前端开发面试知识点大纲：

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、选择器优先级及使用、HTML5、CSS3、移动端适应

JavaScript：

数据类型、面向对象、继承、闭包、插件、作用域、跨域、原型链、模块化、自定义事件、内存泄漏、事件机制、异步装载回调、模板引擎、Nodejs、JSON、ajax 等。

其他：

HTTP、安全、正则、优化、重构、响应式、移动端、团队协作、可维护、SEO、UED、架构、职业生涯

作为一名前端工程师，无论工作年头长短都应该必须掌握的知识点：

此条由 王子墨 发表在 前端随笔

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。

资料答案不够正确和全面，欢迎补充答案、题目；最好是现在网上没有的。

HTML

•Doctype 作用？严格模式与混杂模式如何区分？它们有何意义？

（1）、<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签

之前。告知浏览器的解析器，

用什么文档类型 规范来解析这个文档。

（2）、严格模式的排版和 JS 运作模式是 以该浏览器支持的最高标准运行。

（3）、在混杂模式中，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。

（4）、DOCTYPE 不存在或格式不正确会导致文档以混杂模式呈现。

http://www.w3help.org/zh-cn/casestudies/002

•行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？

（1）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，

比如 div 默认 display 属性值为“block”，成为“块级”元素；

span 默认 display 属性值为“inline”，是“行内”元素。

（2）行内元素有：a b span img input select strong（强调的语气）

块级元素有：div ul ol li dl dt dd h1 h2 h3 h4…p

（3）知名的空元素：

 <hr> <img> <input> <link> <meta>

鲜为人知的是：

<area> <base> <col> <command> <embed> <keygen>

<param> <source> <track> <wbr>

•link 和@import 的区别是？

（1）link 属于 XHTML 标签，而@import 是 CSS 提供的;

（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 只在 IE5 以上才能识别，而 link 是 XHTML 标签，无兼容问题;

（4）link 方式的样式的权重 高于@import 的权重.

•浏览器的内核分别是什么?

* IE 浏览器的内核 Trident、Mozilla 的 Gecko、Chrome 的 Blink（WebKit 的分支）、Opera 内核原为 Presto，现为 Blink；

•常见兼容性问题？

•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 、aside

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、使用最多的是 html5shim 框架

```
<!--[if lt IE 9]>
<script>
src="http://html5shim.googlecode.com/svn/trunk/html5.js"</script>
```

<![endif]-->

如何区分： DOCTYPE 声明\新增的结构元素\功能元素

•语义化的理解？

用正确的标签做正确的事情！

html 语义化就是让页面的内容结构化，便于对浏览器、搜索引擎解析；

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。

搜索引擎的爬虫依赖于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？

localStorage 长期存储数据，浏览器关闭后数据不丢失；

sessionStorage 数据在浏览器关闭后自动删除。

•(写)描述一段语义的 html 代码吧。

（HTML5 中新增加的很多标签（如：<article>、<nav>、<header>和<footer>等）

就是基于语义化设计原则）

```
< div id="header">
< h1>标题< /h1>
< h2>专注 Web 前端技术< /h2>
```

< /div>

•iframe 有那些缺点？

*iframe 会阻塞主页面的 Onload 事件；

*iframe 和主页面共享连接池，而浏览器对相同域的连接有限制(6-8 目前)，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，这样可以可以绕开以上两个问题。

•请描述一下 cookies，sessionStorage 和 localStorage 的区别？

cookie 在浏览器和服务器间来回传递。 sessionStorage 和 localStorage 不会

sessionStorage 和 localStorage 的存储空间更大；

sessionStorage 和 localStorage 有更多丰富易用的接口；

sessionStorage 和 localStorage 各自独立的存储空间；

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？优先级算法如何计算？

CSS3 新增伪类有那些？

* 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 ;

* 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？

•给 div 设置一个宽度，然后添加 margin:0 auto 属性

```
1. div{
2.     width:200px;
3.     margin:0 auto;
4. }
```

•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

```
1. .div {
2.   Width:500px ; height:300px;//高度可以不设
3.   Margin: -150px 0 0 -250px;
4.   position:relative;相对定位
5.   background-color:pink;//方便看效果
6.   left:50%;
7.   top:50%;
8. }
```

•CSS3 有哪些新特性？

CSS3 实现圆角（border-radius:8px），阴影（box-shadow:10px），

对文字加特效（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

•一个满屏 品 字布局 如何设计?

•经常遇到的 CSS 的兼容性有哪些？原因，解决方法是什么？

1. 就是 ie6 双倍边距的问题，在使用了 float 的情况下，不管是向左还是向右都会出现双倍，最简单的解决方法就是用 display:inline;加到 css 里面去。
2. 文字本身的大小不兼容。同样是 font-size:14px 的宋体文字，在不同浏览器下占的空间是不一样的，ie 下实际占高 16px，下留白 3px，ff 下实际占高 17px，上留白 1px，下留白 3px，opera 下就更不一样了。解决方案：给文字设定 line-height 。确保所有文字都有默认的 line-height 值。这点很重要，在高度上我们不能容忍 1px 的差异。
3. ff 下容器高度限定，即容器定义了 height 之后，容器边框的外形就确定了，不会被内容撑大，而 ie 下是会被内容撑大，高度限定失效。所以不要轻易给容器定义 height。
4. 还讨论内容撑破容器问题，横向上的。如果 float 容器未定义宽度，ff 下内容会尽可能撑开容器宽度，ie 下则会优先考虑内容折行。故，内容可能撑破的浮动容器需要定义 width。
5. 浮动的清除，ff 下不清除浮动是不行的。
6. mirrormargin bug，当外层元素内有 float 元素时，外层元素如定义 margin-top:14px，将自动生成 margin-bottom:14px。padding 也会出现类似问题，都是 ie6 下的特产，该类 bug 出现的情况较为复杂，远不只这一种出现条件，还没系统整理。解决方案：外层元素设定 border 或 设定 float。
7. 吞吃现象，限于篇幅，我就不展开了。还是 ie6，上下两个 div，上面的 div 设置背景，却发现下面没有设置背景的 div 也有了背景，这就是吞吃现象。对应上面的背景吞吃现象，还有滚动下边框缺失的现象。解决方案：使用 zoom:1。这个 zoom 好象是专门为解决 ie6 bug 而生的。
8. 注释也能产生 bug~~~"多出来的一只猪。"这是前人总结这个 bug 使用的文案，ie6 的这个 bug 下，大家会在页面看到猪字出现两遍，重复的内容量因注释的多少而变。解决方案：用"<!–[if !IE]> picRotate start <![endif]–>"方法写注释。
9. 里加 float <div/>，这是一个典型的，棘手的兼容问题，希望引起大家正视 ，给 li 不同的属性会有不同的解释效果，ff 下的解释稍可理解，ie6 下的解释会让你摸不着头脑，由于问题的复杂性，将另起一文专门讨论该问题。在《ul 使用心得》一文里有相关成果，却没给出问题解决的过程。
10. img 下的留白。解决方案：给 img 设定 display:block。
11. 失去 line-height。<div style="line-height:20px"><img />文字 </div>，很遗憾，在 ie6 下单行文字 line-height 效果消失了。。。，原因是<img />这个 inline-block 元素和 inline 元素写在一起了。解决方案：让 img 和文字都 float 起来。
12. 链接的 hover 状态。a:hover img{width:300px} 我们想让鼠标 hover 时，链接里包含的图片宽度变化，可惜在 ie6 下无效，ie7、ff 下有效。
13. 非链接的 hover 状态。div:hover{} 这样的样式 ie6 是不认的，在 ie7、ff 下才有效果。
14. ie 下 overflow:hidden 对其下的绝对层 position:absolute 或者相对层 position:relative 无效。解决方案：给 overflow:hidden 加 position:relative 或者 position:absolute。另，ie6 支持 overflow-x 或者 overflow-y 的特性，ie7、ff 不支持。

15. ie6 下严重的 bug，float 元素如没定义宽度，内部如有 div 定义了 height 或 zoom:1，这个 div 就会占满一整行，即使你给了宽度。float 元素如果作为布局用或复杂的容器，都要给个宽度的。
16. ie6 下的 bug，绝对定位的 div 下包含相对定位的 div，如果给内层相对定位的 div 高度 height 具体值，内层相对层将具有 100%的 width 值，外层绝对层将被撑大。解决方案给内层相对层 float 属性。
17. ie6 下的 bug，<head></head>内有<base target=”_blank”/>的情况下，position:relative 层下的 float 层内文字无法选中。
18. 终于来了个 ff 的缺点。width:100%这个东西在 ie 里用很方便，会向上逐层搜索 width 值，忽视浮动层的影响，ff 下搜索至浮动层结束，如此，只能给中间的所有浮动层加 width:100%才行，累啊。opera 这点倒学乖了跟了 ie。

•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就是： * {padding: 0; margin: 0;} （不建议）

淘宝的样式初始化：

1. 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; }
2.
3. body, button, input, select, textarea { font:12px/1.5tahoma, arial, \5b8b\4f53; }
4.
5. h1, h2, h3, h4, h5, h6{ font-size:100%; }
6.
7. address, cite, dfn, em, var { font-style:normal; }
8.

```
9.  code, kbd, pre, samp { font-family:couriernew, courier, monospace; }
10.
11. small{ font-size:12px; }
12.
13. ul, ol { list-style:none; }
14.
15. a { text-decoration:none; }
16.
17. a:hover { text-decoration:underline; }
18.
19. sup { vertical-align:text-top; }
20.
21. sub{ vertical-align:text-bottom; }
22.
23. legend { color:#000; }
24.
25. fieldset, img { border:0; }
26.
27. button, input, select, textarea { font-size:100%; }
28.
29. table { border-collapse:collapse; border-spacing:0; }
```

- absolute 的 containing block 计算方式跟正常流有什么不同？

http://www.w3help.org/zh-cn/kb/008/

•列出 display 的值，说明他们的作用。position 的值， relative 和 absolute 定位原点是？

1.

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。

值	描述
none	此元素不会被显示。
block	此元素将显示为块级元素，此元素前后会带有换行符。
inline	默认。此元素会被显示为内联元素，元素前后没有换行符。
inline-block	行内块元素。（CSS2.1 新增的值）
list-item	此元素会作为列表显示。
run-in	此元素会根据上下文作为块级元素或内联元素显示。
compact	CSS 中有值 compact，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
marker	CSS 中有值 marker，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
table	此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。
inline-table	此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。
table-row-group	此元素会作为一个或多个行的分组来显示（类似 <tbody>）。
table-header-group	此元素会作为一个或多个行的分组来显示（类似 <thead>）。
table-footer-group	此元素会作为一个或多个行的分组来显示（类似 <tfoot>）。
table-row	此元素会作为一个表格行显示（类似 <tr>）。
table-column-group	此元素会作为一个或多个列的分组来显示（类似

	<colgroup>）。
table-column	此元素会作为一个单元格列显示（类似 <col>）
table-cell	此元素会作为一个表格单元格显示（类似 <td> 和 <th>）
table-caption	此元素会作为一个表格标题显示（类似 <caption>）
inherit	规定应该从父元素继承 display 属性的值。

2.

值	描述
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。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
fixed	生成绝对定位的元素，相对于浏览器窗口进行定位。 元素的位置通过 "left", "top", "right" 以及 "bottom" 属性进行规定。
relative	生成相对定位的元素，相对于其正常位置进行定位。 因此，"left:20" 会向元素的 LEFT 位置添加 20 像素。
static	默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
inherit	规定应该从父元素继承 position 属性的值。

什么是文档流？

将窗体自上而下分成一行行，并在每行中按从左至右的顺序排放元素,即为文档流。

只有三种情况会使得元素脱离文档流，分别是：浮动、绝对定位和相对定位。
使用 absoulte 或 fixed 定位的话，必须指定 left、right、 top、 bottom 属性中的至少一个，否则 left/right/top/bottom 属性会使用它们的默认值 auto ，这将导致对象遵从正常的 HTML 布局规则，在前一个对象之后立即被呈递，简单讲就是都变成 relative，会占用文档空间，这点非常重要，很多人使用 absolute 定位后发现没有脱离文档流就是这个原因。

•position 跟 display、margin collapse、overflow、float 这些特性相互叠加后会怎么样？

position 控制元素是否在文档流中，static 和 relative 都存在于文档流，fixed 直接脱离文档流锁定在以整个文档为 offsetParent 的位置，而 absolute 也是直接脱离文档流锁定在最近的一个有 position 属性的 parent 的位置上。

display 控制布局方式，block 以 block 方式布局，inline 以 inline 方式布局，inline-block 以 inline-block 方式布局，而所有的布局方式的前提是元素在文档流当中，所以如果 position 设置为脱离文档流，display 就只影响子元素的布局了。display:none 直接让元素脱离文档流一样的布局方式。
margin collapse 也是在文档流时才会有效。
overflow 指的是当超出布局边界时的显示效果，跟怎么布局没关系。
float 也是直接脱离文档流了。
margin collapse 在 position 跟 display、overflow、float 下是怎么展现的？'display'、'position' 和 'float' 的相互关系。

两个或多个毗邻的普通流中的块元素垂直方向上的 margin 会折叠

浮动元素、inline-block 元素、绝对定位元素的 margin 不会和垂直方向上其他元素的 margin 折叠

创建了块级格式化上下文的元素，不和它的子元素发生 margin 折叠

元素自身的 margin-bottom 和 margin-top 相邻时也会折叠

http://www.w3cplus.com/css/understanding-bfc-and-margin-collapse.html
http://blog.csdn.net/chen_zw/article/details/8741365
http://www.cnblogs.com/joe235/archive/2011/03/03/1970006.html
http://www.w3help.org/zh-cn/kb/006/

•对 BFC 规范的理解？
（W3C CSS 2.1 规范中的一个概念,它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。）
什么是 BFC(Block Formatting Context)，简单讲，它是提供了一个独立布局的环境，每个 BFC 都遵守同一套布局规则。例如，在同一个 BFC 内，盒子会一个挨着一个的排，相邻盒子的间距是由 margin 决定且垂直方向的 margin 会重叠。而 float 和 clear float 也只对同一个 BFC 内的元素有效。
浮动元素和绝对定位元素，非块级盒子的块级容器（例如 inline-blocks, table-cells, 和 table-captions），以及 overflow 值不为"visiable"的块级盒子，都会为他们的内容创建新的 BFC（块级格式上下文）。在 BFC 中，盒子从顶端开始垂直地一个接一个地排列，两个盒子之间的垂直的间隙是由他们的 margin 值所决定的。在一个 BFC 中，两个相邻的块级盒子的垂直外边距会产生折叠。在 BFC 中，每一个盒子的左外边缘（margin-left）会触碰到容器的左边缘(border-left)（对于从右到左的格式来说，则触碰到右边缘）。

•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：

```
1. /*权重为 1*/
2. div{
3.
4. }
5.
6. /*权重为 10*/
7. .class1{
8.
9. }
10.
11./*权重为 100*/
12.#id1{
13.
14.}
15.
16./*权重为 100+1=101*/
17.#id1 div{
18.
19.}
20.
21./*权重为 10+1=11*/
22..class1 div{
23.
```

```
24.}
25.
26./*权重为 10+10+1=21*/
27..class1 .class2 div{
28.
29.}
```

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

•解释下浮动和它的工作原理？清除浮动的技巧

浮动元素脱离文档流，不占据文档空间。 浮动"从流程中被移除出来", 但是与绝对位置的元素（层次）不同，浮动是在他们前面的最后一个块元素之后直接被显示出来（就像块盒一样）。如果该浮动是在一个"行块"中，该浮动的上边界被放置在行块顶部的水平上。当除此以外，浮动与绝对元素相似，原先的块盒会完全忽略浮动和 AP 元素。那些静态的块盒知识保持一个接一个地"跟随"，就好像没有浮动不在那里一样。

1.使用空标签清除浮动。

这种方法是在所有浮动标签后面添加一个空标签 定义 css clear:both. 弊端就是增加了无意义标签。

2.使用 overflow。

给包含浮动元素的父标签添加 css 属性 overflow:auto; zoom:1; zoom:1 用于兼容 IE6。

3.使用 after 伪对象清除浮

动。 #layout:after{display:block;clear:both;content:"";visibility:hidden;height:0;}

我们将现有已知的清楚浮动元素方法罗列下：

1. 采用一个 HTML 标签，以及 css 的 clear 属性，来手工清理浮动；
2. 采用伪类:after，动态建立一个块元素，设定 clear 属性，清理之前的浮动元素；
3. 采用 CSS overflow 非 visible 值(overflow:auto/overflow:hidden)设定使父容器包含浮动元素；
4. 采用 display:table/display:table-cell 等 table 系列属性将父元素变成 table 形式自动包含浮动元素；
5. 使用 TABLE 以及 TD 标签作为浮动元素容器；
6. 采用 float:left/float:right 方式将父元素同样浮动，就可以包含浮动内容；
7. 在 IE 6/7 的标准文档模式中设置“width/height/zoom”等样式来自动清理浮动。

•用过媒体查询，针对移动端的布局吗？

•使用 CSS 预处理器吗？喜欢那个？

Sass、LESS 和 Stylus

JavaScript

•JavaScript 原型，原型链 ？有什么特点？

构造函数，原型对象，实例的关系是：JavaScript 中，每个函数都有一个 prototype 属性，这是一个指针，指向了这个函数的原型对象。

这个对象包含这个函数创建的实例的共享属性和方法。也就是说原型对象中的属性和方法是所有实例共享。而这个原型对象有一个 constructor 属性，指向了该构造函数。每个通过该构造函数创建的对象都包含一个指向原型对象的内部指针__proto__。 原型链作为实现继承的主要方法，其基本思想是：让原型对象等于另一个类型的实例，这样原型对象将包含一个指向另一个原型的指针，相应的，另一个原型中也包含着一个指向另一个构造函数的指针，假如另一个原型又是另一个类型的实例，如此层层递进，就构成了实例与原型的链条，这个链条就称之为**原型链.**

•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）。

•null，undefined 的区别？

null

这是一个对象，但是为空。因为是对象，所以 typeof null 返回 'object' 。

null 是 JavaScript 保留关键字。

null 参与数值运算时其值会自动转换为 0 ，因此，下列表达式计算后会得到正确的数值：

表达式：123 + null 结果值：123

表达式：123 * null 结果值：0

undefined

undefined 是全局对象（window）的一个特殊属性，其值是未定义的。但 typeof undefined 返回 'undefined' 。

虽然 undefined 是有特殊含义的，但它确实是一个属性，而且是全局对象（window）的属性。请看下面的代码：

```
alert('undefined' in window);  //输出：true
 var anObj = {};
 alert('undefined' in anObj);  //输出：false
```

从中可以看出，undefined 是 window 对象的一个属性，但却不是 anObj 对象的一个属性。

•写一个通用的事件侦听器函数。

1. // event(事件)工具集，来源：github.com/markyun

```
2.  markyun.Event = {
3.      // 页面加载完成后
4.      readyEvent : function(fn) {
5.          if (fn==null) {
6.              fn=document;
7.          }
8.
9.          var oldonload = window.onload;
10.         if (typeof window.onload != 'function') {
11.             window.onload = fn;
12.         } else {
13.             window.onload = function() {
14.                 oldonload();
15.                 fn();
16.             };
17.         }
18.     },
19.
20.     // 视能力分别使用 dom0||dom2||IE 方式 来绑定事件
21.
22.     // 参数： 操作的元素,事件名称 ,事件处理程序
23.     addEvent : function(element, type, handler) {
24.         if (element.addEventListener) {
25.             //事件类型、需要执行的函数、是否捕捉
26.             element.addEventListener(type, handler, false);
```

```
27.         } else if (element.attachEvent) {
28.             element.attachEvent('on' + type, function() {
29.                 handler.call(element);
30.             });
31.         } else {
32.             element['on' + type] = handler;
33.         }
34.     },
35.
36.     // 移除事件
37.     removeEvent：function(element, type, handler) {
38.         if (element.removeEnentListener) {
39.             element.removeEnentListener(type, handler, false);
40.         } else if (element.datachEvent) {
41.             element.detachEvent('on' + type, handler);
42.         } else {
43.             element['on' + type] = null;
44.         }
45.     },
46.
47.     // 阻止事件 (主要是事件冒泡，因为 IE 不支持事件捕获)
48.     stopPropagation：function(ev) {
49.         if (ev.stopPropagation) {
50.             ev.stopPropagation();
51.         } else {
```

```
52.             ev.cancelBubble = true;
53.         }
54.     },
55.
56.     // 取消事件的默认行为
57.     preventDefault : function(event) {
58.         if (event.preventDefault) {
59.             event.preventDefault();
60.         } else {
61.             event.returnValue = false;
62.         }
63.     },
64.
65.     // 获取事件目标
66.     getTarget : function(event) {
67.         return event.target || event.srcElement;
68.     },
69.
70.     // 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；
71.     getEvent : function(e) {
72.         var ev = e || window.event;
73.         if (!ev) {
74.             var c = this.getEvent.caller;
75.             while (c) {
76.                 ev = c.arguments[0];
```

```
77.                 if (ev && Event == ev.constructor) {
78.                     break;
79.                 }
80.                 c = c.caller;
81.             }
82.         }
83.         return ev;
84.     }
85. };
```

•Node.js 的适用场景？

高并发、聊天、实时消息推送

•介绍 js 的基本数据类型。

number,string,boolean,object,undefined

•Javascript 如何实现继承？

通过原型和构造器

•["1", "2", "3"].map(parseInt) 答案是多少？

[1, NaN, NaN] 因为 parseInt 需要两个参数 (val, radix) 但 map 传了 3 个 (element, index, array)

•如何创建一个对象？（画出此对象的内存图）

```
1. function Person(name, age) {
2.     this.name = name;
3.     this.age = age;
4.     this.sing = function() { alert(this.name) }
5. }
```

•谈谈 This 对象的理解。

this 是 js 的一个关键字 ,随着函数使用场合不同 ,this 的值会发生变化。但是有一个总原则，那就是 this 指的是调用函数的那个对象。

this 一般情况下：是全局对象 Global。 作为方法调用，那么 this 就是指这个对象

•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、火狐是 事件捕获；

3. ev.stopPropagation();

•什么是闭包（closure），为什么要用它？

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.

```
1. function say667() {
2.     // Local variable that ends up within closure
3.     var num = 666;
4.     var sayAlert = function() { alert(num); }
5.     num++;
6.     return sayAlert;
7. }
8. var sayAlert = say667();
9. sayAlert()//执行结果应该弹出的 667
```

所谓“闭包”，就是在构造函数体内定义另外的函数作为目标对象的方法函数，而这个对象的方法函数反过来引用外层函数体中的临时变量。这使得只要目标对象在生存期内始终能保持其方法，就能间接保持原构造函数体当时用到的临时变量值。尽管最开始的构造函数调用已经结束，临时变量的名称也都消失了，但在目标对象的方法内却始终能引用到该变量的值，而且该值只能通这种方法来访问。即使再次调用相同的构造函数，但只会生成新对象和方法，新的临时变量只是对应新的值，和上次那次调用的是各自独立的。

•"use strict";是什么意思 ？ 使用它的好处和坏处分别是什么？

使用"use strict"指令的目的是说明（脚本或函数中）后续的代码将会解析为严格代码(strict code)。如果顶层（不存在任何函数内的）代码使用了"use strict"指令，那么它们就是严格代码。如果在函数体内定义，则函数整体使用的是严格代码。严格代码和非严格代码之间的区别如下：

1）在严格模式中禁止使用 with 语句;

2）在严格模式中，所有变量都要先声明，如果给一个未声明的变量、函数、函数参数，catch 从句参数或全局对象的属性赋值，将会 throw 一个引用错误（在非严格模式下，这种隐式声明的全局变量的方法是给全局对象添加一个新属性）

3）在严格模式中，调用的函数（不是方法）中的一个 this 值是 undefined。（在非严格模式下，调用函数中的 this 值总是全局对象）。可以利用这种特性来判断 js 实现是否支持严格模式。

4）同样，在严格模式中，当通过 call()或 apply()来调用函数时，其中的 this 值就是通过

call()或 apply()传入的第一个参数（在非严格模式下，null 和 undefined 值被全局对象和转换为对象的非对象值所代替）
5）在严格模式下，给只读属性赋值和给不可扩展的对象创建新成员都将抛出一个类型错误异常（在非严格模式下，这些操作只是简单的操作失败，不会抛错）
6）在严格模式下，传入 eval_r()的代码不能在调用程序所在的上下文中声明变量或定义函数，而在非严格模式中是可以这样的。相反，变量和函数定义是在 eval_r()创建的新作用域中，这个作用域在 eval_r()返回时就弃用了。
严格模式的一大目标是让你能更快更方便的调试.strict 模式会让你面临由于第三方代码没有为严格模式做好准备而引发的问题。 在严格模式中，标识符 eval 和 arguments 当作关键字，它们的值是不能更改的。严格模式中限制了对调用栈的检测能力，在严格模式的函数中，arguments.caller 和 arguments.callee 都会抛出一个类型错误异常。

•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);
```

•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

•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

{'age':'12', 'name':'back'}

•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?

•同步和异步的区别?

•如何解决跨域问题?

jsonp、 iframe、window.name、window.postMessage、服务器上设置代理页面

JSONP 的最基本的原理是：动态添加一个<script>标签，而 script 标签的 src 属性是没有跨域的限制的。这样说来，这种跨域方式其实与 ajax XmlHttpRequest 协议无关了。**创建一个回调函数，然后在远程服务上调用这个函数并且将 JSON 数据形式作为参数传递，完成回调**。也就是将你自己在客户端定义的回调函数的函数名传送给服务端，服务端则会返回以你定义的回调函数名的方法，将获取的 json 数据传入这个方法完成回调。

•模块化怎么做？

立即执行函数,不暴露私有成员

```
1.  var module1 = (function(){
2.          var _count = 0;
3.          var m1 = function(){
4.              //...
5.          };
6.
7.          var m2 = function(){
8.              //...
9.          };
10.
11.         return {
12.             m1 : m1,
13.             m2 : m2
14.         };
15.     })();
```

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

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 默认推荐的是
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()
...
})
```

虽然 AMD 也支持 CMD 的写法，同时还支持将 require 作为依赖项传递，但 RequireJS 的作者默认是最喜欢上面的写法，也是官方文档里默认的模块定义写法。

3. AMD 的 API 默认是**一个当多个用**，CMD 的 API 严格区分，推崇**职责单一**。比如 AMD 里，require 分全局 require 和局部 require，都叫 require。CMD 里，没有全局 require，而是根据模块系统的完备性，提供 seajs.use 来实现模块系统的加载启动。CMD 里，每个 API 都**简单纯粹**。

•异步加载的方式有哪些？

(1) defer，只支持 IE

(2) async：

(3) 创建 script，插入到 DOM 中，加载完毕后 callBack

http://www.cnblogs.com/tiwlin/archive/2011/12/26/2302554.html

•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

•.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。

```
1. function add(a,b)
2. {
3.     alert(a+b);
4. }
5. function sub(a,b)
6. {
7.     alert(a-b);
8. }
9. add.call(sub,3,1);
10.
```

•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？

```
 1 var jQuery = function() {
 3         // 返回选择器实例
 4         return new jQuery.prototype.init();
 5 };
 6 jQuery.prototype = {
 8         // 选择器构造函数
 9         init: function() {
11         },
13         // 原型方法
14         toArray: function() {
16         },
17         get: function() {
19         }
20 };
22 // 共享原型
23 jQuery.prototype.init.prototype = jQuery.prototype;
```

$(xx)或 Jquery(xx)得到不是真正的 jQuery 函数生成的对象，而是 jQuery.fn.init 函数生成的对象。也是就是 jQuery 的对象继承的是 jQuery.fn.init 的原型。 jQuery.fn = jQuery.prototype={..}。我们基本上不用 new jQuery(xx)，而是直接 jQuery(xx),先生成 jQuery 函数的对象，把原型中的继承下来，返回的也是 jQuery.fn.init 函数生成的对象。

•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：

1. $.fn.stringifyArray = function(array) {

```
2.      return JSON.stringify(array)
3.  }
4.  $.fn.parseArray = function(array) {
5.      return JSON.parse(array)
6.  }
```

然后调用：

```
1. $("").stringifyArray(array)
```

•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++) {}

•JavaScript 中的作用域与变量声明提升？

“一个变量的作用域表示这个变量存在的上下文。它指定了你可以访问哪些变量以及你是否有权限访问某个变量。” 变量作用域分为局部作用域和全局作用域。

javascript 没有块级作用域（被花括号包围的）；当是，javascript 有拥有函数级别的作用域，也就是说，在一个函数内定义的变量只能在函数内部访问或者这个函数内部的函数访问（闭包除外）。

•如何编写高性能的 Javascript？

•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？

•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。

其他问题

- 你遇到过比较难的技术问题是？你是如何解决的？
- 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？
- 列举 IE 与其他浏览器不一样的特性？
- 99%的网站都需要被重构是那本书上写的？
- 什么叫优雅降级和渐进增强？
- WEB 应用从服务器主动推送 Data 到客户端有那些方式？
-
- 你有哪些性能优化的方法？

  （看雅虎 14 条性能优化原则）。 （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。

- http 状态码有那些？分别代表是什么意思？

  100-199 用于指定客户端应相应的某些动作。 200-299 用于表示请求成功。 300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。 400-499 用于指出客户端的错误。400 0 1、语义有误，当前请求无法被服务器理解。401 当前请求需要用户验证 403 服务器已经理解请求，但是拒绝执行它。500-599 用于支持服务器错误。 503 – 服务不可用

- 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）

  查找浏览器缓存 DNS 解析、查找该域名对应的 IP 地址、重定向（301）、发出第二个 GET 请求 进行 HTTP 协议会话 客户端发送报头(请求报头) 服务器回馈报头(响应报头) html 文档开始下载 文档树建立，根据标记请求所需指定 MIME 类型的文件 文件显示 [浏览器这边做的工作大致分为以下几步： 加载：根据请求的 URL 进行域名解析，向服务器发起请求，接收文件（HTML、JS、CSS、图象等）。 解析：对加载到的资源（HTML、JS、CSS 等）进行语法解析，建议相应的内部数据结构（比如 HTML 的 DOM 树，JS 的（对象）属性表，CSS 的样式规则等等） }

- 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？
- 你常用的开发工具是什么，为什么？
- 对前端界面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。
- 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 命民以该 JS 功能为准英文翻译； 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？
- 说说最近最流行的一些东西吧？常去哪些网站？

  Node.js、Mongodb、npm、MVVM、MEAN、three.js
- 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、
- 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？
- 想问公司的问题？

  问公司问题： 目前关注哪些最新的 Web 前端技术（未来的发展方向）？ 前端团队如何工作的(实现一个产品的流程）？ 公司的薪资结构是什么样子的？