引擎。
* 常见的浏览器内核有哪些？
* Trident内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称MSHTML]
* Gecko内核：Netscape6及以上版本，FF,MozillaSuite/SeaMonkey等
* Presto内核：Opera7及以上。 [Opera内核原为：Presto，现为：Blink;]
* Webkit内核：Safari,Chrome等。 [ Chrome的：Blink（WebKit的分支）]

详细文章：[浏览器内核的解析和对比](http://www.cnblogs.com/fullhouse/archive/2011/12/19/2293455.html)

* html5有哪些新特性、移除了那些元素？如何处理HTML5新标签的浏览器兼容问题？如何区分 HTML 和 HTML5？
* \* HTML5 现在已经不是 SGML 的子集，主要是关于图像，位置，存储，多任务等功能的增加。
* 绘画 canvas;
* 用于媒介回放的 video 和 audio 元素;
* 本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失;
* sessionStorage 的数据在浏览器关闭后自动删除;
* 语意化更好的内容元素，比如 article、footer、header、nav、section;
* 表单控件，calendar、date、time、email、url、search;
* 新的技术webworker, websockt, Geolocation;
* 移除的元素：
* 纯表现的元素：basefont，big，center，font, s，strike，tt，u;
* 对可用性产生负面影响的元素：frame，frameset，noframes；
* \* 支持HTML5新标签：
* IE8/IE7/IE6支持通过document.createElement方法产生的标签，
* 可以利用这一特性让这些浏览器支持HTML5新标签，
* 浏览器支持新标签后，还需要添加标签默认的样式。
* 当然最好的方式是直接使用成熟的框架、比如html5shim;
* <!--[if lt IE 9]>
* <script> src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
* <![endif]-->
* \* 如何区分HTML5： DOCTYPE声明\新增的结构元素\功能元素
* 简述一下你对HTML语义化的理解？
* 用正确的标签做正确的事情。
* html语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析;
* 及时在没有样式CCS情况下也以一种文档格式显示，并且是容易阅读的;
* 搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，利于SEO;
* 使阅读源代码的人对网站更容易将网站分块，便于阅读维护理解。
* HTML5的离线储存怎么使用，工作原理能不能解释一下？
* 在用户没有与因特网连接时，可以正常访问站点或应用，在用户与因特网连接时，更新用户机器上的缓存文件。
* 原理：HTML5的离线存储是基于一个新建的.appcache文件的缓存机制(不是存储技术)，通过这个文件上的解析清单离线存储资源，这些资源就会像cookie一样被存储了下来。之后当网络在处于离线状态下时，浏览器会通过被离线存储的数据进行页面展示。
* 如何使用：
* 1、页面头部像下面一样加入一个manifest的属性；
* 2、在cache.manifest文件的编写离线存储的资源；
* CACHE MANIFEST
* #v0.11
* CACHE:
* js/app.js
* css/style.css
* NETWORK:
* resourse/logo.png
* FALLBACK:
* / /offline.html
* 3、在离线状态时，操作window.applicationCache进行需求实现。

详细的使用请参考：[有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617)

* 浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢？
* 在线的情况下，浏览器发现html头部有manifest属性，它会请求manifest文件，如果是第一次访问app，那么浏览器就会根据manifest文件的内容下载相应的资源并且进行离线存储。如果已经访问过app并且资源已经离线存储了，那么浏览器就会使用离线的资源加载页面，然后浏览器会对比新的manifest文件与旧的manifest文件，如果文件没有发生改变，就不做任何操作，如果文件改变了，那么就会重新下载文件中的资源并进行离线存储。
* 离线的情况下，浏览器就直接使用离线存储的资源。

详细的使用请参考：[有趣的HTML5：离线存储](http://segmentfault.com/a/1190000000732617)

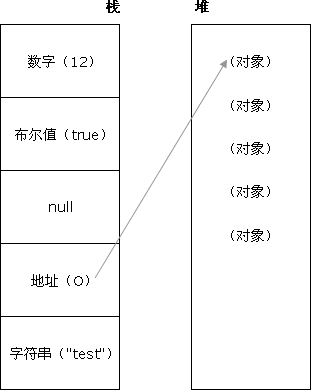
* 请描述一下 cookies，sessionStorage 和 localStorage 的区别？
* cookie是网站为了标示用户身份而储存在用户本地终端（Client Side）上的数据（通常经过加密）。
* cookie数据始终在同源的http请求中携带（即使不需要），记会在浏览器和服务器间来回传递。
* sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。
* 存储大小：
* cookie数据大小不能超过4k。
* sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，可以达到5M或更大。
* 有期时间：
* localStorage 存储持久数据，浏览器关闭后数据不丢失除非主动删除数据；
* sessionStorage 数据在当前浏览器窗口关闭后自动删除。
* cookie 设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭
* iframe有那些缺点？
* \*iframe会阻塞主页面的Onload事件；
* \*搜索引擎的检索程序无法解读这种页面，不利于SEO;
* \*iframe和主页面共享连接池，而浏览器对相同域的连接有限制，所以会影响页面的并行加载。
* 使用iframe之前需要考虑这两个缺点。如果需要使用iframe，最好是通过javascript
* 动态给iframe添加src属性值，这样可以绕开以上两个问题。
* Label的作用是什么？是怎么用的？
* label标签来定义表单控制间的关系,当用户选择该标签时，浏览器会自动将焦点转到和标签相关的表单控件上。
* <label for="Name">Number:</label>
* <input type=“text“name="Name" id="Name"/>
* <label>Date:<input type="text" name="B"/></label>
* HTML5的form如何关闭自动完成功能？
* 给不想要提示的 form 或下某个input 设置为 autocomplete=off。
* 如何实现浏览器内多个标签页之间的通信? (阿里)
* 调用localstorge、cookies等本地存储方式
* webSocket如何兼容低浏览器？(阿里)
* Adobe Flash Socket 、
* ActiveX HTMLFile (IE) 、
* 基于 multipart 编码发送 XHR 、
* 基于长轮询的 XHR
* 页面可见性（Page Visibility）API 可以有哪些用途？
* 在页面被切换到其他后台进程的时候，自动暂停音乐或视频的播放；
* 如何在页面上实现一个圆形的可点击区域？
* 1、map+area或者svg
* 2、border-radius
* 3、纯js实现 需要求一个点在不在圆上简单算法、获取鼠标坐标等等
* 实现不使用 border 画出1px高的线，在不同浏览器的标准模式与怪异模式下都能保持一致的效果。
* <div style="height:1px;overflow:hidden;background:#ccc"></div>
* 网页验证码是干嘛的，是为了解决什么安全问题。
* 区分用户是计算机还是人的公共全自动程序。可以防止：恶意破解密码、刷票、论坛灌水；
* 有效防止黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试；

**CSS**

* 介绍一下CSS的盒子模型？
* （1）有两种， IE 盒子模型、标准 W3C 盒子模型；IE的content部分包含了 border 和 pading;
* （2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border).
* CSS选择符有哪些？哪些属性可以继承？
* \* 1.id选择器（ # myid）
* 2.类选择器（.myclassname）
* 3.标签选择器（div, h1, p）
* 4.相邻选择器（h1 + p）
* 5.子选择器（ul > li）
* 6.后代选择器（li a）
* 7.通配符选择器（ \* ）
* 8.属性选择器（a[rel = "external"]）
* 9.伪类选择器（a: hover, li: nth - child）
* \* 可继承的样式： font-size font-family color, UL LI DL DD DT;
* \* 不可继承的样式：border padding margin width height ;
* CSS优先级算法如何计算？ \* 优先级就近原则，同权重情况下样式定义最近者为准;
* \* 载入样式以最后载入的定位为准;
* 优先级为:
* !important > id > class > tag
* important 比 内联优先级高
* CSS3新增伪类有那些？
* CSS3新增伪类举例：
* p:first-of-type 选择属于其父元素的首个 <p> 元素的每个 <p> 元素。
* p:last-of-type 选择属于其父元素的最后 <p> 元素的每个 <p> 元素。
* p:only-of-type 选择属于其父元素唯一的 <p> 元素的每个 <p> 元素。
* p:only-child 选择属于其父元素的唯一子元素的每个 <p> 元素。
* p:nth-child(2) 选择属于其父元素的第二个子元素的每个 <p> 元素。
* :enabled :disabled 控制表单控件的禁用状态。
* :checked 单选框或复选框被选中。
* 如何居中div？如何居中一个浮动元素？如何让绝对定位的div居中？
  + 给div设置一个宽度，然后添加margin:0 auto属性
  + div{
  + width:200px;
  + margin:0 auto;
  + }
  + 居中一个浮动元素
  + 确定容器的宽高 宽500 高 300 的层
  + 设置层的外边距
  + .div {
  + width:500px ; height:300px;//高度可以不设
  + margin: -150px 0 0 -250px;
  + position:relative; //相对定位
  + background-color:pink; //方便看效果
  + left:50%;
  + top:50%;
  + }
  + 让绝对定位的div居中
  + position: absolute;
  + width: 1200px;
  + background: none;
  + margin: 0 auto;
  + top: 0;
  + left: 0;
  + bottom: 0;
  + right: 0;
* display有哪些值？说明他们的作用。
* block 象块类型元素一样显示。
* none 缺省值。象行内元素类型一样显示。
* inline-block 象行内元素一样显示，但其内容象块类型元素一样显示。
* list-item 象块类型元素一样显示，并添加样式列表标记。
* position的值relative和absolute定位原点是？
* absolute
* 生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。
* fixed （老IE不支持）
* 生成绝对定位的元素，相对于浏览器窗口进行定位。
* relative
* 生成相对定位的元素，相对于其正常位置进行定位。
* static
* 默认值。没有定位，元素出现在正常的流中
* （忽略 top, bottom, left, right z-index 声明）。
* inherit
* 规定从父元素继承 position 属性的值。
* CSS3有哪些新特性？
* CSS3实现圆角（border-radius:8px），
* 阴影（box-shadow:10px），
* 文字特效（text-shadow、），
* 线性渐变（gradient），
* 旋转（transform）
* transform:rotate(9deg) scale(0.85,0.90) translate(0px,-30px) skew(-9deg,0deg);//旋转,缩放,定位,倾斜
* 增加了更多的CSS选择器
* 多背景 rgba
* 请解释一下CSS3的Flexbox（弹性盒布局模型）,以及适用场景？
* 用纯CSS创建一个三角形的原理是什么？
* 把上、左、右三条边隐藏掉（颜色设为 transparent）
* #demo {
* width: 0;
* height: 0;
* border-width: 20px;
* border-style: solid;
* border-color: transparent transparent red transparent;
* }
* 一个满屏 品 字布局 如何设计?
* 简单的方式：
* 上面的div宽100%，
* 下面的两个div分别宽50%，
* 然后用float或者inline使其不换行即可
* 常见兼容性问题？
* \* png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.
* \* 浏览器默认的margin和padding不同。解决方案是加一个全局的\*{margin:0;padding:0;}来统一。
* \* IE6双边距bug:块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。
* 浮动ie产生的双倍距离 #box{ float:left; width:10px; margin:0 0 0 100px;}
* 这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中加入 ——\_display:inline;将其转化为行内属性。(\_这个符号只有ie6会识别)
* 渐进识别的方式，从总体中逐渐排除局部。
* 首先，巧妙的使用“\9”这一标记，将IE游览器从所有情况中分离出来。
* 接着，再次使用“+”将IE8和IE7、IE6分离开来，这样IE8已经独立识别。
* css
* .bb{
* background-color:#f1ee18;/\*所有识别\*/
* .background-color:#00deff\9; /\*IE6、7、8识别\*/
* +background-color:#a200ff;/\*IE6、7识别\*/
* \_background-color:#1e0bd1;/\*IE6识别\*/
* }
* \* IE下,可以使用获取常规属性的方法来获取自定义属性,
* 也可以使用getAttribute()获取自定义属性;
* Firefox下,只能使用getAttribute()获取自定义属性。
* 解决方法:统一通过getAttribute()获取自定义属性。
* \* IE下,even对象有x,y属性,但是没有pageX,pageY属性;
* Firefox下,event对象有pageX,pageY属性,但是没有x,y属性。
* \* 解决方法：（条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。
* \* Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,
* 可通过加入 CSS 属性 -webkit-text-size-adjust: none; 解决。
* 超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了解决方法是改变CSS属性的排列顺序:
* L-V-H-A : a:link {} a:visited {} a:hover {} a:active {}
* li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法？
* 经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用hack的技巧 ？
* 为什么要初始化CSS样式。
* - 因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。
* - 当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。
* 最简单的初始化方法： \* {padding: 0; margin: 0;} （强烈不建议）
* 淘宝的样式初始化代码：
* body, h1, h2, h3, h4, h5, h6, hr, p, blockquote, dl, dt, dd, ul, ol, li, pre, form, fieldset, legend, button, input, textarea, th, td { margin:0; padding:0; }
* body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }
* h1, h2, h3, h4, h5, h6{ font-size:100%; }
* address, cite, dfn, em, var { font-style:normal; }
* code, kbd, pre, samp { font-family:couriernew, courier, monospace; }
* small{ font-size:12px; }
* ul, ol { list-style:none; }
* a { text-decoration:none; }
* a:hover { text-decoration:underline; }
* sup { vertical-align:text-top; }
* sub{ vertical-align:text-bottom; }
* legend { color:#000; }
* fieldset, img { border:0; }
* button, input, select, textarea { font-size:100%; }
* table { border-collapse:collapse; border-spacing:0; }
* absolute的containing block(容器块)计算方式跟正常流有什么不同？
* 无论属于哪种，都要先找到其祖先元素中最近的 position 值不为 static 的元素，然后再判断：
* 1、若此元素为 inline 元素，则 containing block 为能够包含这个元素生成的第一个和最后一个 inline box 的 padding box (除 margin, border 外的区域) 的最小矩形；
* 2、否则,则由这个祖先元素的 padding box 构成。
* 如果都找不到，则为 initial containing block。
* 补充：
* 1. static(默认的)/relative：简单说就是它的父元素的内容框（即去掉padding的部分）
* 2. absolute: 向上找最近的定位为absolute/relative的元素
* 3. fixed: 它的containing block一律为根元素(html/body)，根元素也是initial containing block
* CSS里的visibility属性有个collapse属性值是干嘛用的？在不同浏览器下以后什么区别？
* position跟display、margin collapse、overflow、float这些特性相互叠加后会怎么样？
* 对BFC规范(块级格式化上下文：block formatting context)的理解？
* （W3C CSS 2.1 规范中的一个概念,它是一个独立容器，决定了元素如何对其内容进行定位,以及与其他元素的关系和相互作用。）
* 一个页面是由很多个 Box 组成的,元素的类型和 display 属性,决定了这个 Box 的类型。
* 不同类型的 Box,会参与不同的 Formatting Context（决定如何渲染文档的容器）,因此Box内的元素会以不同的方式渲染,也就是说BFC内部的元素和外部的元素不会互相影响。
* css定义的权重
* 以下是权重的规则：标签的权重为1，class的权重为10，id的权重为100，以下例子是演示各种定义的权重值：
* /\*权重为1\*/
* div{
* }
* /\*权重为10\*/
* .class1{
* }
* /\*权重为100\*/
* #id1{
* }
* /\*权重为100+1=101\*/
* #id1 div{
* }
* /\*权重为10+1=11\*/
* .class1 div{
* }
* /\*权重为10+10+1=21\*/
* .class1 .class2 div{
* }
* 如果权重相同，则最后定义的样式会起作用，但是应该避免这种情况出现
* 请解释一下为什么会出现浮动和什么时候需要清除浮动？清除浮动的方式
* 移动端的布局用过媒体查询吗？
* 使用 CSS 预处理器吗？喜欢那个？
* SASS (SASS、LESS没有本质区别，只因为团队前端都是用的SASS)
* CSS优化、提高性能的方法有哪些？
* 浏览器是怎样解析CSS选择器的？
* 在网页中的应该使用奇数还是偶数的字体？为什么呢？
* margin和padding分别适合什么场景使用？
* 抽离样式模块怎么写，说出思路，有无实践经验？[阿里航旅的面试题]
* 元素竖向的百分比设定是相对于容器的高度吗？
* 全屏滚动的原理是什么？用到了CSS的那些属性？
* 什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE？
* 视差滚动效果，如何给每页做不同的动画？（回到顶部，向下滑动要再次出现，和只出现一次分别怎么做？）
* ::before 和 :after中双冒号和单冒号 有什么区别？解释一下这2个伪元素的作用。
* 如何修改chrome记住密码后自动填充表单的黄色背景 ？
* 你对line-height是如何理解的？
* 设置元素浮动后，该元素的display值是多少？（自动变成display:block）
* 怎么让Chrome支持小于12px 的文字？
* 让页面里的字体变清晰，变细用CSS怎么做？（-webkit-font-smoothing: antialiased;）
* font-style属性可以让它赋值为“oblique” oblique是什么意思？
* position:fixed;在android下无效怎么处理？
* 如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）
* 多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60＊1000ms ＝ 16.7ms
* display:inline-block 什么时候会显示间隙？(携程)
* 移除空格、使用margin负值、使用font-size:0、letter-spacing、word-spacing
* overflow: scroll时不能平滑滚动的问题怎么处理？
* 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度。
* png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp？
* 什么是Cookie 隔离？（或者说：请求资源的时候不要让它带cookie怎么做）
* 如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，
* 所以不如隔离开。
* 因为cookie有域的限制，因此不能跨域提交请求，故使用非主要域名的时候，请求头中就不会带有cookie数据，
* 这样可以降低请求头的大小，降低请求时间，从而达到降低整体请求延时的目的。
* 同时这种方式不会将cookie传入Web Server，也减少了Web Server对cookie的处理分析环节，
* 提高了webserver的http请求的解析速度。

**JavaScript**

* 介绍js的基本数据类型。
* Undefined、Null、Boolean、Number、String
* 介绍js有哪些内置对象？
* Object 是 JavaScript 中所有对象的父对象
* 数据封装类对象：Object、Array、Boolean、Number 和 String
* 其他对象：Function、Arguments、Math、Date、RegExp、Error
* 说几条写JavaScript的基本规范？
* 1.不要在同一行声明多个变量。
* 2.请使用 ===/!==来比较true/false或者数值
* 3.使用对象字面量替代new Array这种形式
* 4.不要使用全局函数。
* 5.Switch语句必须带有default分支
* 6.函数不应该有时候有返回值，有时候没有返回值。
* 7.For循环必须使用大括号
* 8.If语句必须使用大括号
* 9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。
* JavaScript原型，原型链 ? 有什么特点？
* 每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，
* 如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，
* 于是就这样一直找下去，也就是我们平时所说的原型链的概念。
* 关系：instance.constructor.prototype = instance.\_\_proto\_\_
* 特点：
* JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。当我们修改原型时，与之相关的对象也会继承这一改变。
* 当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，
* 就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。
* function Func(){}
* Func.prototype.name = "Sean";
* Func.prototype.getInfo = function() {
* return this.name;
* }
* var person = new Func();//现在可以参考var person = Object.create(oldObject);
* console.log(person.getInfo());//它拥有了Func的属性和方法
* //"Sean"
* console.log(Func.prototype);
* // Func { name="Sean", getInfo=function()}
* JavaScript有几种类型的值？，你能画一下他们的内存图吗？
* 堆：原始数据类型（Undefined，Null，Boolean，Number、String）
* 栈：引用数据类型（对象、数组和函数）
* 两种类型的区别是：存储位置不同；
* 原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；
* 引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定,如果存储在栈中，将会影响程序运行的性能；引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，会首先检索其
* 在栈中的地址，取得地址后从堆中获得实体

[](https://camo.githubusercontent.com/d1947e624a0444d1032a85800013df487adc5550/687474703a2f2f7777772e77337363686f6f6c2e636f6d2e636e2f692f63745f6a735f76616c75652e676966)

* Javascript如何实现继承？
* 1、构造继承
* 2、原型继承
* 3、实例继承
* 4、拷贝继承
* 原型prototype机制或apply和call方法去实现较简单，建议使用构造函数与原型混合方式。
* function Parent(){
* this.name = 'wang';
* }
* function Child(){
* this.age = 28;
* }
* Child.prototype = new Parent();//继承了Parent，通过原型
* var demo = new Child();
* alert(demo.age);
* alert(demo.name);//得到被继承的属性
* }
* javascript创建对象的几种方式？
* javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。
* 1、对象字面量的方式
* person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};
* 2、用function来模拟无参的构造函数
* function Person(){}
* var person=new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class
* person.name="Mark";
* person.age="25";
* person.work=function(){
* alert(person.name+" hello...");
* }
* person.work();
* 3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）
* function Pet(name,age,hobby){
* this.name=name;//this作用域：当前对象
* this.age=age;
* this.hobby=hobby;
* this.eat=function(){
* alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");
* }
* }
* var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象
* maidou.eat();//调用eat方法
* 4、用工厂方式来创建（内置对象）
* var wcDog =new Object();
* wcDog.name="旺财";
* wcDog.age=3;
* wcDog.work=function(){
* alert("我是"+wcDog.name+",汪汪汪......");
* }
* wcDog.work();
* 5、用原型方式来创建
* function Dog(){
* }
* Dog.prototype.name="旺财";
* Dog.prototype.eat=function(){
* alert(this.name+"是个吃货");
* }
* var wangcai =new Dog();
* wangcai.eat();
* 5、用混合方式来创建
* function Car(name,price){
* this.name=name;
* this.price=price;
* }
* Car.prototype.sell=function(){
* alert("我是"+this.name+"，我现在卖"+this.price+"万元");
* }
* var camry =new Car("凯美瑞",27);
* camry.sell();
* Javascript作用链域?
* 全局函数无法查看局部函数的内部细节，但局部函数可以查看其上层的函数细节，直至全局细节。
* 当需要从局部函数查找某一属性或方法时，如果当前作用域没有找到，就会上溯到上层作用域查找，
* 直至全局函数，这种组织形式就是作用域链。
* 谈谈This对象的理解。
* eval是做什么的？
* 它的功能是把对应的字符串解析成JS代码并运行；
* 应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。
* 什么是window对象? 什么是document对象?
* null，undefined 的区别？
* null 表示一个对象被定义了，值为“空值”；
* undefined 表示不存在这个值。
* typeof undefined
* //"undefined"
* undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；
* 例如变量被声明了，但没有赋值时，就等于undefined
* typeof null
* //"object"
* null : 是一个对象(空对象, 没有任何属性和方法)；
* 例如作为函数的参数，表示该函数的参数不是对象；
* 注意：
* 在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined
* 再来一个例子：
* null
* Q：有张三这个人么？
* A：有！
* Q：张三有房子么？
* A：没有！
* undefined
* Q：有张三这个人么？
* A：没有！
* 写一个通用的事件侦听器函数。
* // event(事件)工具集，来源：github.com/markyun
* markyun.Event = {
* // 页面加载完成后
* readyEvent : function(fn) {
* if (fn==null) {
* fn=document;
* }
* var oldonload = window.onload;
* if (typeof window.onload != 'function') {
* window.onload = fn;
* } else {
* window.onload = function() {
* oldonload();
* fn();
* };
* }
* },
* // 视能力分别使用dom0||dom2||IE方式 来绑定事件
* // 参数： 操作的元素,事件名称 ,事件处理程序
* addEvent : function(element, type, handler) {
* if (element.addEventListener) {
* //事件类型、需要执行的函数、是否捕捉
* element.addEventListener(type, handler, false);
* } else if (element.attachEvent) {
* element.attachEvent('on' + type, function() {
* handler.call(element);
* });
* } else {
* element['on' + type] = handler;
* }
* },
* // 移除事件
* removeEvent : function(element, type, handler) {
* if (element.removeEventListener) {
* element.removeEventListener(type, handler, false);
* } else if (element.datachEvent) {
* element.detachEvent('on' + type, handler);
* } else {
* element['on' + type] = null;
* }
* },
* // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)
* stopPropagation : function(ev) {
* if (ev.stopPropagation) {
* ev.stopPropagation();
* } else {
* ev.cancelBubble = true;
* }
* },
* // 取消事件的默认行为
* preventDefault : function(event) {
* if (event.preventDefault) {
* event.preventDefault();
* } else {
* event.returnValue = false;
* }
* },
* // 获取事件目标
* getTarget : function(event) {
* return event.target || event.srcElement;
* },
* // 获取event对象的引用，取到事件的所有信息，确保随时能使用event；
* getEvent : function(e) {
* var ev = e || window.event;
* if (!ev) {
* var c = this.getEvent.caller;
* while (c) {
* ev = c.arguments[0];
* if (ev && Event == ev.constructor) {
* break;
* }
* c = c.caller;
* }
* }
* return ev;
* }
* };
* ["1", "2", "3"].map(parseInt) 答案是多少？
* [1, NaN, NaN] 因为 parseInt 需要两个参数 (val, radix)，
* 其中 radix 表示解析时用的基数。
* map 传了 3 个 (element, index, array)，对应的 radix 不合法导致解析失败。
* 事件是？IE与火狐的事件机制有什么区别？ 如何阻止冒泡？
* 1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。
* 2. 事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；
* 3. ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）
* 什么是闭包（closure），为什么要用它？
* 闭包是指有权访问另一个函数作用域中变量的函数，创建闭包的最常见的方式就是在一个函数内创建另一个函数，通过另一个函数访问这个函数的局部变量,利用闭包可以突破作用链域，将函数内部的变量和方法传递到外部。
* 闭包的特性：
* 1.函数内再嵌套函数
* 2.内部函数可以引用外层的参数和变量
* 3.参数和变量不会被垃圾回收机制回收
* //li节点的onclick事件都能正确的弹出当前被点击的li索引
* <ul id="testUL">
* <li> index = 0</li>
* <li> index = 1</li>
* <li> index = 2</li>
* <li> index = 3</li>
* </ul>
* <script type="text/javascript">
* var nodes = document.getElementsByTagName("li");
* for(i = 0;i<nodes.length;i+= 1){
* nodes[i].onclick = function(){
* console.log(i+1);//不用闭包的话，值每次都是4
* }(i);
* }
* </script>
* 执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在
* 使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源
* 因为say667()的内部函数的执行需要依赖say667()中的变量
* 这是对闭包作用的非常直白的描述
* function say667() {
* // Local variable that ends up within closure
* var num = 666;
* var sayAlert = function() {
* alert(num);
* }
* num++;
* return sayAlert;
* }
* var sayAlert = say667();
* sayAlert()//执行结果应该弹出的667
* javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？
* use strict是一种ECMAscript 5 添加的（严格）运行模式,这种模式使得 Javascript 在更严格的条件下运行,
* 使JS编码更加规范化的模式,消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为。
* 默认支持的糟糕特性都会被禁用，比如不能用with，也不能在意外的情况下给全局变量赋值;
* 全局变量的显示声明,函数必须声明在顶层，不允许在非函数代码块内声明函数,arguments.callee也不允许使用；
* 消除代码运行的一些不安全之处，保证代码运行的安全,限制函数中的arguments修改，严格模式下的eval函数的行为和非严格模式的也不相同;
* 提高编译器效率，增加运行速度；
* 为未来新版本的Javascript标准化做铺垫。
* 如何判断一个对象是否属于某个类？
* 使用instanceof （待完善）
* if(a instanceof Person){
* alert('yes');
* }
* new操作符具体干了什么呢?
* 1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。
* 2、属性和方法被加入到 this 引用的对象中。
* 3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。
* var obj = {};
* obj.\_\_proto\_\_ = Base.prototype;
* Base.call(obj);
* 用原生JavaScript的实现过什么功能吗？
* Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？
* hasOwnProperty
* javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。
* 使用方法：
* object.hasOwnProperty(proName)
* 其中参数object是必选项。一个对象的实例。
* proName是必选项。一个属性名称的字符串值。
* 如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。
* JSON 的了解？
* JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。
* 它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小
* 如：{"age":"12", "name":"back"}
* [].forEach.call($$("\*"),function(a){a.style.outline="1px solid #"+(~~(Math.random()\*(1<<24))).toString(16)}) 能解释一下这段代码的意思吗？
* js延迟加载的方式有哪些？
* defer和async、动态创建DOM方式（用得最多）、按需异步载入js
* Ajax 是什么? 如何创建一个Ajax？
* ajax的全称：Asynchronous Javascript And XML。
* 异步传输+js+xml。
* 所谓异步，在这里简单地解释就是：向服务器发送请求的时候，我们不必等待结果，而是可以同时做其他的事情，等到有了结果它自己会根据设定进行后续操作，与此同时，页面是不会发生整页刷新的，提高了用户体验。
* (1)创建XMLHttpRequest对象,也就是创建一个异步调用对象
* (2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息
* (3)设置响应HTTP请求状态变化的函数
* (4)发送HTTP请求
* (5)获取异步调用返回的数据
* (6)使用JavaScript和DOM实现局部刷新
* 同步和异步的区别?

同步：浏览器访问服务器请求，用户看得到页面刷新，重新发请求,等请求完，页面刷新，新内容出现，用户看到新内容,j进行下一步操作。

异步：浏览器访问服务器请求，用户正常操作，浏览器后端进行请求。等请求完，页面不刷新，新内容也会出现，用户看到新内容。

（待完善）

* 如何解决跨域问题?
* jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面
* 页面编码和被请求的资源编码如果不一致如何处理？
* 模块化开发怎么做？

[立即执行函数](http://benalman.com/news/2010/11/immediately-invoked-function-expression/),不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

（待完善）

* AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

AMD 规范在这里：<https://github.com/amdjs/amdjs-api/wiki/AMD>

CMD 规范在这里：<https://github.com/seajs/seajs/issues/242>

Asynchronous Module Definition，异步模块定义，所有的模块将被异步加载，模块加载不影响后面语句运行。所有依赖某些模块的语句均放置在回调函数中。

区别：

1. 对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2. CMD 推崇依赖就近，AMD 推崇依赖前置。看代码：

// CMD

define(function(require, exports, module) {

var a = require('./a')

a.doSomething()

// 此处略去 100 行

var b = require('./b') // 依赖可以就近书写

b.doSomething()

// ...

})

// AMD 默认推荐

define(['./a', './b'], function(a, b) { // 依赖必须一开始就写好

a.doSomething()

// 此处略去 100 行

b.doSomething()

// ...

})

* requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）
* 谈一谈你对ECMAScript6的了解？
* ECMAScript6 怎么写class么，为什么会出现class这种东西?
* 异步加载JS的方式有哪些？
* (1) defer，只支持IE
* (2) async：
* (3) 创建script，插入到DOM中，加载完毕后callBack
* documen.write和 innerHTML的区别
* document.write只能重绘整个页面
* innerHTML可以重绘页面的一部分
* DOM操作——怎样添加、移除、移动、复制、创建和查找节点?
* （1）创建新节点
* createDocumentFragment() //创建一个DOM片段
* createElement() //创建一个具体的元素
* createTextNode() //创建一个文本节点
* （2）添加、移除、替换、插入
* appendChild()
* removeChild()
* replaceChild()
* insertBefore() //在已有的子节点前插入一个新的子节点
* （3）查找
* getElementsByTagName() //通过标签名称
* getElementsByName() //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)
* getElementById() //通过元素Id，唯一性
* .call() 和 .apply() 的区别？
* 例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);
* 注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。
* function add(a,b)
* {
* alert(a+b);
* }
* function sub(a,b)
* {
* alert(a-b);
* }
* add.call(sub,3,1);
* 数组和对象有哪些原生方法，列举一下？
* JS 怎么实现一个类。怎么实例化这个类
* JavaScript中的作用域与变量声明提升？
* 如何编写高性能的Javascript？
* 那些操作会造成内存泄漏？
* JQuery的源码看过吗？能不能简单概况一下它的实现原理？
* jQuery.fn的init方法返回的this指的是什么对象？为什么要返回this？
* jquery中如何将数组转化为json字符串，然后再转化回来？
* jQuery 的属性拷贝(extend)的实现原理是什么，如何实现深拷贝？
* jquery.extend 与 jquery.fn.extend的区别？
* jQuery 的队列是如何实现的？队列可以用在哪些地方？
* 谈一下Jquery中的bind(),live(),delegate(),on()的区别？
* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* 是否知道自定义事件。jQuery里的fire函数是什么意思，什么时候用？
* jQuery 是通过哪个方法和 Sizzle 选择器结合的？（jQuery.fn.find()进入Sizzle）
* 针对 jQuery性能的优化方法？
* Jquery与jQuery UI 有啥区别？
* \*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。
* \*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。
* 提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等
* JQuery的源码看过吗？能不能简单说一下它的实现原理？
* jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

* jQuery和Zepto的区别？各自的使用场景？
* 针对 jQuery 的优化方法？
* \*基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。
* \*频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。
* 比如：var str=$("a").attr("href");
* \*for (var i = size; i < arr.length; i++) {}
* for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：
* for (var i = size, length = arr.length; i < length; i++) {}
* Zepto的点透问题如何解决？
* jQueryUI如何自定义组件?
* 需求：实现一个页面操作不会整页刷新的网站，并且能在浏览器前进、后退时正确响应。给出你的技术实现方案？
* 如何判断当前脚本运行在浏览器还是node环境中？（阿里）
* 通过判断Global对象是否为window，如果不为window，当前脚本没有运行在浏览器中
* 移动端最小触控区域是多大？
* jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?
* 把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？
* 移动端的点击事件的有延迟，时间是多久，为什么会有？ 怎么解决这个延时？（click 有 300ms 延迟,为了实现safari的双击事件的设计，浏览器要知道你是不是要双击操作。）
* 知道各种JS框架(Angular, Backbone, Ember, React, Meteor, Knockout...)么? 能讲出他们各自的优点和缺点么?
* Underscore 对哪些 JS 原生对象进行了扩展以及提供了哪些好用的函数方法？
* 解释JavaScript中的作用域与变量声明提升？
* 那些操作会造成内存泄漏？
* 内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。
* 垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。
* setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。
* 闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）
* JQuery一个对象可以同时绑定多个事件，这是如何实现的？
* Node.js的适用场景？
* (如果会用node)知道route, middleware, cluster, nodemon, pm2, server-side rendering么?
* 解释一下 Backbone 的 MVC 实现方式？
* 什么是“前端路由”?什么时候适合使用“前端路由”? “前端路由”有哪些优点和缺点?
* 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?
* 如何测试前端代码么? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?
* 前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?
* 简述一下 Handlebars 的基本用法？
* 简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？
* 用js实现千位分隔符?(来源：[前端农民工](http://div.io/topic/744)，提示：正则+replace)
* function commafy(num) {
* num = num + '';
* var reg = /(-?d+)(d{3})/;
* if(reg.test(num)){
* num = num.replace(reg, '$1,$2');
* }
* return num;
* }
* 检测浏览器版本版本有哪些方式？
* 功能检测、userAgent特征检测
* 比如：navigator.userAgent
* //"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

**其他问题**

* 原来公司工作流程是怎么样的，如何与其他人协作的？如何夸部门合作的？
* 你遇到过比较难的技术问题是？你是如何解决的？
* 设计模式 知道什么是singleton, factory, strategy, decrator么?
* 常使用的库有哪些？常用的前端开发工具？开发过什么应用或组件？
* 页面重构怎么操作？
* 网站重构：在不改变外部行为的前提下，简化结构、添加可读性，而在网站前端保持一致的行为。
* 也就是说是在不改变UI的情况下，对网站进行优化，在扩展的同时保持一致的UI。
* 对于传统的网站来说重构通常是：
* 表格(table)布局改为DIV+CSS
* 使网站前端兼容于现代浏览器(针对于不合规范的CSS、如对IE6有效的)
* 对于移动平台的优化
* 针对于SEO进行优化
* 深层次的网站重构应该考虑的方面
* 减少代码间的耦合
* 让代码保持弹性
* 严格按规范编写代码
* 设计可扩展的API
* 代替旧有的框架、语言(如VB)
* 增强用户体验
* 通常来说对于速度的优化也包含在重构中
* 压缩JS、CSS、image等前端资源(通常是由服务器来解决)
* 程序的性能优化(如数据读写)
* 采用CDN来加速资源加载
* 对于JS DOM的优化
* HTTP服务器的文件缓存
* 列举IE与其他浏览器不一样的特性？
* 1、事件不同之处：
* 触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；
* 获取字符代码、如果按键代表一个字符（shift、ctrl、alt除外），IE 的 keyCode 会返回字符代码（Unicode），DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；
* 阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，需要调用 preventDefault() 方法；
* 停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，需要调用 stopPropagation()；
* 99%的网站都需要被重构是那本书上写的？
* 网站重构：应用web标准进行设计（第2版）
* 什么叫优雅降级和渐进增强？
* 优雅降级：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，则代码会针对旧版本的IE进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。
* 如：border-shadow
* 渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新版本浏览器才支持的功能,向页面增加不影响基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。
* 如：默认使用flash上传，但如果浏览器支持 HTML5 的文件上传功能，则使用HTML5实现更好的体验；
* 是否了解公钥加密和私钥加密。
* 一般情况下是指私钥用于对数据进行签名，公钥用于对签名进行验证;
* HTTP网站在浏览器端用公钥加密敏感数据，然后在服务器端再用私钥解密。
* WEB应用从服务器主动推送Data到客户端有那些方式？
* html5提供的Websocket
* 不可见的iframe
* WebSocket通过Flash
* XHR长时间连接
* XHR Multipart Streaming
* <script>标签的长时间连接(可跨域)
* 对Node的优点和缺点提出了自己的看法？
* \*（优点）因为Node是基于事件驱动和无阻塞的，所以非常适合处理并发请求，
* 因此构建在Node上的代理服务器相比其他技术实现（如Ruby）的服务器表现要好得多。
* 此外，与Node代理服务器交互的客户端代码是由javascript语言编写的，
* 因此客户端和服务器端都用同一种语言编写，这是非常美妙的事情。
* \*（缺点）Node是一个相对新的开源项目，所以不太稳定，它总是一直在变，
* 而且缺少足够多的第三方库支持。看起来，就像是Ruby/Rails当年的样子。
* 你有用过哪些前端性能优化的方法？
* （1） 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip，CDN托管，data缓存 ，图片服务器。
* （2） 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数
* （3） 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。
* （4） 当需要设置的样式很多时设置className而不是直接操作style。
* （5） 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。
* （6） 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。
* （7） 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。
* （8） 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。
* 对普通的网站有一个统一的思路，就是尽量向前端优化、减少数据库操作、减少磁盘IO。向前端优化指的是，在不影响功能和体验的情况下，能在浏览器执行的不要在服务端执行，能在缓存服务器上直接返回的不要到应用服务器，程序能直接取得的结果不要到外部取得，本机内能取得的数据不要到远程取，内存能取到的不要到磁盘取，缓存中有的不要去数据库查询。减少数据库操作指减少更新次数、缓存结果减少查询次数、将数据库执行的操作尽可能的让你的程序完成（例如join查询），减少磁盘IO指尽量不使用文件系统作为缓存、减少读写文件次数等。程序优化永远要优化慢的部分，换语言是无法“优化”的。
* http状态码有那些？分别代表是什么意思？
* 简单版
* [
* 100 Continue 继续，一般在发送post请求时，已发送了http header之后服务端将返回此信息，表示确认，之后发送具体参数信息
* 200 OK 正常返回信息
* 201 Created 请求成功并且服务器创建了新的资源
* 202 Accepted 服务器已接受请求，但尚未处理
* 301 Moved Permanently 请求的网页已永久移动到新位置。
* 302 Found 临时性重定向。
* 303 See Other 临时性重定向，且总是使用 GET 请求新的 URI。
* 304 Not Modified 自从上次请求后，请求的网页未修改过。
* 400 Bad Request 服务器无法理解请求的格式，客户端不应当尝试再次使用相同的内容发起请求。
* 401 Unauthorized 请求未授权。
* 403 Forbidden 禁止访问。
* 404 Not Found 找不到如何与 URI 相匹配的资源。
* 500 Internal Server Error 最常见的服务器端错误。
* 503 Service Unavailable 服务器端暂时无法处理请求（可能是过载或维护）。
* ]
* 完整版
* 1\*\*(信息类)：表示接收到请求并且继续处理
* 100——客户必须继续发出请求
* 101——客户要求服务器根据请求转换HTTP协议版本
* 2\*\*(响应成功)：表示动作被成功接收、理解和接受
* 200——表明该请求被成功地完成，所请求的资源发送回客户端
* 201——提示知道新文件的URL
* 202——接受和处理、但处理未完成
* 203——返回信息不确定或不完整
* 204——请求收到，但返回信息为空
* 205——服务器完成了请求，用户代理必须复位当前已经浏览过的文件
* 206——服务器已经完成了部分用户的GET请求
* 3\*\*(重定向类)：为了完成指定的动作，必须接受进一步处理
* 300——请求的资源可在多处得到
* 301——本网页被永久性转移到另一个URL
* 302——请求的网页被转移到一个新的地址，但客户访问仍继续通过原始URL地址，重定向，新的URL会在response中的Location中返回，浏览器将会使用新的URL发出新的Request。
* 303——建议客户访问其他URL或访问方式
* 304——自从上次请求后，请求的网页未修改过，服务器返回此响应时，不会返回网页内容，代表上次的文档已经被缓存了，还可以继续使用
* 305——请求的资源必须从服务器指定的地址得到
* 306——前一版本HTTP中使用的代码，现行版本中不再使用
* 307——申明请求的资源临时性删除
* 4\*\*(客户端错误类)：请求包含错误语法或不能正确执行
* 400——客户端请求有语法错误，不能被服务器所理解
* 401——请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用
* HTTP 401.1 - 未授权：登录失败
* HTTP 401.2 - 未授权：服务器配置问题导致登录失败
* HTTP 401.3 - ACL 禁止访问资源
* HTTP 401.4 - 未授权：授权被筛选器拒绝
* HTTP 401.5 - 未授权：ISAPI 或 CGI 授权失败
* 402——保留有效ChargeTo头响应
* 403——禁止访问，服务器收到请求，但是拒绝提供服务
* HTTP 403.1 禁止访问：禁止可执行访问
* HTTP 403.2 - 禁止访问：禁止读访问
* HTTP 403.3 - 禁止访问：禁止写访问
* HTTP 403.4 - 禁止访问：要求 SSL
* HTTP 403.5 - 禁止访问：要求 SSL 128
* HTTP 403.6 - 禁止访问：IP 地址被拒绝
* HTTP 403.7 - 禁止访问：要求客户证书
* HTTP 403.8 - 禁止访问：禁止站点访问
* HTTP 403.9 - 禁止访问：连接的用户过多
* HTTP 403.10 - 禁止访问：配置无效
* HTTP 403.11 - 禁止访问：密码更改
* HTTP 403.12 - 禁止访问：映射器拒绝访问
* HTTP 403.13 - 禁止访问：客户证书已被吊销
* HTTP 403.15 - 禁止访问：客户访问许可过多
* HTTP 403.16 - 禁止访问：客户证书不可信或者无效
* HTTP 403.17 - 禁止访问：客户证书已经到期或者尚未生效
* 404——一个404错误表明可连接服务器，但服务器无法取得所请求的网页，请求资源不存在。eg：输入了错误的URL
* 405——用户在Request-Line字段定义的方法不允许
* 406——根据用户发送的Accept拖，请求资源不可访问
* 407——类似401，用户必须首先在代理服务器上得到授权
* 408——客户端没有在用户指定的饿时间内完成请求
* 409——对当前资源状态，请求不能完成
* 410——服务器上不再有此资源且无进一步的参考地址
* 411——服务器拒绝用户定义的Content-Length属性请求
* 412——一个或多个请求头字段在当前请求中错误
* 413——请求的资源大于服务器允许的大小
* 414——请求的资源URL长于服务器允许的长度
* 415——请求资源不支持请求项目格式
* 416——请求中包含Range请求头字段，在当前请求资源范围内没有range指示值，请求也不包含If-Range请求头字段
* 417——服务器不满足请求Expect头字段指定的期望值，如果是代理服务器，可能是下一级服务器不能满足请求长。
* 5\*\*(服务端错误类)：服务器不能正确执行一个正确的请求
* HTTP 500 - 服务器遇到错误，无法完成请求
* HTTP 500.100 - 内部服务器错误 - ASP 错误
* HTTP 500-11 服务器关闭
* HTTP 500-12 应用程序重新启动
* HTTP 500-13 - 服务器太忙
* HTTP 500-14 - 应用程序无效
* HTTP 500-15 - 不允许请求 global.asa
* Error 501 - 未实现
* HTTP 502 - 网关错误
* HTTP 503：由于超载或停机维护，服务器目前无法使用，一段时间后可能恢复正常
* 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）
* 注：这题胜在区分度高，知识点覆盖广，再不懂的人，也能答出几句，
* 而高手可以根据自己擅长的领域自由发挥，从URL规范、HTTP协议、DNS、CDN、数据库查询、
* 到浏览器流式解析、CSS规则构建、layout、paint、onload/domready、JS执行、JS API绑定等等；
* 详细版：
* 1、浏览器会开启一个线程来处理这个请求，对 URL 分析判断如果是 http 协议就按照 Web 方式来处理;
* 2、调用浏览器内核中的对应方法，比如 WebView 中的 loadUrl 方法;
* 3、通过DNS解析获取网址的IP地址，设置 UA 等信息发出第二个GET请求;
* 4、进行HTTP协议会话，客户端发送报头(请求报头);
* 5、进入到web服务器上的 Web Server，如 Apache、Tomcat、Node.JS 等服务器;
* 6、进入部署好的后端应用，如 PHP、Java、JavaScript、Python 等，找到对应的请求处理;
* 7、处理结束回馈报头，此处如果浏览器访问过，缓存上有对应资源，会与服务器最后修改时间对比，一致则返回304;
* 8、浏览器开始下载html文档(响应报头，状态码200)，同时使用缓存;
* 9、文档树建立，根据标记请求所需指定MIME类型的文件（比如css、js）,同时设置了cookie;
* 10、页面开始渲染DOM，JS根据DOM API操作DOM,执行事件绑定等，页面显示完成。
* 简洁版：
* 浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；
* 服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；
* 浏览器对加载到的资源（HTML、JS、CSS等）进行语法解析，建立相应的内部数据结构（如HTML的DOM）；
* 载入解析到的资源文件，渲染页面，完成。
* 部分地区用户反应网站很卡，请问有哪些可能性的原因，以及解决方法？
* 从打开app到刷新出内容，整个过程中都发生了什么，如果感觉慢，怎么定位问题，怎么解决?
* 除了前端以外还了解什么其它技术么？你最最厉害的技能是什么？
* 你用的得心应手用的熟练地编辑器&开发环境是什么样子？
* Sublime Text 3 + 相关插件编写前端代码
* Google chrome 、Mozilla Firefox浏览器 +firebug 兼容测试和预览页面UI、动画效果和交互功能
* Node.js+Gulp
* git 用于版本控制和Code Review
* 对前端工程师这个职位是怎么样理解的？它的前景会怎么样？
* 前端是最贴近用户的程序员，比后端、数据库、产品经理、运营、安全都近。
* 1、实现界面交互
* 2、提升用户体验
* 3、有了Node.js，前端可以实现服务端的一些事情
* 前端是最贴近用户的程序员，前端的能力就是能让产品从 90分进化到 100 分，甚至更好，
* 参与项目，快速高质量完成实现效果图，精确到1px；
* 与团队成员，UI设计，产品经理的沟通；
* 做好的页面结构，页面重构和用户体验；
* 处理hack，兼容、写出优美的代码格式；
* 针对服务器的优化、拥抱最新前端技术。
* 你怎么看待Web App 、hybrid App、Native App？
* 你移动端前端开发的理解？（和 Web 前端开发的主要区别是什么？）
* 你对加班的看法？
* 加班就像借钱，原则应当是------救急不救穷
* 平时如何管理你的项目？
* 先期团队必须确定好全局样式（globe.css），编码模式(utf-8) 等；
* 编写习惯必须一致（例如都是采用继承式的写法，单样式都写成一行）；
* 标注样式编写人，各模块都及时标注（标注关键样式调用的地方）；
* 页面进行标注（例如 页面 模块 开始和结束）；
* CSS跟HTML 分文件夹并行存放，命名都得统一（例如style.css）；
* JS 分文件夹存放 命名以该JS功能为准的英文翻译。
* 图片采用整合的 images.png png8 格式文件使用 尽量整合在一起使用方便将来的管理
* 如何设计突发大规模并发架构？
* 当团队人手不足，把功能代码写完已经需要加班的情况下，你会做前端代码的测试吗？
* 说说最近最流行的一些东西吧？常去哪些网站？
* ES6\WebAssembly\Node\MVVM\Web Components\React\React Native\Webpack 组件化
* 知道什么是SEO并且怎么优化么? 知道各种meta data的含义么?
* 移动端（Android IOS）怎么做好用户体验?
* 清晰的视觉纵线、
* 信息的分组、极致的减法、
* 利用选择代替输入、
* 标签及文字的排布方式、
* 依靠明文确认密码、
* 合理的键盘利用、
* 简单描述一下你做过的移动APP项目研发流程？
* 你在现在的团队处于什么样的角色，起到了什么明显的作用？
* 你认为怎样才是全端工程师（Full Stack developer）？
* 介绍一个你最得意的作品吧？
* 你有自己的技术博客吗，用了哪些技术？
* 对前端安全有什么看法？
* 是否了解Web注入攻击，说下原理，最常见的两种攻击（XSS 和 CSRF）了解到什么程度？
* 项目中遇到国哪些印象深刻的技术难题，具体是什么问题，怎么解决？。
* 最近在学什么东西？
* 你的优点是什么？缺点是什么？
* 如何管理前端团队?
* 最近在学什么？能谈谈你未来3，5年给自己的规划吗？

**前端学习网站推荐**

1. 极客标签： http://www.gbtags.com/

2. 码农周刊： http://weekly.manong.io/issues/

3. 前端周刊： http://www.feweekly.com/issues

4. 慕课网： http://www.imooc.com/

5. div.io： http://div.io

6. Hacker News： https://news.ycombinator.com/news

7. InfoQ： http://www.infoq.com/

8. w3cplus： http://www.w3cplus.com/

9. Stack Overflow： http://stackoverflow.com/

10.w3school： http://www.w3school.com.cn/

11.mozilla： https://developer.mozilla.org/zh-CN/docs/Web/JavaScript

**文档推荐**

1. [jQuery 基本原理](http://docs.huihoo.com/jquery/jquery-fundamentals/zh-cn/index.html)
2. [JavaScript 秘密花园](http://bonsaiden.github.io/JavaScript-Garden/zh/)
3. [CSS参考手册](http://css.doyoe.com/)
4. [JavaScript 标准参考教程](http://javascript.ruanyifeng.com/)
5. [ECMAScript 6入门](http://es6.ruanyifeng.com/)

# Front-End-Develop-Guide 前端开发指南

创建此项目灵感从[SwiftGuide](https://github.com/ipader/SwiftGuide)而来，这份指南汇集了前端开发所使用语言的主流学习资源，并以开发者的视角进行整理编排而成。

GitHub：[icepy/Front-End-Develop-Guide](https://github.com/icepy/Front-End-Develop-Guide) ｜ 网站：[http://front-end-developers-hunan.github.io](http://front-end-developers-hunan.github.io/) ｜ 欢迎开发者一起[*维护*](https://github.com/icepy/Front-End-Develop-Guide/pulls)，或[*反馈/投稿*](https://github.com/icepy/Front-End-Develop-Guide/issues/new)

想了解关于该指南及 前端开发 更多信息的同学，可以阅读短文《[致 Front-End Developers](https://github.com/icepy/Front-End-Develop-Guide/blob/master/2015letter.md)》。 想查看优秀的开源项目，可以访问[《Front-End Develop 项目精选》](https://github.com/icepy/Front-End-Develop-Guide/blob/master/Featured.md)。希望查看精选文章，可以访问[《Front-End Develop 文章精选》](https://github.com/icepy/Front-End-Develop-Guide/blob/master/Featured-Articles.md)。也欢迎企业招聘人才提供工作岗位(請附上薪資)，可以提交到[工作岗位](https://github.com/icepy/Front-End-Develop-Guide/issues/3)。

由[@icepy](http://weibo.com/2455876310)负责整理

**由于个人精力有限，不足之处还望大家多多包容与提供建议。**

## 目录

* [Front-End Develop 文档](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_doc)
  + [Welcome to Front-End Develop](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_we)
  + [Front-End Develop Language](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_language)
    - [HTML&HTML5 文档](https://github.com/zhanglongde/Front-End-Develop-Guide#html_doc)
    - [CSS&CSS3 文档](https://github.com/zhanglongde/Front-End-Develop-Guide#css_doc)
      * [CSS Style Guide](https://github.com/zhanglongde/Front-End-Develop-Guide#css_style_guide)
      * [CSS Language](https://github.com/zhanglongde/Front-End-Develop-Guide#css_language)
      * [Sass Guide](https://github.com/zhanglongde/Front-End-Develop-Guide#sass_guide)
      * [PostCss](https://github.com/zhanglongde/Front-End-Develop-Guide#postcss_guide)
    - [JavaScript 文档](https://github.com/zhanglongde/Front-End-Develop-Guide#javascript_doc)
      * [JavaScript Style Guide](https://github.com/zhanglongde/Front-End-Develop-Guide#javascript_style_guide)
      * [JavaScript Language](https://github.com/zhanglongde/Front-End-Develop-Guide#javascript_language)
  + [Front-End Develop Fly](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_fly)
    - [常用库与框架的使用文档](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_library)
    - [JavaScript 日志](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_log)
    - [开放网络标准](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_openAPI)
    - [HTTP 文档](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_http)
    - [缓存策略](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_cache)
    - [前端自动化工具](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_automate_kit)
    - [React&React Native生态-探索学习](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_point_recommend)
    - [移动前端技术](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_mobile_jishu)
    - [动画](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_animation)
    - [性能调优](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_xingneng_tiaoyou)
  + [Front-End Develop 项目与工具](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_project_kit)
    - [相关工具](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_kit)
    - [Front-End Develop 开源项目](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_opensource)
  + [Front-End Questions](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_questions)
  + [Front-End 需要了解的数据结构与算法](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_datastructure)
  + [Front-End Develop 优秀文章](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_article)
* [延伸可扩展的方向](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_scalable)
  + [Node.js](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_nodejs)
  + [Mongodb](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_mongodb)
  + [WebGL](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_webGL)
  + [Chrome扩展开发](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_chrome)
  + [桌面应用程序开发](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_PCAPP)
  + [移动应用程序开发](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_mobile)
  + [V8 引擎文献](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_v8)
* [Front-End Developer Blog](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_blog)
* [通用Developer 社区](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_community)
* [技术分享会议](https://github.com/zhanglongde/Front-End-Develop-Guide#fed_jishu_share)

## Front-End Develop 文档

初略读懂Front-End Develop要做些什么

[Web Front-End Stack](https://github.com/unruledboy/WebFrontEndStack/blob/master/README.md)：这张图非常详细。

## Welcom to Front-End Develop

W3C组织提供了官网的地址：[w3.org](http://www.w3.org/)，可以去查询最近实现的标准以及将来可能实现的讨论。

## Front-End Develop Language

web前端开发可能包括HTML,CSS,JavaScript,Web API,SVG,WebGL,MathML等，语言是编程的基础，火狐的MDN出了一份比较通略的文档，可以快速查看[Web 技术文档](https://developer.mozilla.org/zh-CN/docs/Web)。

### HTML&HTML5 文档

* [无处不在的html](https://www.phodal.com/blog/be-a-geek-chapter-1-anywhere-html/)：HTML是Web的核心语言，也是最基础的语言；
* [XHTML2 Working Group Home Page](http://www.w3.org/MarkUp/)
* [HTML5 API Search](http://html5index.org/)：可以快速的查询某个HTML5的API；
* [HTML5 Rocks](http://www.html5rocks.com/zh/tutorials/?page=1)：通过这个网站可以搜索到很多关于HTML5的教程；

书籍：

* [图灵程序设计丛书:HTML5权威指南](http://www.amazon.cn/%E5%9B%BE%E7%81%B5%E7%A8%8B%E5%BA%8F%E8%AE%BE%E8%AE%A1%E2%80%A69B%BC/dp/B00H706BIG/ref=sr_1_4?ie=UTF8&qid=1424220765&sr=8-4&keywords=html)

### CSS&CSS3 文档

#### CSS Style Guide

* [编码规范 by @mdo](http://zoomzhao.github.io/code-guide/)：开发灵活，稳定，可持续 HTML 和 CSS 代码的规范，每一项都有很详细的解释；
* [css guide lines](http://cssguidelin.es/)
* [通用 CSS 笔记、建议与指导](https://github.com/chadluo/CSS-Guidelines/blob/master/README.md)：本文档第一部分将探讨语法、格式以及分析 CSS 结构；第二部分将围绕方法论、思维框架以及编写与规划 CSS 的看法；

#### CSS Language

* [学习CSS布局](http://zh.learnlayout.com/)：教授的是现在广泛使用于网站布局领域的CSS基础；
* [CSS3 Tutorial 《CSS3 教程》](https://github.com/waylau/css3-tutorial)：CSS3 Tutorial 是一本关于 CSS3 的开源书，作者利用业余时间写了本书，图文并茂，用大量实例带你一步一步走进 CSS3 的世界；
* [Magic of CSS](http://adamschwartz.co/magic-of-css/)
* [CSS词汇表](http://yisibl.github.io/css-vocabulary/)：查询CSS的词汇；
* [CSS参考](http://css.doyoe.com/)

#### Sass Guide

* [Sass基础知识](http://sass-lang.com/guide)：Sass官网，讲述Sass/SCSS应用的基础知识，变量、函数、条件、循环、宏等等；
* [Sass中文网](http://www.sasschina.com/)：中文大，给E文不好的小朋友们；
* [Sass Guidelines](http://sass-guidelin.es/zh/)：这是由Sass专家[Hugo Giraudel](http://hugogiraudel.com/)撰写的一份“编写稳健、可维护和可扩展的Sass”的指南，这份指南已被翻译成中文；
* [Sass meister](http://www.sassmeister.com/)：在线Sass/SCSS调试器；

#### PostCss

* [Post Css](https://github.com/postcss/postcss)

### JavaScript 文档

注明：JavaScript文档将引用[justjavac/free-programming-books-zh\_CN#JavaScript](https://github.com/justjavac/free-programming-books-zh_CN#javascript)所整理的免费图书以及自己所收集的资料；

#### JavaScript Style Guide

* [Google JavaScript 代码风格指南](http://bq69.com/blog/articles/script/868/google-javascript-style-guide.html)：Google风格指南不但指出每条规范，还解释了为什么这样写的原因，同时给出了对与错的实例；
* [Airbnb JavaScript 规范](https://github.com/adamlu/javascript-style-guide)：A mostly reasonable approach to JavaScript，跟Google规范类似；
* [Google JSON 风格指南](https://github.com/darcyliu/google-styleguide/blob/master/JSONStyleGuide.md)：该风格指南是对在Google创建JSON APIs而提供的指导性准则和建议。总体来讲，JSON APIs应遵循JSON.org上的规范。这份风格指南澄清和标准化了特定情况，从而使Google的JSON APIs有一种标准的外观和感觉。这些指南适用于基于RPC和基于REST风格的API的JSON请求和响应；
* [Javascript编程指南](http://pij.robinqu.me/) ([源码](https://github.com/RobinQu/Programing-In-Javascript))

#### JavaScript Language

* [重新介绍 JavaScript（JS 教程）](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/A_re-introduction_to_JavaScript)：是火狐推出的一系列的JS教程，为什么会有这一篇“重新介绍”呢？因为 JavaScript 堪称世界上被人误解最深的编程语言。虽然常被视作“玩具语言”，但它看似简洁外衣下，还隐藏着强大的语言特性。 JavaScript 目前广泛应用于一大批知名应用中，对于网页和移动开发者来说，深入理解 JavaScript 就尤有必要；
* [JavaScript 标准参考教程（alpha）](http://javascript.ruanyifeng.com/)
* [javascript 的 12 个怪癖](https://github.com/justjavac/12-javascript-quirks)
* [JavaScript 秘密花园](http://bonsaiden.github.io/JavaScript-Garden/zh/)
* [JavaScript核心概念及实践](http://icodeit.org/jsccp/)：(PDF) (此书已由人民邮电出版社出版发行，但作者依然免费提供PDF版本，希望开发者们去购买，支持作者)；
* [《JavaScript 模式》](https://github.com/jayli/javascript-patterns)：“JavaScript patterns”中译本；
* [命名函数表达式探秘](http://justjavac.com/named-function-expressions-demystified.html) ：(注:原文由[为之漫笔](http://www.cn-cuckoo.com/)翻译，原始地址无法打开，所以此处地址为我博客上的备份)；
* [学用 JavaScript 设计模式](http://www.oschina.net/translate/learning-javascript-design-patterns) ：(开源中国)
* [深入理解JavaScript系列](http://www.cnblogs.com/TomXu/archive/2011/12/15/2288411.html)：汤姆大叔2011年翻译的一系列的文章，很值得一读；
* [ECMAScript 6 入门](http://es6.ruanyifeng.com/)：(作者：阮一峰)
* [JavaScript Promise迷你书](http://liubin.github.io/promises-book/)：讲述Promise实现；
* [You-Dont-Know-JS](https://github.com/getify/You-Dont-Know-JS)：(深入JavaScript语言核心机制的系列图书)；

## Front-End Develop Fly

如果你想飞起来，这正是好去处。

### 了解常用库与框架的使用文档

* jQuery
  + [jQuery 解构](http://www.cn-cuckoo.com/deconstructed/jquery.html)
  + [简单易懂的JQuery魔法](http://www.nowamagic.net/librarys/books/contents/jquery)
  + [How to write jQuery plugin](http://i5ting.github.io/How-to-write-jQuery-plugin/build/jquery.plugin.html)
* underscore.js
  + [Underscore.js中文文档](http://learningcn.com/underscore/)
* backbone.js
  + [backbone.js入门教程](http://www.the5fire.com/backbone-js-tutorials-pdf-download.html) (PDF)
  + [Backbone.js入门教程第二版](https://github.com/the5fire/backbonejs-learning-note)
  + [Developing Backbone.js Applications(中文版)](http://feliving.github.io/developing-backbone-applications)
* AngularJS
  + [AngularJS最佳实践和风格指南](https://github.com/mgechev/angularjs-style-guide/blob/master/README-zh-cn.md)
  + [AngularJS中译本](https://github.com/peiransun/angularjs-cn)
  + [AngularJS入门教程](https://github.com/zensh/AngularjsTutorial_cn)
  + [构建自己的AngularJS](https://github.com/xufei/Make-Your-Own-AngularJS/blob/master/01.md)
  + [在Windows环境下用Yeoman构建AngularJS项目](http://www.waylau.com/build-angularjs-app-with-yeoman-in-windows/)
* Zepto.js
  + [Zepto.js 中文文档](http://mweb.baidu.com/zeptoapi/)
* Sea.js
  + [Hello Sea.js](http://island205.github.io/HelloSea.js/)
* impress.js
  + [impress.js的中文教程](https://github.com/kokdemo/impress.js-tutorial-in-Chinese)
* CoffeeScript
  + [CoffeeScript Cookbook](http://island205.github.io/coffeescript-cookbook.github.com/)
  + [The Little Book on CoffeeScript中文版](http://island205.github.io/tlboc/)
  + [CoffeeScript 编码风格指南](https://github.com/geekplux/coffeescript-style-guide)
* ExtJS
  + [Ext4.1.0 中文文档](http://extjs-doc-cn.github.io/ext4api/)
* Meteor
  + [Discover Meteor](http://zh.discovermeteor.com/)
* TypeScript
  + [TypeScript 指南](https://github.com/vilic/typescript-guide)
* requirejs
  + [RequireJS 中文网](http://requirejs.cn/)
* vue.js
  + [vue 官方网站](http://cn.vuejs.org/)

### JavaScript 日志

[前端代码异常日志与监控](http://www.cnblogs.com/hustskyking/p/fe-monitor.html)

### 开放网络标准

[理解OAuth 2.0](http://www.ruanyifeng.com/blog/2014/05/oauth_2_0.html)

### HTTP 文档

下面两篇文章不会对 HTTP 的细节进行深究，而是从够高和更结构化的角度将 HTTP 协议的元素进行分类讲解，可以先阅读[HTTP 协议漫谈](http://blog.jobbole.com/88199/)和[图解Http协议](http://www.bysocket.com/?p=282)来对HTTP有一个大概的了解。

* [HTTP协议](http://www.cnblogs.com/TankXiao/category/415412.html)
* [HTTP/2协议的背景、内容、实现和未来](https://ye11ow.gitbooks.io/http2-explained/content/)
* [HTTP 下午茶](http://happypeter.github.io/tealeaf-http/#chinese)
* [HTTP接口设计指南](https://github.com/bolasblack/http-api-guide)
* [白话 HTTPS & SSL/TSL](http://www.jianshu.com/p/992bad24412e)：你肯定能阅读明白。

深入全面了解，可阅读[HTTP权威指南](http://book.douban.com/subject/10746113/)。

### 缓存策略

cache是提高应用性能重要的一个环节，必需要有所了解。

[Web缓存机制系列](http://www.alloyteam.com/2012/03/web-cache-1-web-cache-overview/)

### 前端自动化工具

现在的前端自动化工具主要如下三个流派，目测涵盖90%以上经常使用。

* [Grunt JavaScript世界的构建工具](http://www.gruntjs.net/)
* [Gulp 用自动化构建工具增强你的工作流](http://www.gulpjs.com.cn/)
* [Webpack](https://github.com/webpack/webpack)

### React&React Native生态-探索学习

* [React&React Native生态-探索学习](https://github.com/icepy/Front-End-Develop-Guide/blob/master/React.md)

### 移动前端技术

* [移动端尺寸基础知识](http://colachan.com/post/3435)
* [移动端高清、多屏适配方案](http://www.html-js.com/article/Mobile-terminal-H5-mobile-terminal-HD-multi-screen-adaptation-scheme%203041)
* [移动端开发小记 - Flexbox](http://taobaofed.org/blog/2015/11/11/flexbox-in-mobile-web/)

### 动画

* [搞定这些疑难杂症，向css3动画说yes](http://mp.weixin.qq.com/s?__biz=MjM5MTA1MjAxMQ==&mid=400429327&idx=1&sn=e535eac5d3c74e2fb1633d3e8cc0e0ed&scene=4#wechat_redirect)

### 性能调优

* [web移动端性能调优及16ms优化](http://www.ghugo.com/gone-in-60-frames-per-second/)
* [无线性能优化：域名收敛](http://taobaofed.org/blog/2015/12/16/h5-performance-optimization-and-domain-convergence/)
* [缓存机制浅析 移动端 Web 加载性能优化](http://mp.weixin.qq.com/s?__biz=MzA3NTYzODYzMg==&mid=402077566&idx=1&sn=def3337205c3aec5e0fde2476ee03397&scene=0&key=ac89cba618d2d976159e30761eefe9953dc2030a7d72c1872c445a8caaa0f1d3cc4eb416a1c7cfb82651db48d11f3f90&ascene=0&uin=MjAyNzY1NTU%3D)
* [移动端网络优化－同样适用于HTML5页面](http://www.trinea.cn/android/mobile-performance-optimization/)
* [移动H5前端性能优化指南](http://isux.tencent.com/h5-performance.html)

## Front-End Develop 项目与工具

工欲善其事，必先利其器。开源项目与工具的使用，将极大的提高前端开发之效率。

### 相关工具

#### 1. 开发工具

* [Sublime Text](http://www.sublimetext.com/)：前端开发好用到爆；
* [Atom](https://atom.io/)：Github出品的一个类似Sublime Text编辑器，长的蛮像的，快捷键也非常类似；
* [Vundle.vim](https://github.com/VundleVim/Vundle.vim)：使用vim开发前端的利剑，包括其他语言；

Sublime Text常用的插件：

* [Emmet](https://github.com/sergeche/emmet-sublime)：zen coding的升级版，对于前端来说，可是必备插件;
* [sublimelint](https://github.com/lunixbochs/sublimelint)：用于代码的校验，支持 HTML、CSS、JS、PHP、Java、C++ 等16种语言;
* [JsFormat](https://packagecontrol.io/packages/JsFormat)：用于JavaScript格式化;
* [jQuery](https://packagecontrol.io/packages/jQuery)：jQuery语法提示;
* [Underscore](https://packagecontrol.io/packages/Underscore.js%20Snippets)：Underscore语法提示;
* [DocBlockr](https://packagecontrol.io/packages/DocBlockr)：快速编写注释;
* [SublimeCodeIntel](https://packagecontrol.io/packages/SublimeCodeIntel)：智能语法分析提示;
* [ColorPicker](https://packagecontrol.io/packages/ColorPicker)：快速选取颜色
* [HTML-CSS-JS Prettify](https://packagecontrol.io/packages/HTML-CSS-JS%20Prettify)：html/css/js 格式化
* [Alignment](https://packagecontrol.io/packages/Alignment)：用于代码对齐;

Atom常用的插件：

* [TypeScript](https://github.com/TypeStrong/atom-typescript)：支持TypeScript编译；
* [run-in-browser](https://atom.io/packages/run-in-browser)：支持运行网页；
* [Emmet](https://atom.io/packages/emmet)：快速编写HTML；
* [minimap](https://atom.io/packages/minimap)：右侧小地图；
* [color-picker](https://atom.io/packages/color-picker)：CSS颜色取色器；
* [autoprefixer](https://atom.io/packages/autoprefixer)：自动补全CSS前缀；
* [autocomplete-paths](https://atom.io/packages/autocomplete-paths)：对路径选择进行增强；
* [autocomplete-python](https://atom.io/packages/autocomplete-python)：对Python提示进行增强；
* [docblockr](https://atom.io/packages/docblockr)：编写注释好帮手；
* [file-icons](https://atom.io/packages/file-icons)：美化编辑器图标；
* [linter-jshint](https://atom.io/packages/linter-jshint)：JavaScript语法检查；

vim常用的插件：

Bundle 'christoomey/vim-run-interactive'

Bundle 'Valloric/YouCompleteMe'

Bundle 'croaky/vim-colors-github'

Bundle 'danro/rename.vim'

Bundle 'majutsushi/tagbar'

Bundle 'kien/ctrlp.vim'

Bundle 'pbrisbin/vim-mkdir'

Bundle 'scrooloose/syntastic'

Bundle 'slim-template/vim-slim'

Bundle 'thoughtbot/vim-rspec'

Bundle 'tpope/vim-bundler'

Bundle 'tpope/vim-endwise'

Bundle 'tpope/vim-fugitive'

Bundle 'tpope/vim-rails'

Bundle 'tpope/vim-surround'

Bundle 'vim-scripts/ctags.vim'

Bundle 'vim-scripts/tComment'

Bundle "mattn/emmet-vim"

Bundle "scrooloose/nerdtree"

Bundle "Lokaltog/vim-powerline"

Bundle "godlygeek/tabular"

Bundle "msanders/snipmate.vim"

Bundle "jelera/vim-javascript-syntax"

Bundle "altercation/vim-colors-solarized"

Bundle "othree/html5.vim"

Bundle "xsbeats/vim-blade"

Bundle "Raimondi/delimitMate"

Bundle "groenewege/vim-less"

Bundle "evanmiller/nginx-vim-syntax"

Bundle "Lokaltog/vim-easymotion"

Bundle "tomasr/molokai"

Bundle "klen/python-mode"

Bundle "leafgarland/typescript-vim"

Bundle "scrooloose/nerdcommenter"

* [HTML5 Please](http://html5please.com/)：HTML5与CSS3技术评估
* [Mobile HTML5](http://mobilehtml5.org/)：HTML5兼容性速查表
* [HTML5 Cross Browser Polyfills](https://github.com/Modernizr/Modernizr/wiki/HTML5-Cross-Browser-Polyfills)

#### 2. 源代码管理工具

* [Github](https://github.com/)：声望日盛-全球最大的“同性”网站；
* [Github For Mac](https://desktop.github.com/)：设计的非常美观的git管理客户端，它能取代命令行所获得的功能；
* [Gitcafe](https://gitcafe.com/)：国内的Github，相比之下拥有速度优势；
* [Git@OSC](https://git.oschina.net/)：也是类似的项目，在国内，可以免费建立1000+私有项目；

#### 3. Mac工具

关于Mac平台的使用，推荐大家阅读[入门精选](http://wsgzao.github.io/post/mac/)

* [HomeBrew](http://brew.sh/index_zh-cn.html)：OS X上非常优秀的包管理工具；
* [HomeBrew-Cask](http://ksmx.me/homebrew-cask-cli-workflow-to-install-mac-applications/)：简洁优雅的Mac OS X软件安装体验；
* [iTerm 2](http://www.iterm2.com/index.html)：OS X上一个增强版的shell终端；
* [oh-my-zsh](http://ohmyz.sh/)：OS X上用来增强shell命令行的工具；
* [Dash](https://kapeli.com/dash)：Dash is an API Documentation Browser and Code Snippet Manager. Dash stores snippets of code and instantly searches offline documentation sets for 150+ APIs (for a full list, see below). You can even generate your own docsets or request docsets to be included；
* [tmux](https://aquaregia.gitbooks.io/tmux-productive-mouse-free-development_zh/content/index.html)：终端复用神器，帮助在Mac或者Linux上做开发的程序员在使用终端时更加得心应手；
* [MacVim](https://github.com/b4winckler/macvim)：Mac下GUI vim编辑器；
* [Xcode](https://developer.apple.com/xcode/downloads/)：也许你用不上，安装Xcode主要是为了Xcode command line tools工具包，你也可以选择不安装Xcode，使用xcode-select --install跳过Xcode的安装；

#### 4. VPN

* [云梯VPN](https://www.cloudtizi.com/)

#### 5. ES6编译

* [babel](https://github.com/babel/babel)

#### 6. gulp工具

* [gulp-clean](https://www.npmjs.com/package/gulp-clean)：用于清理;
* [gulp-notify](https://www.npmjs.com/package/gulp-notify)：用于打印消息文本;
* [gulp-rename](https://www.npmjs.com/package/gulp-rename)：用于修改名字;
* [gulp-concat](https://www.npmjs.com/package/gulp-concat)：用于合并文件;
* [gulp-zip](https://www.npmjs.com/package/gulp-zip)：用于生成一个zip压缩包;
* [gulp-minify-css](https://www.npmjs.com/package/gulp-minify-css)：用于压缩css;
* [gulp-autoprefixer](https://www.npmjs.com/package/gulp-autoprefixer)：用于给css添加前缀;
* [gulp-imagemin](https://www.npmjs.com/package/gulp-imagemin)：用于给图片进行优化;
* [gulp-uglify](https://www.npmjs.com/package/gulp-uglify)：用于压缩js;
* [amd-optimize](https://www.npmjs.com/package/amd-optimize)：用于AMD模块引用编译；
* [gulp-import-css](https://www.npmjs.com/package/gulp-import-css)：如果css文件是通过import导入的可以使用此插件进行合并优化;
* [gulp-rev-replace](https://www.npmjs.com/package/gulp-rev-replace)：用于替换;
* [gulp-useref](https://www.npmjs.com/package/gulp-useref)：引入使用build标记，进行替换;
* [gulp-rev](https://www.npmjs.com/package/gulp-rev)：生成md5文件名;
* [gulp-filter](https://www.npmjs.com/package/gulp-filter)：对文件进行过滤;
* [gulp-header](https://www.npmjs.com/package/gulp-header/)：压缩之后将注释写入到文件的头部
* [gulp-if](https://www.npmjs.com/package/gulp-if)：进行逻辑判断
* [gulp-size](https://www.npmjs.com/package/gulp-size)：获取文件大小

### Front-End Develop 开源项目

#### 1. 有趣味的开源项目

* [Web Developer技能树](https://github.com/phodal/skilltree)：类似《魔兽世界》中的天赋树；
* [octotree](https://github.com/buunguyen/octotree)：浏览器扩展(Chrome,Firefox,Safari和Opera)显示GitHub树格式的代码；
* [react-native](https://github.com/facebook/react-native)：出品的一个可以使用JavaScript来构建Native UI的库；
* [ionic](https://github.com/driftyco/ionic)：Ionic是一个强大的HTML5 原生应用（native app） 开发框架，帮助你用HTMLL，css和javascript构建具有原生味道的移动应用（mobile apps）；
* [passport-bnet](https://github.com/Blizzard/passport-bnet)：暴雪提供的Nodejs版SDK
* [hexo](https://github.com/hexojs/hexo)：编写博客使用的工具
* [traceur-compiler](https://github.com/google/traceur-compiler)：可以运行未来的JavaScript的JavaScript；
* [EpicEditor](https://github.com/OscarGodson/EpicEditor)：可以嵌入网页的Markdown编辑器；
* [gitbook](https://github.com/GitbookIO/gitbook)：用于生成在线书籍的工具；
* [kityminder](https://github.com/fex-team/kityminder)：作为一款在线的脑图编辑工具，它有着不亚于 native 脑图工具的交互体验；

#### 2. 开源库

* [three.js](https://github.com/mrdoob/three.js)：three.js是JavaScript编写的WebGL第三方库。提供了非常多的3D显示功能。Three.js 是一款运行在浏览器中的 3D 引擎，你可以用它创建各种三维场景，包括了摄影机、光影、材质等各种对象；
* [fullPage.js](https://github.com/alvarotrigo/fullPage.js)：fullPage.js 是一个基于 jQuery 的插件，它能够很方便、很轻松的制作出全屏网站；
* [angular-ui-router](https://github.com/angular-ui/ui-router)：angular-ui-router是AngularJS的路由框架，和默认的$route不同，它将所有路由包装成可划分层级的状态机状态,路由路径在ui-router中不是必须的。由于ui-router的路由状态机是分层级的，所以使用ui-router可以非常方便地创建包含多个嵌入的子模板；
* [marked](https://github.com/chjj/marked)：markdown解析器；
* [tsd](https://github.com/DefinitelyTyped/tsd)：在TypeScript中替换第三方库所使用的引用文件；
* [react](https://github.com/facebook/react)：可以用一种声明性、高效和灵活的JavaScript来构建用户界面的框架；
* [hammer.js](https://github.com/hammerjs/hammer.js)：处理触摸事件，以及手势的JavaScript库；
* [resumable.js](https://github.com/23/resumable.js)：这是用于HMLT5 File Api封装可以方便操作文件的一个库；
* [commander.js](https://github.com/tj/commander.js)：可以帮助快速开发Nodejs命令行工具的一个包；
* [persistencejs](https://github.com/coresmart/persistencejs)：是一个异步的Javascript库的对象关系映射框架；
* [artTemplate](https://github.com/aui/artTemplate)： JavaScript模板引擎；
* [TypeScript](https://github.com/Microsoft/TypeScript)：JavaScript的超集，扩展了类型，类，接口等语法，兼容ES6；
* [fastclick](https://github.com/ftlabs/fastclick)：FastClick是一个简单的、易于使用的库，消除在移动浏览器上触发click事件与一个物理Tap(敲击)之间的300延迟；
* [node-browserify](https://github.com/substack/node-browserify)：Browserify是一个node.js模块，主要用于改写现有的CommonJS模块，使得浏览器端也可以使用这些模块；
* [socket.io](https://github.com/socketio/socket.io)：Socket.io是使用得非常普遍的前端调用Websocket的库；
* [superagent](https://github.com/visionmedia/superagent)：superagent 是nodejs里一个非常方便的客户端请求代理模块；
* [cheerio](https://github.com/cheeriojs/cheerio)：它可以从一坨html的片断中构建DOM结构，然后提供像jquery一样的css选择器查询；
* [express](https://github.com/strongloop/express)：它用于构建Web网站，非常快速；
* [node-mongodb-native](https://github.com/mongodb/node-mongodb-native)：nodejs用于操作mongodb；
* [ejs](https://github.com/tj/ejs)：JavaScript模板引擎；
* [riot](https://github.com/riot/riot)：一个小巧的UI和代码结构的框架；
* [JavaScript-MD5](https://github.com/blueimp/JavaScript-MD5)：JS版的MD5;
* [zepto](https://github.com/madrobby/zepto)：在浏览器(Safari、Chrome和Firefox)上开发页面应用或者构建基于html的web-view本地应用，如PhoneGap，使用Zepto是一个不错的选择；
* [async](https://github.com/caolan/async)：JS实现的一个方便快捷异步编程的库；
* [echarts](https://github.com/ecomfe/echarts)：基于Canvas，纯Javascript图表库，提供直观，生动，可交互，可个性化定制的数据可视化图表。创新的拖拽重计算、数据视图、值域漫游等特性大大增强了用户体验，赋予了用户对数据进行挖掘、整合的能力；
* [koa](https://github.com/koajs/koa)：下一代编写Web网站的nodejs框架；
* [vue](https://github.com/vuejs/vue)：MVVM框架；
* [underscore](https://github.com/jashkenas/underscore)：JavaScript函数工具库；
* [backbone](https://github.com/jashkenas/backbone)：Backbone.js为复杂WEB应用程序提供模型(models)、集合(collections)、视图(views)的结构。其中模型用于绑定键值数据和自定义事件；集合附有可枚举函数的丰富API； 视图可以声明事件处理函数，并通过RESRful JSON接口连接到应用程序。
* [jquery](https://github.com/jquery/jquery)：jQuery 是一个高效、精简并且功能丰富的JavaScript 工具库；
* [angular.js](https://github.com/angular/angular.js)：AngularJS是为了克服HTML在构建应用上的不足而设计的，通过为开发者呈现一个更高层次的抽象来简化应用的开发。如同其他的抽象技术一样，这也会损失一部分灵活性。换句话说，并不是所有的应用都适合用AngularJS来做。AngularJS主要考虑的是构建CRUD应用。幸运的是，至少90%的WEB应用都是CRUD应用。但是要了解什么适合用AngularJS构建，就得了解什么不适合用AngularJS构建。
* [Flow](https://github.com/facebook/flow)：Facebook出品的一个JavaScript静态编译器；
* [avalon](https://github.com/RubyLouvre/avalon)：用于构建Web界面的MVVM JavaScript框架；
* [requirejs](https://github.com/jrburke/requirejs)：JavaScript模块化的开山鼻祖；
* [seajs](https://github.com/seajs/seajs)：中国开发者编写的一个JavaScript模块化库；
* [d3](https://github.com/mbostock/d3)：基于JavaScript编写的一个图表库，非常强大。
* [sugar.js](http://sugarjs.com/)：它扩展了现有的JS对象的方法，让你可以用更少的代码做更多的事情；
* [lo\_dash](https://lodash.com/)：JavaScript函数工具库；

#### 3. 开放平台

* [wx](https://github.com/baoshan/wx)：wx是极简设计的微信（公共平台）应用参考级框架，而并非微信接口在node.js下的幂等映射；
* [api-wow-docs](https://github.com/Blizzard/api-wow-docs)：暴雪提供的魔兽世界API文档

#### 4. 开源web网站系统

* [candy](https://github.com/guo-yu/candy)：基于多说社交评论的社会化论坛系统，采用 Node.js/Mongodb 构建；

#### 5. 调试工具

* [node-inspector](https://github.com/node-inspector/node-inspector)：nodejs的调试工具；
* [node-supervisor](https://github.com/petruisfan/node-supervisor)：nodejs用于无需重启的工具

## Front-End Questions

* [Front-end-Developer-Interview-Questions](https://github.com/h5bp/Front-end-Developer-Interview-Questions)：面试题集合；
* [中文前端开发面试题](https://github.com/markyun/My-blog/tree/master/Front-end-Developer-Questions) 中文前端开发面试题
* [Mars - mobile needs a hero](https://github.com/AlloyTeam/Mars)：腾讯出品的移动端实践；
* [mobileTech](https://github.com/jtyjty99999/mobileTech)：收集了移动端出现的问题以及解决思路与技巧；
* [移动web资源整理](http://www.cnblogs.com/PeunZhang/p/3407453.html)：移动web资源整理；

## Front-End 需要了解的数据结构与算法

* [数据结构与算法（JS 版） (@进击的Luke)](https://github.com/LukeLin/data-structure-with-js)
* [What are the 10 algorithms one must know in order to solve most algorithm problems](http://www.quora.com/What-are-the-10-algorithms-one-must-know-in-order-to-solve-most-algorithm-problems)
* [基础算法](https://www.coursera.org/course/spalgo)
* [那些少人所知而又有用的数据结构（StackOverflow）](http://stackoverflow.com/questions/500607/what-are-the-lesser-known-but-useful-data-structures)
* [高级数据结构大全](http://isa.unomaha.edu/wp-content/uploads/2012/08/Advanced-Data-structures.pdf)

## Front-End Develop 优秀文章集合

* [从码农周刊抓取的Front-End Develop 优秀文章集合](https://github.com/w3crange/Front-End-Develop-Guide/blob/master/Articles.md)

## 延伸可扩展的方向

延伸可扩展的方向是指，脱离了浏览器环境的编程。

关于编译，强烈建议阅读[工程中的编译原理--Jison入门篇](http://icodeit.org/2015/09/write-a-parser/)

#### Node.js

**入门**

* [Node入门](http://www.nodebeginner.org/index-zh-cn.html)
* [七天学会NodeJS](http://nqdeng.github.io/7-days-nodejs/)
* [Nodejs Wiki Book](https://github.com/nodejs-tw/nodejs-wiki-book)
* [Node.js 包教不包会](https://github.com/alsotang/node-lessons)

**文档**

* [nodejs中文文档](https://www.gitbook.com/book/0532/nodejs/details)
* [express.js 中文文档](http://expressjs.jser.us/)
* [koa 中文文档](https://github.com/guo-yu/koa-guide)
* [express框架](http://javascript.ruanyifeng.com/nodejs/express.html)

**阅读**

* [使用 Express + MongoDB 搭建多人博客](https://github.com/nswbmw/N-blog)
* [Learn You The Node.js For Much Win! (中文版)](https://www.npmjs.com/package/learnyounode-zh-cn)
* [Node debug 三法三例](http://i5ting.github.io/node-debug-tutorial/)
* [深入浅出Nodejs读书笔记](http://tw93.github.io/2015-03-01/shen-ru-qian-chu-nodejs-reading-mind-map.html)
* [NodeJS的代码调试和性能调优](http://www.barretlee.com/blog/2015/10/07/debug-nodejs-in-command-line/)
* [在 Node.js 应用中集成 Redis](http://www.ibm.com/developerworks/cn/opensource/os-cn-nodejs-redis/index.html)
* [Node.js 应用程序的 5 条性能建议](http://zhuanlan.zhihu.com/FrontendMagazine/20432208)

#### Mongodb

* [the-little-mongodb-book-cn](https://github.com/justinyhuang/the-little-mongodb-book-cn/blob/master/mongodb.md)

#### Chrome扩展开发

* [Chrome扩展及应用开发](http://www.ituring.com.cn/minibook/950)

#### 桌面应用程序开发

* [使用node-webkit构建桌面应用程序（一）](http://www.infoq.com/cn/articles/using-node-webkit-to-build-desktop-applications-part1)
* [使用node-webkit构建桌面应用程序（二）](http://www.infoq.com/cn/articles/using-node-webkit-to-build-desktop-applications-part2)
* [youdao出品的HEX](http://hex.youdao.com/zh-cn/tutorial/index.html)
* [electron](http://electron.atom.io/)

#### 移动应用程序开发

* [ionic中文指南](https://github.com/ychow/ionic-guide)
* [React Native 中文版](http://wiki.jikexueyuan.com/project/react-native/)

#### WebGL

* [WebGL 中文版](http://wiki.jikexueyuan.com/project/webgl/)

#### v8 引擎文献

可以先通过阅读[V8引擎简介](http://impd.tencent.com/?p=35)来了解JavaScript现在最好的引擎是怎样的，然后访问[Chrome V8](https://developers.google.com/v8/)来获取最新的资料，API文档可以访问[v8-docs](http://izs.me/v8-docs/)。

* [V8 Javascript 引擎设计理念](http://blog.pluskid.org/?p=186)
* [JavaScript引擎的性能优化](http://velocity.oreilly.com.cn/2015/2013/index.php?func=session&id=27)
* [V8引擎中的hidden class](https://github.com/BE-FE/Blog/blob/master/blogs/V8%E5%BC%95%E6%93%8E%E4%B8%AD%E7%9A%84hidden%20class.md)
* [JavaScript V8 性能小贴士](http://justjavac.com/javascript/2015/12/14/performance-tips-for-javascript-in-v8.html)

## Front-End Developer Blog

中文 Front-End 开发博客列表，本博客列表会长期维护，如果有推荐的博客，请到此处[提交博客信息](https://github.com/icepy/Front-End-Develop-Guide/issues/1)

| **博客地址** | **RSS地址** |
| --- | --- |
| [Plane Art](http://www.planeart.cn/) | <http://www.planeart.cn/?feed=rss2> |
| [王玮的博客](http://imvirgo.com/) | <http://imvirgo.com/feed/> |
| [UED TEAM](http://www.ued163.com/) | <http://www.ued163.com/?feed=rss2> |
| [随网之舞](http://dancewithnet.com/) | <http://feeds.feedburner.com/dancewithnet> |
| [不名一格](http://www.microidc.com/) | <http://www.microidc.com/feed/> |
| [Hugo Web前端开发](http://www.ghugo.com/) | <http://www.ghugo.com/feed/> |
| [阿里巴巴（中文站）用户体验设计部博客](http://www.aliued.cn/) | <http://www.aliued.cn/feed> |
| [何畏](http://www.lzlu.com/blog) | <http://www.lzlu.com/blog/?feed=rss2> |
| [PlanABC - 怿飞’s Blog](http://www.planabc.net/) | <http://feed.planabc.net/> |
| [Designsor](http://www.designsor.com/) | <http://www.designsor.com/feed> |
| [Time Machine](http://rlog.cn/) | <http://rlog.cn/?feed=rss2> |
| [为之漫笔](http://blog.cn-cuckoo.com/) | <http://www.cn-cuckoo.com/feed> |
| [博客园\_风流涕淌](http://www.cnblogs.com/Philoo) | <http://www.cnblogs.com/Philoo/rss> |
| [彬Go](http://blog.bingo929.com/) | <http://blog.bingo929.com/feed> |
| [Gracecode.com](http://www.gracecode.com/) | <http://feeds.gracecode.com/gracecode/> |
| [Koubei UED](http://ued.koubei.com/) | <http://ued.koubei.com/?feed=rss2> |
| [Alibaba.com UED](http://www.aliued.com/) | <http://www.aliued.com/feed/> |
| [Taobao UED Team](http://ued.taobao.org/blog/) | <http://ued.taobao.com/blog/feed/> |
| [博客园-振之博文](http://www.cnblogs.com/zhenn/rss) | <http://www.cnblogs.com/zhenn/> |
| [Kejun's Blog](http://hikejun.com/blog/) | <http://hikejun.com/blog/feed/> |
| [博客园\_纸头折飞机](http://www.cnblogs.com/xueduanyang/) | <http://www.cnblogs.com/xueduanyang/rss> |
| [最前端的生活](http://www.bq69.com/) | <http://www.bq69.com/feed> |
| [腾讯CDC](http://cdc.tencent.com/) | <http://cdc.tencent.com/?feed=rss2> |
| [博客园\_叶小钗](http://www.cnblogs.com/yexiaochai/) | <http://www.cnblogs.com/yexiaochai/rss> |
| [Hua's View](http://huaview.com/) | <http://huaview.com/feed> |
| [RainBlog-烟雨空间](http://rain520.net/) | <http://rain520.net/Feed/rss.xml> |
| [web前端开发工程师的随记--黑妞haha](http://www.heiniuhaha.cn/blog) | <http://www.heiniuhaha.cn/blog/?feed=rss2> |
| [蓝色理想](http://www.blueidea.com/) | <http://www.blueidea.com/common/rss.asp> |
| [博客园\_Ruby's Louvre](http://www.cnblogs.com/rubylouvre/) | <http://www.cnblogs.com/rubylouvre/rss> |
| [hax的技术部落格](http://hax.iteye.com/) | <http://hax.javaeye.com/rss> |
| [CssRain-前端技术](http://www.cssrain.cn/) | <http://www.cssrain.cn/?feed=rss2> |
| [MED](http://mux.alimama.com/) | <http://mux.alimama.com/feed/> |
| [W3CPLUS](http://www.w3cplus.com/) | <http://feed.feedsky.com/W3CPlus> |
| [博客园\_winter-cn](http://www.cnblogs.com/winter-cn/rss) | <http://www.cnblogs.com/winter-cn/> |
| [走走停停看看](http://shawphy.com/) | <http://shawphy.com/feed> |
| [Hi, I'm Adam Lu.](http://adamlu.com/) | <http://adamlu.com/?feed=rss2> |
| [WEB前端开发](http://www.css88.com/) | <http://www.css88.com/feed> |
| [岁月如歌](https://lifesinger.wordpress.com/) | <https://lifesinger.wordpress.com/feed/> |
| [Web开发中需要了解的东西](http://coolshell.cn/) | <http://coolshell.cn/feed> |
| [样式之美](http://www.aoao.org.cn/) | <http://feeds2.feedburner.com/aoao> |
| [時計坂一刻館三号室](https://quchao.com/) | <http://feed.quchao.com/> |
| [CSS森林(CSS Forest)](http://www.cssforest.org/blog/) | <http://www.cssforest.org/blog/index.php?atom=1> |
| [Ross Wan's World!](https://mrwlwan.wordpress.com/) | <https://mrwlwan.wordpress.com/feed/> |
| [博客园\_汤姆大叔的博客](http://www.cnblogs.com/TomXu/) | <http://www.cnblogs.com/TomXu/rss> |
| [前端观察](https://www.qianduan.net/) | <https://www.qianduan.net/rss/> |
| [张云龙个人博客](https://github.com/fouber/blog) | <https://github.com/fouber/blog> |

## 通用Developer 社区

在Github上找到一个社区网站收集的项目-- [front-end-collect](https://github.com/foru17/front-end-collect)

中文 Developer 社区列表，本社区列表会长期维护，如果有推荐的社区，请到此处[提交社区信息](https://github.com/icepy/Front-End-Develop-Guide/issues/2)

| **中文 Developer 社区列表** | **地址** |
| --- | --- |
| [领略前端技术 阅读奇舞周刊](http://www.75team.com/weekly/) | <http://www.75team.com/weekly/> |
| [div.io](http://div.io/#/welcome) | <http://div.io/#/welcome> |
| [稀土掘金](http://gold.xitu.io/) | [http://gold.xitu.io](http://gold.xitu.io/) |
| [开发者头条](http://toutiao.io/) | <http://toutiao.io/> |
| [码农周刊](http://weekly.manong.io/) | <http://weekly.manong.io/> |
| [InfoQ](http://www.infoq.com/cn/) | <http://www.infoq.com/cn/> |
| [Segmentfault](http://segmentfault.com/) | <http://segmentfault.com/> |
| [Get社区](http://get.ftqq.com/) | <http://get.jobdeer.com/> |
| [极客头条](http://geek.csdn.net/hotest) | <http://geek.csdn.net/hotest> |
| [极客头条－前端](http://geek.csdn.net/forum/47) | <http://geek.csdn.net/forum/47> |

## 技术分享会议

说明：技术分享会议引用了[ufologist](https://github.com/ufologist)项目中的收集，与自己收集的资源整理而成。

| **会议组织列表** | **地址** |
| --- | --- |
| [D2前端技术论坛](http://www.d2forum.org/) | <http://www.d2forum.org/> |
| [WebReBuild](http://webrebuild.org/) | [http://webrebuild.org](http://webrebuild.org/) |
| [Velocity](http://velocity.oreilly.com.cn/) | [http://velocity.oreilly.com.cn](http://velocity.oreilly.com.cn/) |
| [JSCONF.CN](http://jsconf.cn/) | <http://jsconf.cn/> |
| [阿里技术嘉年华](http://adc.taobao.com/) | [http://adc.taobao.com](http://adc.taobao.com/) |
| [中国CSS开发者大会](http://css.w3ctech.com/) | <http://css.w3ctech.com/> |
| [w3ctech](http://www.w3ctech.com/event) | <http://www.w3ctech.com/event> |
| [HTML5梦工厂](http://www.html5dw.com/) | <http://www.html5dw.com/> |
| [前端圈](http://www.fequan.com/) | <http://www.fequan.com/> |
| [QCon北京](http://qconbeijing.com/) | <http://qconbeijing.com/> |
| [QCon上海](http://qconshanghai.com/) | <http://qconshanghai.com/> |
| [QCon@InfoQ](http://www.infoq.com/cn/qcon) | <http://www.infoq.com/cn/qcon> |
| [ArchSummit](http://www.archsummit.com/) | <http://www.archsummit.com/> |
| [阿里技术沙龙](http://club.alibabatech.org/) | [http://club.alibabatech.org](http://club.alibabatech.org/) |
| [Apple WWDC](https://developer.apple.com/wwdc/) | <https://developer.apple.com/wwdc/> |
| [Google I/O](https://events.google.com/io2015/) | <https://events.google.com/io2015/> |

# Front-End Develop 项目精选

站在个人的角度，针对收录过的开源项目做了一个甄别、筛选，标准是个人使用过。

由[@icepy](http://weibo.com/2455876310)负责整理

说明：顺序不分排名先后。

| **项目名称** | **开发者&组织** | **备注** |
| --- | --- | --- |
| [jQuery](https://github.com/jquery/jquery) | [jQuery Foundation](https://github.com/jquery) | jQuery 是一个高效、精简并且功能丰富的JavaScript 工具库； |
| [Zepto](https://github.com/madrobby/zepto/) | [madrobby](https://github.com/madrobby) | Zepto是一个轻量级的针对现代高级浏览器的JavaScript库， 它与jquery有着类似的api。 如果你会用jquery，那么你也会用zepto； |
| [fullPage.js](https://github.com/alvarotrigo/fullPage.js) | [alvarotrigo](https://github.com/alvarotrigo) | fullPage.js 是一个基于 jQuery 的插件，它能够很方便、很轻松的制作出全屏网站； |
| [AngularUI Router](https://github.com/angular-ui/ui-router) | [AngularUI](https://github.com/angular-ui) | angular-ui-router是AngularJS的路由框架，和默认的$route不同，它将所有路由包装成可划分层级的状态机状态,路由路径在ui-router中不是必须的。由于ui-router的路由状态机是分层级的，所以使用ui-router可以非常方便地创建包含多个嵌入的子模板； |
| [React](https://github.com/facebook/react) | [Facebook](https://github.com/facebook) | 这是该公司用于构建反应式图形界面的JavaScript库; |
| [react-redux](https://github.com/rackt/react-redux) | [rackt](https://github.com/rackt) | Flux架构实现 |
| [react-router](https://github.com/rackt/react-router) | [rackt](https://github.com/rackt) | React路由系统 |
| [TSD](https://github.com/DefinitelyTyped/tsd) | [DefinitelyTyped](https://github.com/DefinitelyTyped) | 在TypeScript中替换第三方库所使用的引用文件; |
| [TypeScript](https://github.com/Microsoft/TypeScript) | [Microsoft](https://github.com/Microsoft) | JavaScript的超集，扩展了类型，类，接口等语法，兼容ES6； |
| [Commander.js](https://github.com/tj/commander.js) | [tj](https://github.com/tj) | 可以帮助快速开发Nodejs命令行工具的一个包 |
| [FastClick](https://github.com/ftlabs/fastclick) | [FT Labs](https://github.com/ftlabs) | FastClick是一个简单的、易于使用的库，消除在移动浏览器上触发click事件与一个物理Tap(敲击)之间的300延迟； |
| [browserify](https://github.com/substack/node-browserify) | [substack](https://github.com/substack) | browserify是一个node.js模块，主要用于改写现有的CommonJS模块，使得浏览器端也可以使用这些模块； |
| [socket.io](https://github.com/socketio/socket.io) | [Socket.IO](https://github.com/socketio) | Socket.io是使用得非常普遍的前端调用Websocket的库； |
| [express](https://github.com/strongloop/express) | [StrongLoop, Inc.](https://github.com/strongloop) | 它用于构建Web网站，非常快速； |
| [React Native](https://github.com/facebook/react-native) | [Facebook](https://github.com/facebook) | 该公司出品的一个可以使用JavaScript来构建Native UI的库； |
| [Flow](https://github.com/facebook/flow) | [Facebook](https://github.com/facebook) | 该公司出品的一个JavaScript静态编译器 |
| [ejs](https://github.com/tj/ejs) | [tj](https://github.com/tj) | JavaScript模板引擎； |
| [artTemplate](https://github.com/aui/artTemplate) | [aui](https://github.com/aui) | JavaScript模板引擎； |
| [underscore](https://github.com/jashkenas/underscore) | [jashkenas](https://github.com/jashkenas) | JavaScript函数工具库； |
| [avalon](https://github.com/RubyLouvre/avalon) | [RubyLouvre](https://github.com/RubyLouvre) | 用于构建Web界面的MVVM JavaScript框架； |
| [backbone](https://github.com/jashkenas/backbone) | [jashkenas](https://github.com/jashkenas) | Backbone.js为复杂WEB应用程序提供模型(models)、集合(collections)、视图(views)的结构。其中模型用于绑定键值数据和自定义事件；集合附有可枚举函数的丰富API； 视图可以声明事件处理函数，并通过RESRful JSON接口连接到应用程序； |
| [angular.js](https://github.com/angular/angular.js) | [Angular](https://github.com/angular) | AngularJS是为了克服HTML在构建应用上的不足而设计的，通过为开发者呈现一个更高层次的抽象来简化应用的开发。如同其他的抽象技术一样，这也会损失一部分灵活性。换句话说，并不是所有的应用都适合用AngularJS来做。AngularJS主要考虑的是构建CRUD应用。幸运的是，至少90%的WEB应用都是CRUD应用。但是要了解什么适合用AngularJS构建，就得了解什么不适合用AngularJS构建。 |
| [requirejs](https://github.com/jrburke/requirejs) | [jrburke](https://github.com/jrburke) | JavaScript模块化的开山鼻祖； |
| [seajs](https://github.com/seajs/seajs) | [seajs](https://github.com/seajs) | JavaScript模块化库； |
| [three.js](https://github.com/mrdoob/three.js) | [mrdoob](https://github.com/mrdoob) | three.js是JavaScript编写的WebGL第三方库，提供了非常多的3D显示功能。 |
| [echarts](https://github.com/ecomfe/echarts) | [Baidu EFE team](https://github.com/ecomfe) | 基于Canvas，纯Javascript图表库，提供直观，生动，可交互，可个性化定制的数据可视化图表； |
| [d3](https://github.com/mbostock/d3) | [mbostock](https://github.com/mbostock) | 基于JavaScript编写的一个图表库，非常强大； |
| [vue](https://github.com/yyx990803/vue) | [yyx990803](https://github.com/yyx990803) | 用于构建Web界面的MVVM JavaScript框架； |
| [ionic](https://github.com/driftyco/ionic) | [Ionic](https://github.com/driftyco) | Ionic是一个强大的HTML5 原生应用（native app） 开发框架，帮助你用HTMLL，css和javascript构建具有原生味道的移动应用（mobile apps）； |
| [koa](https://github.com/koajs/koa) | [koajs](https://github.com/koajs) | 下一代Node.js Web开发框架 |
| [amazeui](https://github.com/amazeui/amazeui) | [amazeui](https://github.com/amazeui) | 国人出品的HTML5框架 |

# Front-End Develop 文章精选

站在个人的角度，针对收录过的文章做了一个甄别、筛选。目的是摒弃杂音，更利于提高学习效率。

由[@icepy](http://weibo.com/2455876310)负责整理

### 教程类

零基础的同学非常适合在学JavaScript之前阅读它：[重新介绍 JavaScript（JS 教程）](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/A_re-introduction_to_JavaScript)，然后接着阅读[消除一些经常被误解的概念](http://bolinfest.com/javascript/misunderstood.html)。

* [译：理解并掌握 JavaScript 中 this 的用法](http://code.mforever78.com/translation/2015/05/19/understand-javascripts-this-with-clarity-and-master-it/)
* [理解JavaScript的原型链和继承](http://blog.oyanglul.us/javascript/understand-prototype.html)
* [理解js闭包](http://www.w3cfuns.com/blog-5435393-5399202.html)
* [细嗅Promise](http://www.cnblogs.com/hustskyking/p/promise.html)
* [实战JS正则表达式](http://www.cnblogs.com/skylar/p/4265490.html)

37个JavaScript题目，看看你能做正确多少个：[JavaScript Puzzlers](http://javascript-puzzlers.herokuapp.com/)

编程风格指南推荐：[Airbnb JavaScript Style Guide](https://github.com/yuche/javascript)

### ES6 前瞻

个人认为是时候开始学习了，可以先了解一下有哪些特性，[Essential EcmaScript 6](http://blog.oyanglul.us/javascript/essential-ecmascript6.html)

* [ECMAScript 6入门](http://es6.ruanyifeng.com/)
* [ECMAScript 6 Features 中文版](https://github.com/ES-CN/es6features/blob/master/README.md)
* [ECMAScript 6 Tools](https://github.com/addyosmani/es6-tools)

### 深入浅出ES6系列

从[InfoQ中文站](http://www.infoq.com/cn/minibooks/ES6-in-Depth)收集而来

* [深入浅出ES6（一）：ES6是什么](http://www.infoq.com/cn/articles/es6-in-depth-an-introduction)
* [深入浅出ES6（二）：迭代器和for-of循环](http://www.infoq.com/cn/articles/es6-in-depth-iterators-and-the-for-of-loop)
* [深入浅出ES6（三）：生成器 Generators](http://www.infoq.com/cn/articles/es6-in-depth-generators)
* [深入浅出ES6（四）：模板字符串](http://www.infoq.com/cn/articles/es6-in-depth-template-string)
* [深入浅出ES6（五）：不定参数和默认参数](http://www.infoq.com/cn/articles/es6-in-depth-rest-parameters-and-defaults)
* [深入浅出ES6（六）：解构 Destructuring](http://www.infoq.com/cn/articles/es6-in-depth-destructuring)
* [深入浅出ES6（七）：箭头函数 Arrow Functions](http://www.infoq.com/cn/articles/es6-in-depth-arrow-functions)
* [深入浅出ES6（八）：Symbols](http://www.infoq.com/cn/articles/es6-in-depth-symbols)
* [学习Babel和Broccoli，马上就用ES6](http://www.infoq.com/cn/articles/es6-in-depth-babel-and-broccoli)
* [深入浅出ES6（十）：集合](http://www.infoq.com/cn/articles/es6-in-depth-collections?utm_campaign=rightbar_v2&utm_source=infoq&utm_medium=articles_link&utm_content=link_text)
* [深入浅出ES6（十一）：生成器 Generators，续篇](http://www.infoq.com/cn/articles/es6-in-depth-generators-continued)
* [深入浅出ES6（十二）：代理 Proxies](http://www.infoq.com/cn/articles/es6-in-depth-proxies-and-reflect)
* [深入浅出ES6（十三）：类 Class](http://www.infoq.com/cn/articles/es6-in-depth-classes)
* [深入浅出ES6（十四）：let和const](http://www.infoq.com/cn/articles/es6-in-depth-let-and-const)
* [深入浅出ES6（十五）：子类 Subclassing](http://www.infoq.com/cn/articles/es6-in-depth-subclassing)
* [深入浅出ES6（十六）：模块 Modules](http://www.infoq.com/cn/articles/es6-in-depth-modules)
* [深入浅出ES6（十七）：展望未来](http://www.infoq.com/cn/articles/es6-in-depth-the-future)