HTML 部分

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;
- 即使在没有样式 CSS 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。

你能描述一下当你制作一个网页的工作流程吗？

- 内容分析：分清展现在网络中内容的层次和逻辑关系
- 结构设计：写出合理的 html 结构代码
- 布局设计：使用 html+css 进行布局
- 样式设计：首先要使用 reset.css
- 交互设计：鼠标特效。行为设计：js 代码，ajax 页面行为和从服务器获取数据。最后测试兼容性。优化性能

你如何对网站的文件和资源进行优化？

- 期待的解决方案包括：
    - 文件合并
    - 文件最小化/文件压缩
    - 使用 CDN 托管
    - 缓存的使用
    - 其他

请说出三种减少页面加载时间的方法。（加载时间指感知的时间或者实际加载时间）

- 1). 尽量减少页面中重复的 HTTP 请求数量
- 2). 服务器开启 gzip 压缩
- 3). css 样式的定义放置在文件头部
- 4). Javascript 脚本放在文件末尾
- 5). 压缩 Javascript、CSS 代码
- 7). 尽可能减少 DCOM 元素
- 8). 使用多域名负载网页内的多个文件、图片
- 9). 使用 CDN
- 10). 在服务器端配置 control-cache last-modify-date
- 11). 在服务器配置 Entity-Tag if-none-match

23 条 Web 性能优化最佳实践和规则

- 1.尽可能减少 HTTP 请求次数
- 2.使用 CDN
- 3.避免使用 src 和 href 标签
- 4.加入 Expires 或 Cache-Control Header
- 5.使用 Gzip 压缩
- 6.在 html 文件顶部放置样式表
- 7.在 html 文件底部放置 Javascript 脚本

- 8.避免使用 CSS 表达式
- 9.使用外部 Javascript 和 CSS 外部文件
- 10.减少使用 DNS 查找次数
- 11.精简 Javascript 和 CSS
- 12.避免重定向
- 13.移除重复的脚本
- 14.配置 ETag
- 15.缓存 AJAX
- 16.使用 GET 完成 AJAX 请求
- 17.减少 DOM 元素数量
- 18.避免 404
- 19.减少 Cookie 大小
- 20.使用无 Cookie 的域
- 21.避免使用滤镜
- 22.不要在 HTML 中缩放图片
- 23.使用小 favicon.ico 文件，并让其可缓存

如何进行网站性能优化

-

content 方面

-

i. 减少 HTTP 请求：合并文件、CSS 精灵、inline Image

ii. 减少 DNS 查询：DNS 查询完成之前浏览器不能从这个主机下载任何任何文件。方法：DNS 缓存、将资源分布到恰当数量的主机名，平衡并行下载和 DNS 查询

iii. 避免重定向：多余的中间访问

iv. 使 Ajax 可缓存

v. 非必须组件延迟加载

vi. 未来所需组件预加载

vii. 减少 DOM 元素数量

viii. 将资源放到不同的域下：浏览器同时从一个域下载资源的数目有限，增加域可以提高并行下载量

ix. 减少 iframe 数量

x. 不要 404

-

Server 方面

-

i. 使用 CDN

ii. 添加 Expires 或者 Cache-Control 响应头

iii. 对组件使用 Gzip 压缩

iv. 配置 ETag

v. Flush Buffer Early

vi. Ajax 使用 GET 进行请求

vii. 避免空 src 的 img 标签

- Cookie 方面

i. 减小 cookie 大小

ii. 引入资源的域名不要包含 cookie

- css 方面

i. 将样式表放到页面顶部

ii. 不使用 CSS 表达式

iii. 使用不使用@import

iv. 不使用 IE 的 Filter

- Javascript 方面

i. 将脚本放到页面底部

ii. 将 javascript 和 css 从外部引入

iii. 压缩 javascript 和 css

iv. 删除不需要的脚本

v. 减少 DOM 访问

vi. 合理设计事件监听器

- 图片方面

i. 优化图片：根据实际颜色需要选择色深、压缩

ii. 优化 css 精灵

iii. 不要在 HTML 中拉伸图片

iv. 保证 favicon.ico 小并且可缓存

- 移动方面

i. 保证组件小于 25k

ii. Pack Components into a Multipart Document

在制作一个 Web 应用或 Web 站点的过程中，你是如何考虑他的 UI、安全性、高性能、SEO、可维护性以及技术因素的？

- UI：界面美观，要有个性，考虑用户使用的逻辑要简单，用起来舒适自由。使用习惯要符合大部分用户的习惯，比如少让用户输入，采用选择的方式，提供搜索和提示功能。
- 安全性：对输入进行有效性验证（非法字符，特殊字符）如 PHP 中的方法 htmlspecialchars()将特殊字符（>）转化为 html 实体，trim()去掉用户输入的不必要字符，stripslashes()去掉用户输入的反斜杠等等。
- 对交互操作进行身份验证和授权（api-key,authtoken）,异常错误处理（向用户反馈单额错误提示不要让攻击者分析出一些网络环境和配置），内存溢出，注入攻击等。
- 高性能：
    - DNS 负载均衡
    - HTTP 重定向（通过使客户端重定向，来分散和转移请求压力，比如一些下载服务通常都有几个镜像服务器）

- 分布式缓存
- 负载均衡：反向代理负载均衡，
- 数据库扩展：读写分离，垂直分区，水平分区。
- SEO:选好关键字，描述语言，修饰性图片换成文本，合理使用 h1-h6，对图片添加 alt 属性，链接添加 target 属性。
- 可维护性：代码是否容易被理解，是否容易被修改和增加新的功能，当出现问题时是否能快速定位到问题代码。

请尽可能完整得描述下从输入 URL 到整个网页加载完毕及显示在屏幕上的整个流程

- 1）把 URL 分割成几个部分：协议、网络地址、资源路径。其中网络地址指示该连接网络上哪一台计算机，可以是域名或者 IP 地址，可以包括端口号；协议是从该计算机获取资源的方式，常见的是 HTTP、FTP，不同协议有不同的通讯内容格式；资源路径指示从服务器上获取哪一项资源。 例如：http://www.guokr.com/question/554991/ 协议部分：http 网络地址：www.guokr.com 资源路径：/question/554991/
- 2）如果地址不是一个 IP 地址，通过 DNS（域名系统）将该地址解析成 IP 地址。IP 地址对应着网络上一台计算机，DNS 服务器本身也有 IP，你的网络设置包含 DNS 服务器的 IP。 例如：www.guokr.com 不是一个 IP，向 DNS 询问请求 www.guokr.com 对应的 IP，获得 IP： 111.13.57.142。这个过程里，你的电脑直接询问的 DNS 服务器可能没有 www.guokr.com 对应的 IP，就会向它的上级服务器询问，上级服务器同样可能没有，就依此一层层向上找，最高可达根节点，找到或者全部找不到为止。

- 3）如果地址不包含端口号，根据协议的默认端口号确定一个。端口号之于计算机就像窗口号之于银行，一家银行有多个窗口，每个窗口都有个号码，不同窗口可以负责不同的服务。端口只是一个逻辑概念，和计算机硬件没有关系。 例如：www.guokr.com 不包含端口号，http 协议默认端口号是 80。如果你输入的 url 是 http://www.guokr.com:8080/ ，那表示不使用默认的端口号，而使用指定的端口号 8080。
- 4）向 2 和 3 确定的 IP 和端口号发起网络连接。 例如：向 111.13.57.142 的 80 端口发起连接
- 5）根据 http 协议要求，组织一个请求的数据包，里面包含大量请求信息，包括请求的资源路径、你的身份 例如：用自然语言来表达这个数据包，大概就是：请求 /question/554991/ ，我的身份是 xxxxxxx。
- 6）服务器响应请求，将数据返回给浏览器。数据可能是根据 HTML 协议组织的网页，里面包含页面的布局、文字。数据也可能是图片、脚本程序等。现在你可以用浏览器的“查看源代码”功能，感受一下服务器返回的是什么东东。如果资源路径指示的资源不存在，服务器就会返回著名的 404 错误。
- 7）如果（6）返回的是一个页面，根据页面里一些外链的 URL，例如图片的地址，按照（1）—（6）再次获取。
- 8）开始根据资源的类型，将资源组织成屏幕上显示的图像，这个过程叫渲染，网页渲染是浏览器最复杂、最核心的功能。
- 9）将渲染好的页面图像显示出来，并开始响应用户的操作。 以上只是最基本的步骤，实际不可能就这么简单，一些可选的步骤例如网页缓存、连接池、加载策略、加密解密、代理中转等等都没有提及。即使基本步骤本身也有很复杂的子步骤，TCP/IP、DNS、HTTP、HTML：每一个都可以展开成庞大的课题，而浏览器的基础——操作系统、编译器、硬件等更是一个比一个复杂。不是计算机专业的同学看了上面的解释完全不明白是很正常的，可能会问为什么要搞

得那么复杂，但我保证这每一个步骤都经过深思熟虑和时间的考验。你输入 URL 即可浏览互联网，而计算机系统在背后做了无数你看不到的工作，计算机各个子领域无数工程师为此付出你难以想象的努力。

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、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

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

1，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）。
2，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。
3，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。

有效时间：

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 有哪些缺点？

1，iframe 会阻塞主页面的 Onload 事件；
2，搜索引擎的检索程序无法解读这种页面，不利于 SEO；
3，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，这样可以可以绕开以上两个问题。

如何实现浏览器内多个标签页之间的通信？(阿里)

WebSocket、SharedWorker；
也可以调用 localstorge、cookies 等本地存储方式；

localstorge 另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，
我们通过监听事件，控制它的值来进行页面信息通信；
注意 quirks：Safari 在无痕模式下设置 localstorge 值时会抛出 QuotaExceededError 的异常；

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

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]。bug 多，对 w3c 标准的支持不是很好。

Gecko 内核：Netscape6 及以上版本，FF,MozillaSuite/SeaMonkey 等。可以支持很多复杂网页效果，但是能耗高，占内存。

Presto 内核：Opera7 及以上。 [Opera 内核原为：Presto，现为：Blink;]。公认的浏览网页速度最快的内核，处理 js 时比其他内核快 3 倍左右。但是网页兼容性不太好。

Webkit 内核：Safari,Chrome 等。 [Chrome 的：Blink（WebKit 的分支）]。速度仅次于 presto，兼容性较好。

HTTP request 报文结构是怎样的

rfc2616 中进行了定义：
首行是 Request-Line 包括：请求方法，请求 URI，协议版本，CRLF
首行之后是若干行请求头，包括 general-header，request-header 或者 entity-header，每个一行以 CRLF 结束
请求头和消息实体之间有一个 CRLF 分隔
根据实际请求需要可能包含一个消息实体 一个请求报文例子如下：

```
GET /Protocols/rfc2616/rfc2616-sec5.html HTTP/1.1
Host: www.w3.org
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/35.0.1916.153 Safari/537.36
Referer: https://www.google.com.hk/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: zh-CN,zh;q=0.8,en;q=0.6
Cookie: authorstyle=yes
```

If-None-Match: "2cc8-3e3073913b100"
If-Modified-Since: Wed, 01 Sep 2004 13:24:52 GMT

name=qiu&age=25

HTTP response 报文结构是怎样的

rfc2616 中进行了定义：
首行是状态行包括：HTTP 版本，状态码，状态描述，后面跟一个 CRLF
首行之后是若干行响应头，包括：通用头部，响应头部，实体头部
响应头部和响应实体之间用一个 CRLF 空行分隔
最后是一个可能的消息实体 响应报文例子如下：
HTTP/1.1 200 OK
Date: Tue, 08 Jul 2014 05:28:43 GMT
Server: Apache/2
Last-Modified: Wed, 01 Sep 2004 13:24:52 GMT
ETag: "40d7-3e3073913b100"
Accept-Ranges: bytes
Content-Length: 16599
Cache-Control: max-age=21600
Expires: Tue, 08 Jul 2014 11:28:43 GMT
P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Content-Type: text/html; charset=iso-8859-1

{"name": "qiu", "age": 25}

HTTP 状态码

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）。

Doctype作用？严格模式与混杂模式如何区分？它们有何意义？有多少种Doctype文档类型？

1，<!DOCTYPE> 声明位于文档中的最前面，处于 标签之前。告知浏览器以何种模式来渲染文档。
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。
5，可声明三种文档类型，分别表示严格版本、过渡版本以及基于框架的类型。
HTML 4.01 规定了三种文档类型：Strict、Transitional 以及 Frameset。
XHTML 1.0 规定了三种 XML 文档类型：Strict、Transitional 以及 Frameset。
Standards （标准）模式（也就是严格呈现模式）用于呈现遵循最新标准的网页，
Quirks（包容）模式（也就是松散呈现模式或者兼容模式）用于呈现为传统浏览器而设计的网页。

前端需要注意哪些SEO

- 合理的 title、description、keywords：搜索对着三项的权重逐个减小，title 值强调重点即可，重要关键词出现不要超过 2 次，而且要靠前，不同页面 title 要有所不同；description 把页面内容高度概括，长度合适，不可过分堆砌关键词，不同页面 description 有所不同；keywords 列举出重要关键词即可
- 语义化的 HTML 代码，符合 W3C 规范：语义化代码让搜索引擎容易理解网页
- 重要内容 HTML 代码放在最前：搜索引擎抓取 HTML 顺序是从上到下，有的搜索引擎对抓取长度有限制，保证重要内容一定会被抓取
- 重要内容不要用 js 输出：爬虫不会执行 js 获取内容

- 少用 iframe：搜索引擎不会抓取 iframe 中的内容
- 非装饰性图片必须加 alt
- 提高网站速度：网站速度是搜索引擎排序的一个重要指标

HTTP 请求方式

- 一台服务器要与 HTTP1.1 兼容，只要为资源实现 GET 和 HEAD 方法即可
- GET 是最常用的方法，通常用于请求服务器发送某个资源。
- HEAD 与 GET 类似，但服务器在响应中值返回首部，不返回实体的主体部分
- PUT 让服务器用请求的主体部分来创建一个由所请求的 URL 命名的新文档，或者，如果那个 URL 已经存在的话，就用干这个主体替代它
- POST 起初是用来向服务器输入数据的。实际上，通常会用它来支持 HTML 的表单。表单中填好的数据通常会被送给服务器，然后由服务器将其发送到要去的地方。
- TRACE 会在目的服务器端发起一个环回诊断，最后一站的服务器会弹回一个 TRACE 响应并在响应主体中携带它收到的原始请求报文。TRACE 方法主要用于诊断，用于验证请求是否如愿穿过了请求/响应链。
- OPTIONS 方法请求 web 服务器告知其支持的各种功能。可以查询服务器支持哪些方法或者对某些特殊资源支持哪些方法。
- DELETE 请求服务器删除请求 URL 指定的资源

请描述一下 GET 和 POST 的区别?

1、 get 是从服务器获取数据 -----"取"; post 是向服务器提交数据 -----“发”

2、 form 表单默认的 method 为"GET"

3、 get 将数据按照 variable = value 的形式，加上 URL 的后面，中间用"?"连接，各个变量之间用"&"连接; post 将数据不像 get 方式那样

4、 参数上面 3 的数据传输方式，可以得出：post 安全性比 get 方式要高

5、

URL 不存在参数上限的问题，HTTP 协议没有对 URL 长度进行限制，限制的是部分浏览器和服务器的限制。

IE 对 URL 长度的限制为 2083KB

get 方式是通过 URL 传输的数据的，数据量一般在 2KB 左右，但是执行效率比 post 高

理论上 post 方式没有大小限制，HTTP 协议规范也没进行大小限制。post 数据没有限制，限制的是服务器处理程序的能力

浏览器本地存储与服务器端存储之间的区别

- 其实数据既可以在浏览器本地存储，也可以在服务器端存储。
- 浏览器端可以保存一些数据，需要的时候直接从本地获取，sessionStorage、localStorage 和 cookie 都由浏览器存储在本地的数据。
- 服务器端也可以保存所有用户的所有数据，但需要的时候浏览器要向服务器请求数据。
- 1.服务器端可以保存用户的持久数据，如数据库和云存储将用户的大量数据保存在服务器端。
- 2.服务器端也可以保存用户的临时会话数据。服务器端的 session 机制，如 jsp 的 session 对象，数据保存在服务器上。实现上，服务器和浏览器之间仅需传递 session id 即可，服务器根据 session id 找到对应用户的 session 对象。会话数据仅在一段时间内有效，这个时间就是 server 端设置的 session 有效期。
- 服务器端保存所有的用户的数据，所以服务器端的开销较大，而浏览器端保存则把不同用户需要的数据分布保存在用户各自的浏览器中。
- 浏览器端一般只用来存储小数据，而服务器可以存储大数据或小数据。
- 服务器存储数据安全一些，浏览器只适合存储一般数据。

socket 和 http 有什么区别？

socket 是网络传输层的一种技术，跟 http 有本质的区别，http 是应用层的一个网络协议。使用 socket 技术理论上来讲，按照 http 的规范，完全可以使用 socket 来达到发送 http 请求的目的，只要发送的数据包按照 http 协议来即可

Socket 和 http 的区别：

Socket 是长连接，http 是短连接

Socket 是双向通信，http 是单向的，只能客户端向服务器发送数据

Socket 的数据完全由自己组织，http 必须按照 http 协议来发送

Socket 的使用场景:

1.客户端频繁请求服务器，如股票应用，需要一直向服务器请求最新的数据，如果使用 http，那么第一，频繁请求，就会频繁连接，造成服务器压力巨大，如果使用 socket，一次连接，不会消耗服务器太多资源。第二，频繁发送，返回数据，如果使用 http，因为 http 协议的限制，发送的数据包中包含了很多的请求头，请求行等 http 协议必须带的数据，发送的数据量比 socket 大很多，socket 只需要请求和返回需要的数据即可，如股票应用中，只需要返回股票的最新价格即可，及时性更高

2.客户端和服务器互相发送数据，如聊天应用，需要客户端上传聊天内容，同时，别人给你发消息，服务器也能主动把别人发送的消息发送给你

需要使用 socket 技术的场景：网络游戏、即时通信、股票软件、自己实现推送机制.

CSS 部分

CSS3 有哪些新特性（包含哪些模块）？

1、圆角 border-radius、阴影 box-shadow,text-shadow、渐变 gradients、过渡 transitions、
动画 animations、布局 multi-columns, flex box, grid layout, Opacity,color(rgb,rgba,hsl,hsla)
2、子串匹配的属性选择器:E[attribute^="value"], E[attribute$="value"], E[attribute*="value"]
3、新的伪类：
:target, :enabled 和 :disabled,:checked,:indeterminate,:root,:nth-child 和 :nth-last-child,
:nth-of-type 和 :nth-last-of-type,:last-child,:first-of-type 和 :last-of-type,
:only-child 和 :only-of-type,:empty,和 :not
4、伪元素使用两个冒号而不是一个来表示：
:after 变为 ::after, :before 变为 ::before, :first-letter 变为 ::first-letter,
还有 :first-line 变为 ::first-line。

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：

```
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; }
```

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：
  1、关于文字排版的属性如：font,word-break,letter-spacing,text-align,text-rendering, word-spacing,white-space,text-indent,text-transform,text-shadow
  2、line-height
  3、color
  4、visibility
  5、cursor

* 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;
  !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。

解释下浮动和它的工作原理

浮动元素脱离文档流，不占据空间。浮动元素碰到包含它的边框或者浮动元素的边框停留

列举不同的清除浮动的技巧，并指出它们各自适用的使用场景

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。
该方法只适用于非 IE 浏览器。具体写法可参照以下示例。使用中需注意以下几点。一、该方法中必须为需要清除浮动元素的伪对象中设置 height:0，否则该元素会比实际高出若干像素；二、content 属性是必须的，但其值可以为空，蓝色理想讨论该方法的时候 content 属性的值 设为"."，但我发现为空亦是可以的。

介绍一下标准的 *CSS* 的盒子模型？与 *IE* 的盒子模型有什么不同的？

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). 文档中的每个元素被描绘为矩形盒子。确定其大小,属性——比如颜色、背景、边框,及其位置是渲染引擎的目标。CSS 下这些矩形盒子由标准盒模型描述。这个模型描述元素内容占用空间。
盒子有四个边界：外边距边界 margin edge, 边框边界 border edge, 内边距边界 padding edge 与 内容边界 content edge。

a.内容区域 content area 是真正包含元素内容的区域。位于内容边界的内部,
它的大小为内容宽度 或 content-box 宽及内容高度或 content-box 高。
如果 box-sizing 为默认值,
width, min-width, max-width, height, min-height 与 max-height 控制内容大小。

b.内边距区域 padding area 用内容及可能的边框之间的空白区域扩展内容区域。
它位于内边边界内部,通常有背景——颜色或图片（不透明图片盖住背景颜色）.
它的大小为 padding-box 宽与 padding-box 高。
内边距与内容边界之间的空间由 padding-top,padding-right,padding-bottom,padding-left 和简写 padding 控制。

c.边框区域 border area 是包含边框的区域,扩展了内边距区域.

它位于边框边界内部,大小为 border-box 宽和 border-box 高。

由 border-width 及简写属性 border 控制。

d.外边距区域 margin area 用空白区域扩展边框区域,以分开相邻的元素。

它的大小为 margin-box,margin-top,margin-right,margin-bottom,margin-left 及简写属性 margin 控制。

在外边距合并的情况下,由于盒之间共享外边距,外边距不容易弄清楚。

最后注意,对于行内非替换元素,其占用空间（行高）由 line-height 决定,即使有内边距与边框。

display 有哪些值？说明他们的作用。position 的值 relative 和 absolute 定位原点是？absolute 与 fixed 共同点与不同点？

1、display

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。

2、position

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

inherit 规定从父元素继承 position 属性的值。

*（忽略 top, bottom, left, right z-index 声明）

absolute 与 fixed 共同点与不同点

A、共同点：

1.改变行内元素的呈现方式,display 被置为 block;

2.让元素脱离普通流,不占据空间;

3.默认会覆盖到非定位元素上;

B、不同点：

absolute 的”根元素“是可以设置的,而 fixed 的”根元素“固定为浏览器窗口。

当你滚动网页,fixed 元素与浏览器窗口之间的距离是不变的。

display:none 和 visibility:hidden 的区别？

display:none 隐藏对应的元素,在文档布局中不再给它分配空间,它各边的元素会合拢,就当他从来不存在。
visibility:hidden 隐藏对应的元素,但是在文档布局中仍保留原来的空间。

解释下 CSS sprites,以及你要如何在页面或网站中使用它?

CSS Sprites 其实就是把网页中一些背景图片整合到一张图片文件中;
再利用 CSS 的“background-image”,“background- repeat,background-position”的组合进行背景定位;
background-position 可以用数字能精确的定位出背景图片的位置;

这样可以减少很多图片请求的开销,因为请求耗时比较长;请求虽然可以并发,但是也有限制,一般浏览器都是 6 个;
对于未来而言,就不需要这样做了,因为有了`http2`;

*优点:
减少 HTTP 请求数,极大地提高页面加载速度;
增加图片信息重复度,提高压缩比,减少图片大小;
更换风格方便,只需在一张或几张图片上修改颜色或样式即可实现;

*缺点:
图片合并麻烦;
维护麻烦,修改一个图片可能需要从新布局整个图片,样式;

什么叫外边距折叠(collapsing margins)?

毗邻的两个或多个 margin 会合并成一个 margin,叫做外边距折叠。规则如下:

1、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
2、浮动元素/inline-block 元素/绝对定位元素的 margin 不会和垂直方向上的其他元素的 margin 折叠
3、创建了块级格式化上下文的元素,不会和它的子元素发生 margin 折叠
4、元素自身的 margin-bottom 和 margin-top 相邻时也会折叠

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

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>

JS 部分

javascript 有哪几种数据类型

六种基本数据类型

- undefined
- null
- string
- boolean
- Number
- Object

说说 js 中的闭包和变量作用域？

一、变量的作用域

要理解闭包,首先必须理解 Javascript 特殊的变量作用域。

变量的作用域无非就是两种：全局变量和局部变量。

Javascript 语言的特殊之处,就在于函数内部可以直接读取全局变量。

```
var n=999;
function f1(){
    alert(n);
}
f1(); // 999
```

另一方面,在函数外部自然无法读取函数内的局部变量。

```
function f1(){
    var n=999;
}
alert(n); // error
```

需要注意:函数内部声明变量的时候,一定要使用 var 命令。

如果不用的话,你实际上声明了一个全局变量！

```
function f1(){
    n=999;
}
f1();
alert(n); // 999
```

----------分割线----------

二、如何从外部读取局部变量？

出于种种原因,我们有时候需要得到函数内的局部变量。

但是正常情况下,这是办不到的；只有通过变通方法才能实现。

那就是在函数的内部,再定义一个函数。

```
function f1(){
    var n=999;
    function f2(){
        alert(n); // 999
    }
}
```

在上面的代码中,函数 f2 就被包括在函数 f1 内部;

这时 f1 内部的所有局部变量,对 f2 都是可见的;

但是反过来就不行,f2 内部的局部变量,对 f1 就是不可见的;

这就是 Javascript 语言特有的"链式作用域"结构（chain scope）;

子对象会一级一级地向上寻找所有父对象的变量。

所以,父对象的所有变量,对子对象都是可见的,反之则不成立。

既然 f2 可以读取 f1 中的局部变量,那么只要把 f2 作为返回值

,我们不就可以在 f1 外部读取它的内部变量了吗！

```
function f1(){
    var n=999;
    function f2(){
        alert(n);
    }
    return f2;
}
var result=f1();
result(); // 999
```

----------分割线----------

三、闭包的概念

上一节代码中的 f2 函数,就是闭包。

各种专业文献上的"闭包"（closure）定义非常抽象,很难看懂。

我的理解是,闭包就是能够读取其他函数内部变量的函数。

由于在 Javascript 语言中,只有函数内部的子函数才能读取局部变量,

因此可以把闭包简单理解成"定义在一个函数内部的函数"。

所以,在本质上,闭包就是将函数内部和函数外部连接起来的一座桥梁。

----------分割线----------

四、闭包的用途

闭包可以用在许多地方。它的最大用处有两个:

一个是前面提到的可以读取函数内部的变量;

另一个就是让这些变量的值始终保持在内存中。

怎么来理解这句话呢？请看下面的代码。

```
function f1(){
    var n=999;
    nAdd=function(){n+=1}
    function f2(){
        alert(n);
    }
    return f2;
}
var result=f1();
result(); // 999
nAdd();
result(); // 1000
```

在这段代码中,result 实际上就是闭包 f2 函数。

它一共运行了两次,第一次的值是 999,第二次的值是 1000。

这证明了,函数 f1 中的局部变量 n 一直保存在内存中,并没有在 f1 调用后被自动清除。

为什么会这样呢？原因就在于 f1 是 f2 的父函数,而 f2 被赋给了一个全局变量,

这导致 f2 始终在内存中,而 f2 的存在依赖于 f1,因此 f1 也始终在内存中,

不会在调用结束后,被垃圾回收机制（garbage collection）回收。

这段代码中另一个值得注意的地方,就是"nAdd=function(){n+=1}"这一行,

1、首先在 nAdd 前面没有使用 var 关键字,因此 nAdd 是一个全局变量,而不是局部变量;

2、其次,nAdd 的值是一个匿名函数（anonymous function）,而这个匿名函数本身也是一个闭包,

所以 nAdd 相当于是一个 setter,可以在函数外部对函数内部的局部变量进行操作。

----------分割线----------

五、使用闭包的注意点

1、由于闭包会使得函数中的变量都被保存在内存中,内存消耗很大,所以不能滥用闭包,否则会造成网页的性能问题,在 IE 中可能导致内存泄露。解决方法是,在退出函数之前,将不使用的局部变量全部删除。

2、闭包会在父函数外部,改变父函数内部变量的值。

所以,如果你把父函数当作对象（object）使用,把闭包当作它的公用方法（Public Method）,把内部变量当作它的私有属性（private value）,这时一定要小心,不要随便改变父函数内部变量的值。

----------分割线----------

六、思考题

如果你能理解下面两段代码的运行结果,应该就算理解闭包的运行机制了。（首先考虑作用域 this 的指向）

代码片段一。

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        return function(){
            return this.name;
        };
    }
};
alert(object.getNameFunc()());
```

代码片段二。

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        var that = this;
        return function(){
            return that.name;
        };
    }
};
alert(object.getNameFunc()());
```

说说 *js* 中的事件机制？

一、事件机制(事件流)

浏览器中的事件流意味着页面上可有不仅一个,甚至多个元素响应同一个事件。

而这一个或多个元素响应事件发生的先后顺序在各个浏览器（主要针对 IE 和 Netscape）上是不同的。

冒泡型事件（Dubbed Bubbling）

IE 上的解决方案就是冒泡型事件（Dubbed Bubbling）,冒泡型事件的基本思想是,

事件按照从最特定的事件目标到最不特定的事件目标（document 对象）的顺序触发。

捕获型事件（Event Capturing）

相对 IE4.0,Netscape4.0 则使用的是捕获型事件的解决方案。

这个事件触发的过程则正好和冒泡相反——在捕获型事件中,事件从最不精确的对象（document 对象）开始触发,

然后到最精确的对象。

DOM 事件流

这个事件流则是 W3C 制定一个标准规范,它同时支持两种事件流模式,

不过是先发生捕获型事件流,再发生冒泡型事件流。

DOM 事件流最独特的是,它支持文本节点也触发事件（IE 中这不支持）,

不过说实话,我现在还看不出来让文本节点支持事件有什么作用。

最后要说的是,根据最近大家在开发的实践过程中的运用,

我们一般都采取冒泡型的事件流触发方式,这点我们的 IE 做的比较成功。

至于原因,我想你可以通过上面的解释可以看出,毕竟我们给元素触发事件,

肯定是希望从我们最希望先触发（从最精确的）的那个开始。

----------分割线----------

二、事件绑定

事件处理函数/监听函数

1、在 DOM 元素中直接绑定

2、在 JavaScript 代码中绑定

3、绑定事件监听函数

具体实践：

IE 中 attachEvent(”NAME_OF_EVENT_HANDLER”, fnHandler)给元素绑定事件；

而在支持 DOM 事件流的浏览器里，则使用 addEventListener(”NAME_OF_EVENT_HANDLER”, fnHandler, isCapture);

前面我控制 FIREFOX 中触发捕获事件流,就是通过设置 isCapture（ture：捕获；false：冒泡）做到的;

```
function addEvent(obj,type,handle){
    try{ // Chrome、FireFox、Opera、Safari、IE9.0 及其以上版本
```

```
            obj.addEventListener(type,handle,false);
        }catch(e){
            try{ // IE8.0 及其以下版本
                obj.attachEvent('on' + type,handle);
            }catch(e){ //  现代浏览器
                obj['on' + type] = handle;
            }
        }
    }
```

哪些事件是支持冒泡,那些不支持？

基本上只有 onload,unload,focus,blur,submit 和 change 事件是不支持冒泡的；

自然像 keydownkeypress,keyup,click,dbclick,mousedown,mouseout,mouseover,mouseup,mousemove 支持冒泡,

那就是“Mouse and Key Events”支持冒泡;

----------分割线----------

三、事件委托

document.onclick,这个示例把事件委托放到了 document 上,即点击 document 就直接触发我们相应的事件。

```
    document.onclick = function(event){
        var event = event || window.event;              //IE doesn't pass in the event object
        var target = event.target || event.srcElement; //IE uses srcElement as the target
        switch(target.id){
            case "help-btn":
                openHelp();
            break;

            case "save-btn":
                saveDocument();
            break;

            case "undo-btn":
                undoChanges();
            break;
            //如果有其元素需要处理点击事件，只需要在这里添加不同的 case 分支就行。
        }
    };
```

优点：

从“处理速度”、“新增元素事件处理”和“内存消耗”三方面比较了“事件委托”和“事件绑定”的对比,

可以很容易看出,“事件委托”在“处理速度”和“内存消耗”上,有得天独厚的优势。

所以,在 Web 编程的时候,尤其在构建大型系统的时候,应该尽量考虑使用“事件委托”。
但是,“事件委托”并不是万能的；它也有一些弊端。下面我们在论述一下它的弊端。

缺点：
使用“事件委托”时,并不是说把事件委托给的元素越靠近顶层就越好。
事件冒泡的过程也需要耗时,越靠近顶层,事件的”事件传播链”越长,也就越耗时。
----------分割线----------
四、阻止事件冒泡和阻止事件默认行为
a.阻止事件冒泡

```
function stopBubble(e) {
    //如果提供了事件对象,则这是一个非 IE 浏览器
    if ( e && e.stopPropagation ){
        e.stopPropagation();            //因此它支持 W3C 的 stopPropagation()方法
    }else{
        window.event.cancelBubble = true;          //否则,我们需要使用 IE 的方式来取消事件冒泡
    }
}
```

b.当按键后,不希望按键继续传递给如 HTML 文本框对象时,可以取消返回值.即停止事件默认行为;

```
//阻止浏览器的默认行为
function stopDefault(e) {
    //如果提供了事件对象,则这是一个非 IE 浏览器
    if (e && e.preventDefault ){
        e.preventDefault();       //阻止默认浏览器动作(W3C)
    }else{
        window.event.returnValue = false;     //IE 中阻止函数器默认动作的方式
    }
}
```

说说 *http* 请求和 *ajax* 以及 *ajax* 跨域？

1、HTTP 请求
超文本传输协议（HTTP）的设计目的是保证客户机与服务器之间的通信。
HTTP 的工作方式是客户机与服务器之间的请求-应答协议。
两种 HTTP 请求方法：GET 和 POST:
a.GET 方法
请注意,查询字符串（名称/值对）是在 GET 请求的 URL 中发送的：
/test/demo_form.asp？name1=value1&name2=value2
有关 GET 请求的其他一些注释：
GET 请求可被缓存
GET 请求保留在浏览器历史记录中
GET 请求可被收藏为书签

GET 请求不应在处理敏感数据时使用

GET 请求有长度限制

GET 请求只应当用于取回数据

---

b.POST 方法

请注意,查询字符串（名称/值对）是在 POST 请求的 HTTP 消息主体中发送的：

```
POST /test/demo_form.asp HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

有关 POST 请求的其他一些注释：

POST 请求不会被缓存

POST 请求不会保留在浏览器历史记录中

POST 不能被收藏为书签

POST 请求对数据长度没有要求

---

c、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 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

----------分割线----------

2、AJAX

AJAX (异步 JavaScript 和 XML) 是个新产生的术语,专为描述 JavaScript 的两项强大性能.

这两项性能在多年来一直被网络开发者所忽略,

直到最近 Gmail, Google Suggest 和 Google Maps 的横空出世才使人们开始意识到其重要性.

这两项被忽视的性能是:

无需重新装载整个页面便能向服务器发送请求.

对 XML 文档的解析和处理

1、步骤 1 浏览器发送一个 HTTP 请求

为了用 JavaScript 向服务器发送一个 HTTP 请求, 需要一个具备这种功能的类实例.

这样的类首先由 Internet Explorer 以 ActiveX 对象引入, 被称为 XMLHTTP.

后来 Mozilla, Safari 和其他浏览器纷纷仿效, 提供了 XMLHttpRequest 类,它支持微软的 ActiveX 对象所提供的方法和属性.

A、创建一个跨浏览器的这样的类实例(对象), 可以应用如下代码:

```
if (window.XMLHttpRequest) {
```

```
        http_request = new XMLHttpRequest();            // Mozilla, Safari, ...
    }
    else if (window.ActiveXObject)
    {
        http_request = new ActiveXObject("Microsoft.XMLHTTP");          // IE
    }
```

(上例对代码做了一定简化,这是为了解释如何创建 XMLHTTP 类实例. 实际的代码实例可参阅本篇步骤 3.)

如果服务器的响应没有 XML mime-type header,某些 Mozilla 浏览器可能无法正常工作.

为了解决这个问题, 如果服务器响应的 header 不是 text/xml,可以调用其它方法修改该 header.

```
http_request = new XMLHttpRequest();
http_request.overrideMimeType('text/xml');
```

B、决定当收到服务器的响应后,需要做什么.这需要告诉 HTTP 请求对象用哪一个 JavaScript 函数处理这个响应.

可以将对象的 onreadystatechange 属性设置为要使用的 JavaScript 的函数名,如下所示:

```
http_request.onreadystatechange = nameOfTheFunction;
```

注意:在函数名后没有括号,也无需传递参数.另外还有一种方法,

你可以使用一个匿名函数来描述那些要对服务器返回的响应内容所进行的操作,如下所示:

```
http_request.onreadystatechange = function(){
  // do the thing
};
```

C、在定义了如何处理响应后,就要发送请求了.可以调用 HTTP 请求类的 open()和 send()方法, 如下所示:

```
http_request.open('GET', 'http://www.example.org/some.file', true);
http_request.send(null);
```

1、open()的第一个参数是 HTTP 请求方式 – GET, POST, HEAD 或任何服务器所支持的您想调用的方式;

按照 HTTP 规范,该参数要大写;否则,某些浏览器(如 Firefox)可能无法处理请求;

有关 HTTP 请求方法的详细信息可参考 http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

2、第二个参数是请求页面的 URL.由于自身安全特性的限制,该页面不能为第三方域名的页面.

同时一定要保证在所有的页面中都使用准确的域名,否则调用 open()会得到"permission denied"的错误提示.

一个常见的错误是访问站点时使用 domain.tld,而当请求页面时,却使用

www.domain.tld.

3、第三个参数设置请求是否为异步模式.如果是 TRUE, JavaScript 函数将继续执行,而不等待服务器响应.

这就是"AJAX"中的"A"：

1、如果第一个参数是"GET",通过 open()第二个参数,传递参数,一般用于查询,jsonp 的跨域就是这个原理,也只能解决 get 方式的请求；

2、如果第一个参数是"POST",send()方法的参数可以是任何想送给服务器的数据. 这时数据要以字符串的形式送给服务器,如下所示:

name=value&anothername=othervalue&so=on

2、步骤 2 服务器收到请求后的响应

当发送请求时,要提供指定处理响应的 JavaScript 函数名.

```
http_request.onreadystatechange = nameOfTheFunction;
```

我们来看看这个函数的功能是什么.首先函数会检查请求的状态.

如果状态值是 4,就意味着一个完整的服务器响应已经收到了,您将可以处理该响应.

```
if (http_request.readyState == 4) {
    // everything is good, the response is received
} else {
    // still not ready
}
```

readyState 的取值如下:

0 (未初始化)
1 (正在装载)
2 (装载完毕)
3 (交互中)
4 (完成)
(Source)

接着,函数会检查 HTTP 服务器响应的状态值. 完整的状态取值可参见 W3C site. 我们着重看值为 200 OK 的响应:

```
if (http_request.status == 200) {
    // perfect!
} else {
    // there was a problem with the request,
    // for example the response may be a 404 (Not Found)
    // or 500 (Internal Server Error)
    response codes
}
```

在检查完请求的状态值和响应的 HTTP 状态值后, 您就可以处理从服务器得到的数据了.有两种方式可以得到这些数据:

http_request.responseText – 以文本字符串的方式返回服务器的响应

http_request.responseXML – 以 XMLDocument 对象方式返回响应.处理 XMLDocument 对象可以用 JavaScript DOM 函数

3、简单实例

我们现在将整个过程完整地做一次,发送一个简单的 HTTP 请求.

我们用 JavaScript 请求一个 HTML 文件, test.html, 文件的文本内容为"I'm a test.",

然后我们"alert()"test.html 文件的内容.

```
<script type="text/javascript" language="javascript">
    var http_request = false;

    function makeRequest(url) {
      http_request = false;
      if (window.XMLHttpRequest) { // Mozilla, Safari,...
          http_request = new XMLHttpRequest();
          if (http_request.overrideMimeType) {
            http_request.overrideMimeType('text/xml');
          }
       } else if (window.ActiveXObject) { // IE
          try {
              http_request = new ActiveXObject("Msxml2.XMLHTTP");
          }
          catch (e) {
              try {
                  http_request = new ActiveXObject("Microsoft.XMLHTTP");
              }catch(e){}
          }
      }
      if (!http_request) {
          alert('Giving up :( Cannot create an XMLHTTP instance');
          return false;
      }
      http_request.onreadystatechange = alertContents;
      http_request.open('GET', url, true);
      http_request.send(null);
    }
    function alertContents() {
      if (http_request.readyState == 4) {
          if (http_request.status == 200) {
              alert(http_request.responseText);
          }else{
              alert('There was a problem with the request.');
          }
```

```
        }
    }
    </script>
    <style>
        .testbtn{ cursor: pointer; text-decoration: underline;}
    </style>
    <span class='testbtn' onclick="makeRequest('test.html')">Make a request</span>
```

本例中:

用户点击浏览器上的"请求"链接;

接着函数 makeRequest()将被调用.其参数为 HTML 文件 test.html 在同一目录下;

这样就发起了一个请求.onreadystatechange 的执行结果会被传送给 alertContents();

alertContents()将检查服务器的响应是否成功地收到,如果是,就会"alert()"test.html 文件的内容.

----------分割线----------

3、Ajax 的跨域

概念：只要协议、域名、端口有任何一个不同,都被当作是不同的域。

URL	说明	是否允许通信
http://www.a.com/a.js http://www.a.com/b.js	同一域名下	允许
http://www.a.com/lab/a.js http://www.a.com/script/b.js	同一域名下不同文件夹	允许
http://www.a.com:8000/a.js http://www.a.com/b.js	同一域名,不同端口	不允许
http://www.a.com/a.js https://www.a.com/b.js	同一域名,不同协议	不允许
http://www.a.com/a.js http://70.32.92.74/b.js	域名和域名对应 ip	不允许
http://www.a.com/a.js http://script.a.com/b.js	主域相同,子域不同	不允许
http://www.a.com/a.js http://a.com/b.js	同一域名,不同二级域名（同上）	不允许
http://www.cnblogs.com/a.js http://www.a.com/b.js	不同域名	不允许

对于端口和协议的不同,只能通过后台来解决。

A、跨域资源共享（CORS）

1、CORS（Cross-Origin Resource Sharing）跨域资源共享,定义了必须在访问跨域资源时,

浏览器与服务器应该如何沟通。

2、CORS 背后的基本思想就是使用自定义的 HTTP 头部让浏览器与服务器进行沟通,从而决定请求或响应是应该成功还是失败。

```
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "/trigkit4",true);
    xhr.send();
</script>
```

以上的 trigkit4 是相对路径,如果我们要使用 CORS,相关 Ajax 代码可能如下所示：

```
<script type="text/javascript">
    var xhr = new XMLHttpRequest();
    xhr.open("GET", "http://segmentfault.com/u/trigkit4/",true);
    xhr.send();
 </script>
```

代码与之前的区别就在于相对路径换成了其他域的绝对路径,也就是你要跨域访问的接口地址。

服务器端对于 CORS 的支持,主要就是通过设置 Access-Control-Allow-Origin 来进行的。

如果浏览器检测到相应的设置,就可以允许 Ajax 进行跨域的访问。

要解决跨域的问题,我们可以使用以下几种方法：

B、通过 jsonp 跨域

现在问题来了？什么是 jsonp？

1、维基百科的定义是：JSONP（JSON with Padding）是资料格式 JSON 的一种“使用模式”,

可以让网页从别的网域要资料。

2、JSONP 也叫填充式 JSON,是应用 JSON 的一种新方法,只不过是被包含在函数调用中的 JSON,

例如: callback({"name","trigkit4"});

JSONP 由两部分组成：回调函数和数据：

回调函数是当响应到来时应该在页面中调用的函数,

而数据就是传入回调函数中的 JSON 数据。

在 js 中,我们直接用 XMLHttpRequest 请求不同域上的数据时,是不可以的。

但是,在页面上引入不同域上的 js 脚本文件却是可以的,jsonp 正是利用这个特性来实现的。例如：

```
<script type="text/javascript">
    function dosomething(jsondata){
        //处理获得的 json 数据
    }
</script>
<script src="http://example.com/data.php？callback=dosomething"></script>
```

js 文件载入成功后会执行我们在 url 参数中指定的函数,并且会把我们需要的 json 数据作为参数传入。

所以 jsonp 是需要服务器端的页面进行相应的配合的。

```
<？php
    $callback = $_GET['callback'];//得到回调函数名
    $data = array('a','b','c');//要返回的数据
```

```
        echo $callback.'('.json_encode($data).')';//输出
    ？>
```

最终,输出结果为：dosomething(['a','b','c']);

如果你的页面使用 jquery,那么通过它封装的方法就能很方便的来进行 jsonp 操作了。

```
<script type="text/javascript">
    $.getJSON('http://example.com/data.php？callback=？,function(jsondata)'){
        //处理获得的 json 数据
    });
</script>
```

jquery 会自动生成一个全局函数来替换 callback=？中的问号,之后获取到数据后又会自动销毁,

实际上就是起一个临时代理函数的作用。

$.getJSON 方法会自动判断是否跨域：

1、不跨域的话,就调用普通的 ajax 方法；

2、跨域的话,则会以异步加载 js 文件的形式来调用 jsonp 的回调函数。

JSONP 的优点是：

1、它不像 XMLHttpRequest 对象实现的 Ajax 请求那样受到同源策略的限制；

2、它的兼容性更好,在更加古老的浏览器中都可以运行,不需要 XMLHttpRequest 或 ActiveX 的支持；

3、并且在请求完毕后可以通过调用 callback 的方式回传结果。

JSONP 的缺点则是：

1、它只支持 GET 请求而不支持 POST 等其它类型的 HTTP 请求；

2、它只支持跨域 HTTP 请求这种情况,不能解决不同域的两个页面之间如何进行 JavaScript 调用的问题。

CORS 和 JSONP 对比：

CORS 与 JSONP 相比,无疑更为先进、方便和可靠。

1、JSONP 只能实现 GET 请求,而 CORS 支持所有类型的 HTTP 请求。

2、使用 CORS,开发者可以使用普通的 XMLHttpRequest 发起请求和获得数据,比起 JSONP 有更好的错误处理。

3、JSONP 主要被老的浏览器支持,它们往往不支持 CORS,而绝大多数现代浏览器都已经支持了 CORS）。

C、通过修改 document.domain 来跨子域

浏览器都有一个同源策略：

1、第一个限制就是第一种方法中我们说的不能通过 ajax 的方法去请求不同源中的文档。

2、第二个限制是浏览器中不同域的框架之间是不能进行 js 的交互操作的。

不同的框架之间是可以获取 window 对象的,但却无法获取相应的属性和方法。

有一个页面,它的地址是 http://www.example.com/a.html ,
在这个页面里面有一个 iframe,它的 src 是 http://example.com/b.html,

这个页面与它里面的 iframe 框架是不同域的,所以我们是无法通过在页面中书写 js 代码来获取 iframe 中的东西的：

```
<script type="text/javascript">
    function test(){
        var iframe = document.getElementById('ifame');
        var win = document.contentWindow;
        //可以获取到 iframe 里的 window 对象,但该 window 对象的属性和方法几乎是不可用的
        var doc = win.document;//这里获取不到 iframe 里的 document 对象
        var name = win.name;//这里同样获取不到 window 对象的 name 属性
    }
</script>
<iframe id = "iframe" src="http://example.com/b.html" onload = "test()"></iframe>
```

这个时候,document.domain 就可以派上用场了:
1、只要设置 http://www.example.com/a.html 和 http://example.com/b.html
这两个页面的 document.domain 为相同的域名就可以了。
2、但要注意的是：document.domain 的设置是有限制的,
我们只能把 document.domain 设置成自身或更高一级的父域,且主域必须相同。

1.在页面 http://www.example.com/a.html 中设置 document.domain:

```
<iframe id = "iframe" src="http://example.com/b.html" onload = "test()"></iframe>
<script type="text/javascript">
    document.domain = 'example.com';//设置成主域
    function test(){
        //contentWindow 可取得子窗口的 window 对象
        alert(document.getElementById('iframe').contentWindow);
    }
</script>
```

2.在页面 http://example.com/b.html 中也设置 document.domain:

```
<script type="text/javascript">
    //在 iframe 载入这个页面也设置 document.domain,使之与主页面的 document.domain 相同
    document.domain = 'example.com';
</script>
```

修改 document.domain 的方法只适用于不同子域的框架间的交互

D、使用 window.name 来进行跨域

window 对象有个 name 属性,该属性有个特征：

即在一个窗口(window)的生命周期内,窗口载入的所有的页面都是共享一个 window.name 的,

每个页面对 window.name 都有读写的权限,window.name 是持久存在一个窗口载入过的所有页面中的

E、使用 HTML5 的 window.postMessage 方法跨域

window.postMessage(message,targetOrigin) 方法是 html5 新引进的特性；

可以使用它来向其它的 window 对象发送消息,无论这个 window 对象是属于同源或不同源,

目前 IE8+、FireFox、Chrome、Opera 等浏览器都已经支持 window.postMessage 方法。

----------分割线----------

4、优缺点

* 优点

1. 通过异步模式，提升了用户体验
2. 优化了浏览器和服务器之间的传输，减少不必要的数据往返，减少了带宽占用
3. Ajax 在客户端运行，承担了一部分本来由服务器承担的工作，减少了大用户量下的服务器负载。

* 缺点

1、ajax 不支持浏览器 back 按钮。
2、安全问题 AJAX 暴露了与服务器交互的细节。
3、对搜索引擎的支持比较弱。
4、破坏了程序的异常机制。

如何给 *js* 数组去重？

以下是数组去重的三种方法：

```
Array.prototype.unique1 = function () {
  var n = []; //一个新的临时数组
  for (var i = 0; i < this.length; i++) //遍历当前数组
  {
    //如果当前数组的第 i 已经保存进了临时数组，那么跳过，
    //否则把当前项 push 到临时数组里面
    if (n.indexOf(this[i]) == -1) n.push(this[i]);
  }
  return n;
}
```

```
Array.prototype.unique2 = function()
{
    var n = {},r=[]; //n 为 hash 表，r 为临时数组
```

```
    for(var i = 0; i < this.length; i++) //遍历当前数组
    {
        if (!n[this[i]]) //如果 hash 表中没有当前项
        {
            n[this[i]] = true; //存入 hash 表
            r.push(this[i]); //把当前数组的当前项 push 到临时数组里面
        }
    }
    return r;
}
----------------------------------------------------------------------------------------------
Array.prototype.unique3 = function()
{
    var n = [this[0]]; //结果数组
    for(var i = 1; i < this.length; i++) //从第二项开始遍历
    {
        //如果当前数组的第 i 项在当前数组中第一次出现的位置不是 i，
        //那么表示第 i 项是重复的，忽略掉。否则存入结果数组
        if (this.indexOf(this[i]) == i) n.push(this[i]);
    }
    return n;
}
```

sql 注入原理

就是通过把 SQL 命令插入到 Web 表单递交或输入域名或页面请求的查询字符串，最终达到欺骗服务器执行恶意的 SQL 命令。

总的来说有以下几点：

1.永远不要信任用户的输入，要对用户的输入进行校验，可以通过正则表达式，或限制长度，对单引号和双"-"进行转换等。

2.永远不要使用动态拼装 SQL，可以使用参数化的 SQL 或者直接使用存储过程进行数据查询存取。

3.永远不要使用管理员权限的数据库连接，为每个应用使用单独的权限有限的数据库连接。

4.不要把机密信息明文存放，请加密或者 hash 掉密码和敏感的信息。

XSS 原理及防范

Xss(cross-site scripting)攻击指的是攻击者往 Web 页面里插入恶意 html 标签或者 javascript 代码。比如：攻击者在论坛中放一个 看似安全的链接，骗取用户点击后，窃取 cookie 中的用户私密信息；或者攻击者在论坛中加一个恶意表单， 当用户提交表单的时候，却把信息传送到攻击者的服务器中，而不是用户原本以为的信任站点。

XSS 防范方法

1.代码里对用户输入的地方和变量都需要仔细检查长度和对”<”,”>”,”;”,”’”等字符做过滤；其次任何内容写到页面之前都必须加以 encode，避免不小心把 html tag 弄出来。这一个层面做好，至少可以堵住超过一半的 XSS 攻击。

2.避免直接在 cookie 中泄露用户隐私，例如 email、密码等等。 3.通过使 cookie 和系统 ip 绑定来降低 cookie 泄露后的危险。这样攻击者得到的 cookie 没有实际价值，不可能拿来重放。
4.尽量采用 POST 而非 GET 提交表单
XSS 与 CSRF 有什么区别吗？
XSS 是获取信息，不需要提前知道其他用户页面的代码和数据包。CSRF 是代替用户完成指定的动作，需要知道其他用户页面的代码和数据包。
要完成一次 CSRF 攻击，受害者必须依次完成两个步骤：
1.登录受信任网站 A，并在本地生成 Cookie。 2.在不登出 A 的情况下，访问危险网站 B。
CSRF 的防御
1.服务端的 CSRF 方式方法很多样，但总的思想都是一致的，就是在客户端页面增加伪随机数。 2.使用验证码

说说 JS 的对象,类和对象的继承？

一、对象的定义,及构造函数 ： Javascript 面向对象编程（一）：封装

a、Javascript 是一种基于对象（object-based）的语言,
遇到的所有东西几乎都是对象;

b、但是,它又不是一种真正的面向对象编程（OOP）语言,
因为它的语法中没有 class（类）;

1、生成对象的原始模式
假定我们把猫看成一个对象,它有"名字"和"颜色"两个属性。

```
var Cat = {
    name : '',
    color : ''
}
```

现在,我们需要根据这个原型对象的规格（schema）,生成两个实例对象。

```
var cat1 = {}; // 创建一个空对象
    cat1.name = "大毛"; // 按照原型对象的属性赋值
    cat1.color = "黄色";
var cat2 = {};
    cat2.name = "二毛";
    cat2.color = "黑色";
```

好了,这就是最简单的封装了,把两个属性封装在一个对象里面。

但是有两个缺点:
一是如果多生成几个实例,写起来就非常麻烦；
二是实例与原型之间,没有任何办法,可以看出有什么联系。

2、原始模式的改进
我们可以写一个函数,解决代码重复的问题。

```
function Cat(name,color){
```

```
        return {
            name:name,
            color:color
        }
    }
```

然后生成实例对象,就等于是在调用函数：

```
var cat1 = Cat("大毛","黄色");
var cat2 = Cat("二毛","黑色");
```

这种方法的问题依然是,cat1 和 cat2 之间没有内在的联系,不能反映出它们是同一个原型对象的实例。

3、构造函数模式

为了解决从原型对象生成实例的问题,Javascript 提供了一个构造函数（Constructor）模式。

所谓"构造函数",其实就是一个普通函数,但是内部使用了 this 变量。

对构造函数使用 new 运算符,就能生成实例,并且 this 变量会绑定在实例对象上。

比如,猫的原型对象现在可以这样写,

```
function Cat(name,color){
    this.name=name;
    this.color=color;
}
```

我们现在就可以生成实例对象了。

```
var cat1 = new Cat("大毛","黄色");
var cat2 = new Cat("二毛","黑色");
alert(cat1.name); //  大毛
alert(cat1.color); //  黄色
```

这时 cat1 和 cat2 会自动含有一个 constructor 属性,指向它们的构造函数。

```
alert(cat1.constructor == Cat); //true
alert(cat2.constructor == Cat); //true
```

Javascript 还提供了一个 instanceof 运算符,验证原型对象与实例对象之间的关系。

```
alert(cat1 instanceof Cat); //true
alert(cat2 instanceof Cat); //true
```

4、构造函数模式的问题

构造函数方法很好用,但是存在一个浪费内存的问题。

请看,我们现在为 Cat 对象添加一个不变的属性"type"（种类）,再添加一个方法 eat（吃老鼠）,原型对象 Cat 就变成了下面这样：

```
function Cat(name,color){
    this.name = name;
    this.color = color;
    this.type = "猫科动物";
    this.eat = function(){alert("吃老鼠");};
}
```

还是采用同样的方法,生成实例：

```
var cat1 = new Cat("大毛","黄色");
var cat2 = new Cat ("二毛","黑色");
alert(cat1.type); //  猫科动物
cat1.eat(); //  吃老鼠
```

表面上好像没什么问题,但是实际上这样做,有一个很大的弊端。

那就是对于每一个实例对象,type 属性和 eat()方法都是一模一样的内容,每一次生成一个实例,

都必须为重复的内容,多占用一些内存。这样既不环保,也缺乏效率。

```
alert(cat1.eat == cat2.eat); //false
```

能不能让 type 属性和 eat()方法在内存中只生成一次,然后所有实例都指向那个内存地址呢？回答是可以的。

5、Prototype 模式

Javascript 规定,每一个构造函数都有一个 prototype 属性,指向另一个对象。

这个对象的所有属性和方法,都会被构造函数的实例继承。

这意味着,我们可以把那些不变的属性和方法,直接定义在 prototype 对象上。

```
function Cat(name,color){
    this.name = name;
    this.color = color;
}
Cat.prototype.type = "猫科动物";
Cat.prototype.eat = function(){alert("吃老鼠")};
```

然后,生成实例。

```
var cat1 = new Cat("大毛","黄色");
var cat2 = new Cat("二毛","黑色");
alert(cat1.type); //  猫科动物
cat1.eat(); //  吃老鼠
```

这时所有实例的 type 属性和 eat()方法,其实都是同一个内存地址,指向 prototype 对象,因此就提高了运行效率。

```
alert(cat1.eat == cat2.eat); //true
```

6、Prototype 模式的验证方法

为了配合 prototype 属性,Javascript 定义了一些辅助方法,帮助我们使用它;

6.1 isPrototypeOf()

这个方法用来判断,某个 proptotype 对象和某个实例之间的关系。

```
alert(Cat.prototype.isPrototypeOf(cat1)); //true
alert(Cat.prototype.isPrototypeOf(cat2)); //true
```

6.2 hasOwnProperty()

每个实例对象都有一个 hasOwnProperty()方法,

用来判断某一个属性到底是本地属性,还是继承自 prototype 对象的属性。

```
alert(cat1.hasOwnProperty("name")); // true
alert(cat1.hasOwnProperty("type")); // false
```

6.3 in 运算符

in 运算符可以用来判断,某个实例是否含有某个属性,不管是不是本地属性。

```
alert("name" in cat1); // true
alert("type" in cat1); // true
```

in 运算符还可以用来遍历某个"对象"的所有属性。

```
for(var prop in cat1) {
    alert("cat1["+prop+"]="+cat1[prop]);
}
```

----------分割线----------

二、**对象构造函数的继承**　Javascript 面向对象编程（二）：构造函数的继承

今天要介绍的是,对象之间的"继承"的五种方法:

现在有一个"动物"对象的构造函数:

```
function Animal(){
    this.species = "动物";
}
```

还有一个"猫"对象的构造函数:

```
function Cat(name,color){
    this.name = name;
    this.color = color;
}
```

怎样才能使"猫"继承"动物"呢?

1、构造函数绑定

第一种方法也是最简单的方法,使用 call（子对象,单个参数）或 apply（子对象,数组参数）方法,

将父对象的构造函数绑定在子对象上,即在子对象构造函数中加一行：

```
function Cat(name,color){
    Animal.apply(this, arguments);
    this.name = name;
    this.color = color;
}
var cat1 = new Cat("大毛","黄色");
alert(cat1.species); //  动物
```

2、prototype 模式

第二种方法更常见,使用 prototype 属性,

如果"猫"的 prototype 对象,指向一个 Animal 的实例,那么所有"猫"的实例,就能继承 Animal 了。

```
Cat.prototype = new Animal();
Cat.prototype.constructor = Cat;
var cat1 = new Cat("大毛","黄色");
alert(cat1.species); //  动物
```

代码的第一行,我们将 Cat 的 prototype 对象指向一个 Animal 的实例。

```
Cat.prototype = new Animal();
```

它相当于完全删除了 prototype 对象原先的值,然后赋予一个新值。但是,第二行又是什么意思呢?

```
Cat.prototype.constructor = Cat;
```

原来,任何一个 prototype 对象都有一个 constructor 属性,指向它的构造函数;

如果没有"Cat.prototype = new Animal();"这一行,Cat.prototype.constructor 是指向 Cat 的;

加了这一行以后,Cat.prototype.constructor 指向 Animal;

```
alert(Cat.prototype.constructor == Animal); //true
```

更重要的是,每一个实例也有一个 constructor 属性,默认调用 prototype 对象的 constructor 属性。

```
alert(cat1.constructor == Cat.prototype.constructor); // true
```

因此,在运行"Cat.prototype = new Animal();"这一行之后,cat1.constructor 也指向 Animal!

```
alert(cat1.constructor == Animal); // true
```

这显然会导致继承链的紊乱(cat1 明明是用构造函数 Cat 生成的),因此我们必须手动纠正,

将 Cat.prototype 对象的 constructor 值改为 Cat。这就是第二行的意思;

很重要的一点,编程时务必要遵守。下文都遵循这一点,即如果替换了 prototype 对象,

```
o.prototype = {};
```

那么,下一步必然是为新的 prototype 对象加上 constructor 属性,并将这个属性指回原来的构造函数。

```
o.prototype.constructor = o;
```

3、直接继承 prototype

第三种方法是对第二种方法的改进。

由于 Animal 对象中,不变的属性都可以直接写入 Animal.prototype。

所以,我们也可以让 Cat()跳过 Animal(),直接继承 Animal.prototype。

现在,我们先将 Animal 对象改写:

```
function Animal(){}
Animal.prototype.species = "动物";
```

然后,将 Cat 的 prototype 对象指向 Animal 的 prototype 对象,这样就完成了继承。

```
Cat.prototype = Animal.prototype;
Cat.prototype.constructor = Cat;
var cat1 = new Cat("大毛","黄色");
alert(cat1.species); // 动物
```

与前一种方法相比,这样做的优点是效率比较高(不用执行和建立 Animal 的实例了),比较省内存。

缺点是 **Cat.prototype 和 Animal.prototype 现在指向了同一个对象,**
那么任何对 **Cat.prototype 的修改,都会反映到 Animal.prototype。**

所以,上面这一段代码其实是有问题的。请看第二行

```
Cat.prototype.constructor = Cat;
```

这一句实际上把 Animal.prototype 对象的 constructor 属性也改掉了,所有请看第四种方法！！！！

```
alert(Animal.prototype.constructor); // Cat
```

4、利用空对象作为中介

由于"直接继承 prototype"存在上述的缺点,所以就有第四种方法,利用一个空对象作为中介。

```
var F = function(){};
F.prototype = Animal.prototype;
Cat.prototype = new F();
Cat.prototype.constructor = Cat;
```

F 是空对象,所以几乎不占内存。这时,修改 Cat 的 prototype 对象,就不会影响到 Animal 的 prototype 对象。

```
alert(Animal.prototype.constructor); // Animal
```

我们将上面的方法,封装成一个函数,便于使用。

```
function extend(Child, Parent) {
    var F = function(){};
    F.prototype = Parent.prototype;
    Child.prototype = new F();
    Child.prototype.constructor = Child;
    Child.uber = Parent.prototype;
}
```

使用的时候,方法如下

```
extend(Cat,Animal);
var cat1 = new Cat("大毛","黄色");
alert(cat1.species); //  动物
```

这个 extend 函数,就是 YUI 库如何实现继承的方法。

另外,说明一点,函数体最后一行：

```
Child.uber = Parent.prototype;
```

意思是为子对象设一个 uber 属性,这个属性直接指向父对象的 prototype 属性,uber 是一个德语词,意思是"向上"、"上一层"。

这等于在子对象上打开一条通道,可以直接调用父对象的方法。这一行放在这里,只是为了实现继承的完备性,纯属备用性质。

5、拷贝继承

上面是采用 prototype 对象,实现继承。我们也可以换一种思路,纯粹采用"拷贝"方法实现继承。

简单说,如果把父对象的所有属性和方法,拷贝进子对象,不也能够实现继承吗？这样我们就有了第五种方法。

首先,还是把 Animal 的所有不变属性,都放到它的 prototype 对象上。

```
function Animal(){}
Animal.prototype.species = "动物";
```

然后,再写一个函数,实现属性拷贝的目的。

```
function extend2(Child, Parent) {
    var p = Parent.prototype;
    var c = Child.prototype;
    for (var i in p) {
        c[i] = p[i];
    }
    return c;
}
```

这个函数的作用,就是将父对象的 prototype 对象中的属性,一一拷贝给 Child 对象的 prototype 对象

（但这里 for in 属于浅拷贝,因为 for in 会遍历原型链上的属性） 。

使用的时候,这样写：

```
extend2(Cat, Animal);
var cat1 = new Cat("大毛","黄色");
alert(cat1.species); // 动物
```

----------分割线----------

三、非构造函数的继承 Javascript 面向对象编程（三）：非构造函数的继承

第一部分介绍了"封装"；

第二部分介绍了使用构造函数实现"继承"。

第三部分介绍不使用构造函数实现"继承"。

1、什么是"非构造函数"的继承？

比如,现在有一个对象,叫做"中国人"。

```
var Chinese = {
    nation:'中国'
};
```

还有一个对象,叫做"医生"。

```
var Doctor ={
    career:'医生'
}
```

请问怎样才能让"医生"去继承"中国人",也就是说,我怎样才能生成一个"中国医生"的对象？

这里要注意,这两个对象都是普通对象,不是构造函数,无法使用构造函数方法实现"继承"。

2、object()方法 返回 new 原型对象的方法 （创建 jquery 对象就是用这个原理,但多了一步判断）

json 格式的发明人 Douglas Crockford,提出了一个 object()函数,可以做到这一点。

```
function object(o) {
    function F() {}
    F.prototype = o;
    return new F();
}
```

这个 object()函数,其实只做一件事,就是把子对象的 prototype 属性,指向父对象,从而使得子对象与父对象连在一起。

使用的时候,第一步先在父对象的基础上,生成子对象：

```
var Doctor = object(Chinese);
```

然后,再加上子对象本身的属性：

```
Doctor.career = '医生';
```

这时,子对象已经继承了父对象的属性了。

```
alert(Doctor.nation); //中国
```

3、object.create()方法

a.在父对象：

```
var Chinese={"nation":"中国"};
```

b.建立子对象：

```
var Doctor= Object.create(Chinese);
```

c.再加上子对象本身的属性：

```
Doctor.career = '医生';
```

这时,子对象已经继承了父对象的属性了。

```
console.log(Doctor.nation + Doctor.career);//中国医生
```

4、浅拷贝

除了使用"prototype 链"以外,还有另一种思路：把父对象的属性,全部拷贝给子对象,也能实现继承。

下面这个函数,就是在做拷贝：

```
function extendCopy(p) {
    var c = {};
    for (var i in p) {
        c[i] = p[i];
    }
```

```
        return c;
    }
```

使用的时候,这样写：

```
    var Doctor = extendCopy(Chinese);
    Doctor.career = '医生';
    alert(Doctor.nation); //  中国
```

但是,这样的拷贝有一个问题。那就是,如果父对象的属性等于数组或另一个对象,那么实际上,子对象获得的只是一个内存地址,而不是真正拷贝,因此存在父对象被篡改的可能。

请看,现在给 Chinese 添加一个"出生地"属性,它的值是一个数组。

```
    Chinese.birthPlaces = ['北京','上海','香港'];
```

通过 extendCopy()函数,Doctor 继承了 Chinese。

```
    var Doctor = extendCopy(Chinese);
```

然后,我们为 Doctor 的"出生地"添加一个城市：

```
    Doctor.birthPlaces.push('厦门');
```

发生了什么事？Chinese 的"出生地"也被改掉了！

```
    alert(Doctor.birthPlaces); //北京，上海，香港，厦门
    alert(Chinese.birthPlaces); //北京，上海，香港，厦门
```

所以,extendCopy()只是拷贝基本类型的数据,我们把这种拷贝叫做"浅拷贝"。这是早期 jQuery 实现继承的方式。

5、深拷贝

所谓"深拷贝",就是能够实现真正意义上的数组和对象的拷贝。它的实现并不难,只要递归调用"浅拷贝"就行了。

```
    function deepCopy(p, c) {
        var c = c || {};
        for (var i in p) {
            if(p.hasOwnProperty(i)){ //不去继承 p 原型链上的属性
                if (typeof p[i] === 'object') {
                    c[i] = (p[i].constructor === Array)  ？   [] : {};
                    deepCopy(p[i], c[i]);
                } else {
                        c[i] = p[i];
                }
            }
        }
        return c;
    }
```

使用的时候这样写：

```
var Doctor = deepCopy(Chinese);
```

现在,给父对象加一个属性,值为数组。然后,在子对象上修改这个属性：

```
Chinese.birthPlaces = ['北京','上海','香港'];
Doctor.birthPlaces.push('厦门');
```

这时,父对象就不会受到影响了。

```
alert(Doctor.birthPlaces); //北京，上海，香港，厦门
alert(Chinese.birthPlaces); //北京，上海，香港
```

目前,jQuery 库使用的就是这种继承方法。

栈和堆的区别？

栈区（stack）— 由编译器自动分配释放 ，存放函数的参数值，局部变量的值等。
堆区（heap） — 一般由程序员分配释放， 若程序员不释放，程序结束时可能由 OS 回收。
堆（数据结构）：堆可以被看成是一棵树，如：堆排序；
栈（数据结构）：一种先进后出的数据结构。

说说你对 Promise 的理解

依照 Promise/A+ 的定义，Promise 有四种状态：
pending: 初始状态, 非 fulfilled 或 rejected.
fulfilled: 成功的操作.
rejected: 失败的操作.
settled: Promise 已被 fulfilled 或 rejected，且不是 pending
另外， fulfilled 与 rejected 一起合称 settled。
Promise 对象用来进行延迟(deferred) 和异步(asynchronous) 计算。
Promise 的构造函数
构造一个 Promise，最基本的用法如下：

```
var promise = new Promise(function(resolve, reject) {
    if (...) {   // succeed
        resolve(result);
    } else {     // fails
        reject(Error(errMessage));
    }
});
```

Promise 实例拥有 then 方法（具有 then 方法的对象，通常被称为 thenable）。它的使用方法

如下：
promise.then(onFulfilled, onRejected)
接收两个函数作为参数，一个在 fulfilled 的时候被调用，一个在 rejected 的时候被调用，接收参数就是 future，onFulfilled 对应 resolve, onRejected 对应 reject。

谈谈 This 对象的理解

- this 总是指向函数的直接调用者（而非间接调用者）；
- 如果有 new 关键字，this 指向 new 出来的那个对象；
- 在事件中，this 指向触发这个事件的对象，特殊的是，IE 中的 attachEvent 中的 this 总是指向全局对象 Window；

函数内部 arguments 变量有哪些特性,有哪些属性,如何将它转换为数组

- arguments 所有函数中都包含的一个局部变量，是一个类数组对象，对应函数调用时的实参。如果函数定义同名参数会在调用时覆盖默认对象
- arguments[index]分别对应函数调用时的实参，并且通过 arguments 修改实参时会同时修改实参
- arguments.length 为实参的个数（Function.length 表示形参长度）
- arguments.callee 为当前正在执行的函数本身，使用这个属性进行递归调用时需注意 this 的变化
- arguments.caller 为调用当前函数的函数（已被遗弃）
- 转换为数组：`var args = Array.prototype.slice.call(arguments, 0);`

offsetWidth/offsetHeight,clientWidth/clientHeight 与 scrollWidth/scrollHeight 的区别

- offsetWidth/offsetHeight 返回值包含 **content + padding + border**，效果与 e.getBoundingClientRect()相同
- clientWidth/clientHeight 返回值只包含 **content + padding**，如果有滚动条，也**不包含滚动条**
- scrollWidth/scrollHeight 返回值包含 **content + padding + 溢出内容的尺寸**

JSON 的了解？XML 和 JSON 的区别？

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'} JSON.parse('str') //转换 string 为 JSON 格式 JSON.stringfy('JSON') //转换 JSON 为 string 格式
*区别：
(1).数据体积方面。
JSON 相对于 XML 来讲，数据的体积小，传递的速度更快些。
(2).数据交互方面。
JSON 与 JavaScript 的交互更加方便，更容易解析处理，更好的数据交互。
(3).数据描述方面。
JSON 对数据的描述性比 XML 较差。
(4).传输速度方面。
JSON 的速度要远远快于 XML。

哪些地方会出现 css 阻塞,哪些地方会出现 js 阻塞？

js 的阻塞特性：
所有浏览器在下载 JS 的时候,会阻止一切其他活动,比如其他资源的下载,内容的呈现等等。
直到 JS 下载、解析、执行完毕后才开始继续并行下载其他资源并呈现内容。
为了提高用户体验,新一代浏览器都支持并行下载 JS,但是 JS 下载仍然会阻塞其它资源的下载（例如.图片,css 文件等）。

*由于浏览器为了防止出现 JS 修改 DOM 树,需要重新构建 DOM 树的情况,所以就会阻塞其他的下载和呈现。
*嵌入 JS 会阻塞所有内容的呈现,而外部 JS 只会阻塞其后内容的显示,2 种方式都会阻塞其后资源的下载。
也就是说外部样式不会阻塞外部脚本的加载,但会阻塞外部脚本的执行。

CSS 怎么会阻塞加载了？

CSS 本来是可以并行下载的,在什么情况下会出现阻塞加载了(在测试观察中,IE6 下 CSS 都是阻塞加载）

*当 CSS 后面跟着嵌入的 JS 的时候,该 CSS 就会出现阻塞后面资源下载的情况;
*而当把嵌入 JS 放到 CSS 前面,就不会出现阻塞的情况了。

根本原因：

*因为浏览器会维持 html 中 css 和 js 的顺序,样式表必须在嵌入的 JS 执行前先加载、解析完。

*而嵌入的 JS 会阻塞后面的资源加载,所以就会出现上面 CSS 阻塞下载的情况。

嵌入 JS 应该放在什么位置？

1、放在底部,虽然放在底部照样会阻塞所有呈现,但不会阻塞资源下载。
2、如果嵌入 JS 放在 head 中,请把嵌入 JS 放在 CSS 头部。
3、使用 defer（只支持 IE）
4、不要在嵌入的 JS 中调用运行时间较长的函数,如果一定要用,可以用`setTimeout`来调用

其他问题

- 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？
- 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？
- 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？
- 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？
- 平时如何管理你的项目？
- 介绍一个你最得意的作品吧？

- **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？能谈谈你未来 3，5 年给自己的规划吗？**

菜鸟教程前端面试题

前端开发面试题

面试有几点需注意：(来源寒冬 winter 老师，github:@wintercn)

1. 面试题目： 根据你的等级和职位的变化，入门级到专家级，广度和深度都会有所增加。
2. 题目类型： 理论知识、算法、项目细节、技术视野、开放性题、工作案例。
3. 细节追问： 可以确保问到你开始不懂或面试官开始不懂为止，这样可以大大延展题目的区分度和深度，知道你的实际能力。因为这种知识关联是长时期的学习，临时抱佛脚绝对是记不住的。
4. (∴∴)
5. 资深的前端开发能把 absolute 和 relative 弄混，这样的人不要也罢，因为团队需要的是：你这个人具有可以依靠的才能（靠谱）。

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

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：
-
 <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）]

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, websocket, 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。

    当然也可以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;
即使在没有样式 CSS 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

- 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

- 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)

- WebSocket、SharedWorker；
- 也可以调用 localstorge、cookies 等本地存储方式；
-
- localstorge 另一个浏览上下文里被添加、修改或删除时，它都会触发一个事件，
- 我们通过监听事件，控制它的值来进行页面信息通信；

注意 quirks：Safari 在无痕模式下设置 localstorge 值时会抛出 QuotaExceededError 的异

常；

- 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？

- 通过 visibilityState 的值检测页面当前是否可见，以及打开网页的时间等；

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:red"></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。

- title 与 h1 的区别、b 与 strong 的区别、i 与 em 的区别？

- title 属性没有明确意义只表示是个标题，H1 则表示层次明确的标题，对页面信息的抓取也有很大的影响；
-
- strong 是标明重点内容，有语气加强的含义，使用阅读设备阅读网络时：会重读，而<B>是展示强调内容。
-
- i 内容展示为斜体，em 表示强调的文本；
-
- Physical Style Elements -- 自然样式标签

```
        b, i, u, s, pre
        Semantic Style Elements -- 语义样式标签
        strong, em, ins, del, code
```

> 应该准确使用语义样式标签，但不能滥用，如果不能确定时首选使用自然样式标签。

CSS

- 介绍一下标准的 CSS 的盒子模型？低版本 IE 的盒子模型有什么不同的？

```
        （1）有两种， IE 盒子模型、W3C 盒子模型；
        （2）盒模型： 内容(content)、填充(padding)、边界(margin)、 边框(border)；
```

> （3）区 别： IE 的 content 部分把 border 和 padding 计算了进去；

- 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？

- 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。
  -
  - :after 在元素之前添加内容,也可以用来做清除浮动。
  - :before 在元素之后添加内容
  - :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;
```

```
o        top: 0;
o        left: 0;
o        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。
- table        此元素会作为块级表格来显示
```

```
inherit      规定应该从父元素继承 display 属性的值
```

- position 的值 relative 和 absolute 定位原点是？

```
- absolute
  生成绝对定位的元素，相对于值不为 static 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？

```
- 新增各种 CSS 选择器  （: not(.input)：所有 class 不是“input”的节点）
- 圆角          （border-radius:8px）
- 多列布局       （multi-column layout）
- 阴影和反射       （Shadow\Reflect）
- 文字特效      （text-shadow、）
- 文字渲染      （Text-decoration）
- 线性渐变      （gradient）
- 旋转          （transform）
- 增加了旋转,缩放,定位,倾斜,动画，多背景
```

```
transform:\scale(0.85,0.90)\ translate(0px,-30px)\ skew(-9deg,0deg)\Anima
```

```
tion:
```

- 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
```

- 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 ？

```
* 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？

行框的排列会受到中间空白（回车\空格）等的影响，因为空格也属于字符,这些空白也会被应用样式，占据空间，所以会有间隔，把字符大小设为 0，就没有空格了。

- 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。

- style 标签写在 body 后与 body 前有什么区别？
- 什么是 CSS 预处理器 / 后处理器？

- - 预处理器例如：LESS、Sass、Stylus，用来预编译 Sass 或 less，增强了 css 代码的复用性，
- 还有层级、mixin、变量、循环、函数等，具有很方便的 UI 组件模块化开发能力，极大的提高工作效率。
-
- - 后处理器例如：PostCSS，通常被视为在完成的样式表中根据 CSS 规范处理 CSS，让其更有效；目前最常做的

> 是给 CSS 属性添加浏览器私有前缀，实现跨浏览器兼容性的问题。

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(old Object);
console.log(person.getInfo());//它拥有了 Func 的属性和方法
//"Sean"
console.log(Func.prototype);
```

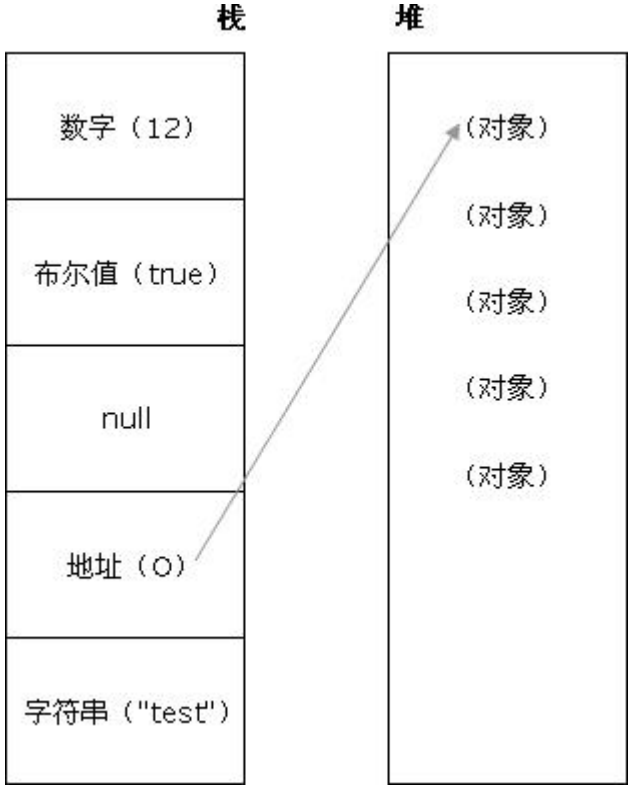
```
// Func { name="Sean", getInfo=function()}
```

- JavaScript 有几种类型的值？，你能画一下他们的内存图吗？

- 栈：原始数据类型（Undefined，Null，Boolean，Number、String）
- 堆：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



- 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 继承的几种实现方式？
  - 参考：构造函数的继承，非构造函数的继承；
- 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。
  - this 总是指向函数的直接调用者（而非间接调用者）；
  - 如果有 new 关键字，this 指向 new 出来的那个对象；
  - 在事件中，this 指向触发这个事件的对象，特殊的是，IE 中的 attachEvent 中的 this 总是指向全局对象 Window；

- 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）。

  由 JSON 字符串转换为 JSON 对象的时候可以用 eval，`var obj =eval('('+ str +')');`

- 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！
```

参考阅读：undefined 与 null 的区别

- 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"}
```

```
JSON 字符串转换为 JSON 对象：
var obj =eval('('+ str +')');
var obj = str.parseJSON();
var obj = JSON.parse(str);

JSON 对象转换为 JSON 字符串：
var last=obj.toJSONString();
```

```
var last=JSON.stringify(obj);
```

- `[].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?

同步的概念应该是来自于 OS 中关于同步的概念:不同进程为协同完成某项工作而在先后次序上调整(通过阻塞,唤醒等方式).同步强调的是顺序性.谁先谁后.异步则不存在这种顺序性.

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,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,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，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"
```

- What is a Polyfill?

- polyfill 是“在旧版浏览器上复制标准 API 的 JavaScript 补充”,可以动态地加载 JavaScript 代码或库，在不支持这些标准 API 的浏览器中模拟它们。
- 例如，geolocation（地理位置）polyfill 可以在 navigator 对象上添加全局的 geolocation 对象，还能添加 getCurrentPosition 函数以及“坐标”回调对象，
- 所有这些都是 W3C 地理位置 API 定义的对象和函数。因为 polyfill 模拟标准 API，所以能够以一种面向所有浏览器未来的方式针对这些 API 进行开发，

> 一旦对这些 API 的支持变成绝对大多数，则可以方便地去掉 polyfill，无需做任何额外工作。

- 做的项目中，有没有用过或自己实现一些 polyfill 方案（兼容性处理方案）？

> 比如： html5shiv、Geolocation、Placeholder

- 我们给一个 dom 同时绑定两个点击事件，一个用捕获，一个用冒泡。会执行几次事件，会先执行冒泡还是捕获？

ECMAScript6 相关

- Object.is() 与原来的比较操作符" ==="、" =="的区别？

- 两等号判等，会在比较时进行类型转换；
- 三等号判等(判断严格)，比较时不进行隐式类型转换，（类型不同则会返回 false）；
-
- Object.is 在三等号判等的基础上特别处理了 NaN 、-0 和 +0 ，保证 -0 和 +0 不再相同，
- 但 Object.is(NaN, NaN) 会返回 true.
-

> Object.is 应被认为有其特殊的用途，而不能用它认为它比其它的相等对比更宽松或严格。

前端框架相关

- react-router 路由系统的实现原理？
- React 中如何解决第三方类库的问题?

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？

转自：@markyun

原文：

https://github.com/markyun/My-blog/tree/master/Front-end-Developer-Questions/Questions-and-Answers

← 一探前端开发中的 JS 调试技巧

前端工程师必备技能图谱 →

- HTML 实体符号
- HTML 拾色器
- JSON 格式化工具

最新更新

- CSS3 text-empha...
- Vue2.0 新手入门...
- AngularJS2 模板...
- AngularJS2 表单
- AngularJS2 用户...
- jQuery Message
- AngularJS2 数据...

站点信息

- 意见反馈
- 免责声明
- 关于我们
- 文章归档

关注微信

反馈