# 面试题

## 浏览器兼容性问题的解决方法

1、不同浏览器的标签默认的外补丁( margin )和内补丁(padding)不同

解决方案： css 里增加通配符 \* { margin: 0; padding: 0; }

2、IE6双边距问题；在 IE6中设置了float , 同时又设置margin , 就会出现边距问题

解决方案：设置display:inline;

3、当标签的高度设置小于10px，在IE6、IE7中会超出自己设置的高度

解决方案：超出高度的标签设置overflow:hidden,或者设置line-height的值小于你的设置高度

4、图片默认有间距

解决方案：使用float 为img 布局

5、IE9以下浏览器不能使用opacity

解决方案：

opacity: 0.5;filter: alpha(opacity = 50);filter: progid:DXImageTransform.Microsoft.Alpha(style = 0, opacity = 50);

6、边距重叠问题；当相邻两个元素都设置了margin 边距时，margin 将取最大值，舍弃最小值；

解决方案：为了不让边重叠，可以给子元素增加一个父级元素，并设置父级元素为overflow:hidden；

7、cursor:hand 显示手型在safari 上不支持

解决方案：统一使用 cursor:pointer

8、两个块级元素，父元素设置了overflow:auto；子元素设置了position:relative ;且高度大于父元素，在IE6、IE7会被隐藏而不是溢出；

解决方案：父级元素设置position:relative

1、html,css,js代码，全部为标准模式

2、针对不支持的属性，需要通过浏览器前缀来解决

-webkit- : chrome、Safari

-moz- ： Firefox

-o- ： opera

-ms- ： Microsoft IE

3、针对微软IE的低版本浏览器，通过CSS HACK方式

（1）CSS类内部Hack

- IE8、9、10：\0

- IE9、10：\9\0

- IE7:\* , + , #

- IE6:\_ , -

-

（2）选择器Hack

\*前缀

\*+前缀

@media screen{}

（3）头部引入Hack

通过IE的条件注释引入不同的CSS文件

## [理解HTML语义化](https://www.cnblogs.com/freeyiyi1993/p/3615179.html)

**1、什么是HTML语义化？**

<基本上都是围绕着几个主要的标签，像标题（H1~H6）、列表（li）、强调（strong em）等等>

　　根据内容的结构化（内容语义化），选择合适的标签（代码语义化）便于开发者阅读和写出更优雅的代码的同时让浏览器的爬虫和机器很好地解析。

**2、为什么要语义化？**

* 为了在没有CSS的情况下，页面也能呈现出很好地内容结构、代码结构:为了裸奔时好看；
* 用户体验：例如title、alt用于解释名词或解释图片信息、label标签的活用；
* 有利于[SEO](http://baike.baidu.com/view/1047.htm" \t "_blank)：和搜索引擎建立良好沟通，有助于爬虫抓取更多的有效信息：[爬虫](http://baike.baidu.com/view/998403.htm" \t "_blank)依赖于标签来确定上下文和各个关键字的权重；
* 方便其他设备解析（如屏幕阅读器、盲人阅读器、移动设备）以意义的方式来渲染网页；
* 便于团队开发和维护，语义化更具可读性，是下一步吧网页的重要动向，遵循W3C标准的团队都遵循这个标准，可以减少差异化。

## H5的新特性

1.   用于绘画 canvas 元素。

2.   用于媒介回放的 video 和 audio 元素。

3.   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。

4.   语意化更好的内容元素，比如 article、footer、header、nav、section

5.   表单控件，calendar、date、time、email、url、search。

HTML5 添加了很多语义元素如下所示：

标签 描述

<article> 定义页面独立的内容区域。

<aside> 定义页面的侧边栏内容。

<bdi> 允许您设置一段文本，使其脱离其父元素的文本方向设置。

<command> 定义命令按钮，比如单选按钮、复选框或按钮

<details> 用于描述文档或文档某个部分的细节

<dialog> 定义对话框，比如提示框

<summary> 标签包含 details 元素的标题

<figure> 规定独立的流内容（图像、图表、照片、代码等等）。

<figcaption> 定义 <figure> 元素的标题

<footer> 定义 section 或 document 的页脚。

<header> 定义了文档的头部区域

<mark> 定义带有记号的文本。

<meter> 定义度量衡。仅用于已知最大和最小值的度量。

<nav> 定义导航链接的部分。

<progress> 定义任何类型的任务的进度。

CSS3新特性

选择器

盒模型

背景和边框

文字特效

2D/3D转换

动画

多列布局

用户界面

## Less和sass

**支持变量、函数、嵌套、混合、导入、运算、循环（when）**

**不支持连接符、if   else、switch  case**

**不能直接定义动画、媒体查询，需要    .animate{@-webkit-keyframes run{ }}间接使用**

**@：代表当前的对象**

**Sass中 $ 代表当前对象 支持判断等**

## Nodejs

Node.js 就是运行在服务端的 JavaScript,由C++语言编写。

Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

丰富的组件、路由

强大的包管理工具 npm,npm 一个全球最大的开源平台

## 如何降低前端耦合

1、 避免全局耦合

这应该是比较常见的耦合。全局耦合就是几个类、模块共用了全局变量或者全局数据结构，特别是一个变量跨了几个文件

2、js/css/html的耦合

不推荐直接在js里面更改样式属性，而应该通过增删类来控制样式，这样子样式还是回归到css文件里面

3、减少重复代码

整一个的思路是这样的：出现了重复代码 -> 封装成一个函数 -> 封装成一个模块 -> 封装成一个插件，抽象级别不断提高，将共有的特性和有差异的地方分离出来。当你走在抽象与封装的路上的时候，那你应该也是走在了大神的路上。

模块化的极端是拆分粒度太细，一个简单的功能，明明十行代码写在一起就可以搞定的事情，硬是写了七、八层函数栈，每个函数只有两、三行。这样除了把你的逻辑搞得太复杂之外，并没有太多的好处。当你出现了重复代码，或者是一个函数太大、功能太多，又或是逻辑里面写了三层循环又再嵌套了三层if，再或是你预感到你写的这个东西其他人也可能会用到，这个时候你才考虑模块化，进行拆分比较合适。

每个模块都有自己的职责，不可拆分，这在面向对象编程里面叫做单一责职原则，一个模块只负责一个功能。

## 什么是闭包（closure）？为什么要用它？

    闭包就是能够读取其他函数内部变量的函数（内部函数在外部函数的作用域外被调用，则形成了内部函数的闭包）

    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。

    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

    重用变量又不能造成全局污染

 使用闭包的注意点

1）滥用闭包，会造成内存泄漏：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）会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。

## 什么叫优雅降级和渐进增强？

**渐进增强 progressive enhancement：**

**针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。**

**优雅降级** **graceful degradation：**

**一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。**

**区别：**

**a. 优雅降级是从复杂的现状开始，并试图减少用户体验的供给**

**b. 渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要**

**c. 降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带**

## javascript面向对象中继承实现？

面向对象的基本特征有：封闭、继承、多态。

在JavaScript中实现继承的方法：

1. 原型链（prototype chaining）

2. call()/apply()

3. 混合方式(prototype和call()/apply()结合)

4. 对象冒充

继承的方法如下：

1、prototype原型链方式：

2、call()/apply()方法

3、混合方法【prototype,call/apply】

4、对象冒充

## cookies sessionStorage和localstorage区别

**相同点：**都存储在客户端

**不同点：**1.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。

2.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

3. 数据与服务器之间的交互方式

· cookie的数据会自动的传递到服务器，服务器端也可以写cookie到客户端

· sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存。

## Ajax

**Ajax：异步无刷新操作，进行局部刷新，更改数据而不是重新加载整个网页**

**同步：每次只能做一件事情；**

**异步：让别人帮忙做事情，返回最后的结过给自己**

* **AJAX的步骤**

**1、创建ajax对象（浏览器兼容）**

**2、ajax发送http请求**

**3、监听请求是否成功(onreadystatechange 进行 readyState 4 status 200)**

**0 请求未初始化（在调用 open() 之前）**

**1 请求已提出（调用 send() 之前）**

**2 请求已发送（这里通常可以从响应得到内容头部）**

**3 请求处理中（响应中通常有部分数据可用，但是服务器还没有完成响应）**

**4 请求已完成（可以访问服务器响应并使用它）**

**4、接收服务器返回的数据(responseText)**

**5、处理返回的数据进行页面渲染**

**第一步**，创建xmlhttprequest对象，var xmlhttp =new XMLHttpRequest（);XMLHttpRequest对象用来和服务器交换数据。

var xhttp;

if (window.XMLHttpRequest) {

//现代主流浏览器

xhttp = new XMLHttpRequest();

} else {

// 针对浏览器，比如IE5或IE6

xhttp = new ActiveXObject("Microsoft.XMLHTTP");

}

**第二步**，使用xmlhttprequest对象的open（）和send（）方法发送http资源请求给服务器。

**第三步**，使用xmlhttprequest对象的responseText或responseXML属性获得服务器的响应。

**第四步**，onreadystatechange函数，当发送请求到服务器，我们想要服务器响应执行一些功能就需要使用onreadystatechange函数，每次xmlhttprequest对象的readyState发生改变都会触发onreadystatechange函数

**什么是Ajax和JSON，他们的优缺点？**

    Ajax是异步JavaScript和XML，用于在Web页面中实现异步数据交互。

　　优点：

        可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量

        避免用户不断刷新或者跳转页面，提高用户体验

    缺点：

        对搜索引擎不友好（

        要实现ajax下的前后退功能成本较大

        可能造成请求数的增加

        跨域问题限制

**JSON是一种轻量级的数据交换格式，ECMA的一个子集**

    　　优点：轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）

## [Ajax中POST和GET的区别](https://www.cnblogs.com/ranyonsue/p/5888692.html)

Get和Post都是向服务器发送的一种请求，只是发送机制不同。

1. GET请求会将参数跟在URL后进行传递，而POST请求则是作为HTTP消息的实体内容发送给WEB服务器。当然在Ajax请求中，这种区别对用户是不可见的。

2. 首先是"GET方式提交的数据最多只能是1024字节"，因为GET是通过URL提交数据，那么GET可提交的数据量就跟URL的长度有直接关系了。而实际上，URL不存在参数上限的问题，HTTP协议规范没有对URL长度进行限制。这个限制是特定的浏览器及服务器对它的限制。IE对URL长度的限制是2083字节(2K+35)。对于其他浏览器，如Netscape、FireFox等，理论上没有长度限制，其限制取决于操作系统的支持。注意这是限制是整个URL长度，而不仅仅是你的参数值数据长度。

3. GET方式请求的数据会被浏览器缓存起来，因此其他人就可以从浏览器的历史记录中读取到这些数据，例如账号和密码等。在某种情况下，GET方式会带来严重的安全问题。而POST方式相对来说就可以避免这些问题。

get请求和post请求在服务器端的区别:

4. 在客户端使用get请求时,服务器端使用Request.QueryString来获取参数,而客户端使用post请求时,服务器端使用Request.Form来获取参数.

若符合下列任一情况，则用POST方法：  
 \* 请求的结果有持续性的副作用，例如，数据库内添加新的数据行。  
 \* 若使用GET方法，则表单上收集的数据可能让URL过长。  
 \* 要传送的数据不是采用7位的ASCII编码。  
若符合下列任一情况，则用GET方法：  
 \* 请求是为了查找资源，HTML表单数据仅用来帮助搜索。  
 \* 请求结果无持续性的副作用。  
 \* 收集的数据及HTML表单内的输入字段名称的总长不超过1024个字符。

## js延迟加载的方式有哪些？

js的延迟加载有助与提高页面的加载速度，以下是延迟加载的几种方法：

1. 使用setTimeout延迟方法的加载时间
2. defer属性：(页面load后执行)

延迟加载js代码，给网页加载留出更多时间

## .call()和.apply()的含义和区别？

    1、call，apply都属于Function.prototype的一个方法，它是JavaScript引擎内在实现的，因为属于Function.prototype，所以每个Function对象实例(就是每个方法)都有call，apply属性。既然作为方法的属性，那它们的使用就当然是针对方法的了，这两个方法是容易混淆的，因为它们的作用一样，只是使用方式不同。

    2、语法：foo.call(this, arg1,arg2,arg3) == foo.apply(this, arguments) == this.foo(arg1, arg2, arg3);

    3、相同点：两个方法产生的作用是完全一样的。

    4、不同点：方法传递的参数不同，单个单数传入，另一个可以以数组方式传入

## JQuery有几种选择器?

(1)、基本选择器：#id，class,element,\*;

(2)、层次选择器：parent > child，prev + next ，prev ~ siblings

(3)、基本[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8)选择器：:first，:last ，:not ，:even ，:odd ，:eq ，:gt ，:lt

(4)、内容[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8)选择器： :contains ，:empty ，:has ，:parent

(5)、可见性[过滤器](http://zhidao.baidu.com/search?word=%E8%BF%87%E6%BB%A4%E5%99%A8&fr=qb_search_exp&ie=utf8)选择器：:hidden ，:visible

(6)、属性过滤器选择器：[attribute] ，[attribute=value] ，[attribute!=value] ，[attribute^=value] ，[attribute$=value] ，[attribute\*=value]

(7)、子元素过滤器选择器：:nth-child ，:first-child ，:last-child ，:only-child

(8)、表单选择器： :input ，:text ，:password ，:radio ，:checkbox ，:submit 等；

(9)、表单过滤器选择器：:enabled ，:disabled ，:checked ，:selected

## src与href的区别

href 是指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，用于超链接。

src是指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求src资源时会将其指向的资源下载并应用到文档内，例如js脚本，img图片和frame等元素。

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将js脚本放在底部而不是头部。

## 一个Web页面的生命周期是什么?

**1. 页请求：页请求发生在页面生命周期开始之前，用户请求页时，[ASP.NET](http://asp.net/)将确定是否需要分析和编译页**

**2.开始：在开始阶段，将设置页属性，如Request和Response。在此阶段，页还将确定请求是回发请求还是新请求，并设置IsPostBack属性**

**3.页初始化：在页初始化期间，可以使用页中的控件，并设置每个控件的UniqueID属性。**

**4.加载：在加载期间，如果当前请求是回发请求，则将使用从视图状态和控件状态恢复的信息加载控件属性。**

**5.验证：在验证期间，将调用所有验证程序控件的Validate方法，此方法将设置各个验证程序控件和页的IsValidate属性。**

**6.回发事件处理：如果请求是回发请求，则将调用所有事件处理程序**

**7.呈现：在呈现之前，会针对该页和所有控件保存视图状态。在呈现阶段中，页会针对每个控件调用Render方法，它会提供一个文本编写器，用于将控件的输出写入页的Response属性和OutputStream中**

**8.卸载：完全呈现页并将页发送至客户端、准备丢弃该页后，将调用卸载。此时，将卸载页属性并执行清理。一般如果面试问到的话，只需要在说时前面稍加有艺术表达一下生命周期的意思，然后说出[ASP.NET](http://asp.net/)生命周期的几个步骤就可以了，估计面试官后面会针对某一个阶段深入问了，这个时候你就需要需要知道具体细节了。**

## http和https的区别

### 相同之处

大多情况下，HTTP和HTTPS是相同的，因为都是采用同一个基础的协议，作为HTTP或HTTPS客户端（浏览器），设立一个连接到Web服务器指定的端口。当服务器接收到请求，它会返回一个状态码以及消息，这个回应可能是请求信息、或者指示某个错误发送的错误信息。系统使用统一资源定位器 URI 模式，因此资源可以被唯一指定。整个过程中，唯一不同的只是一个协议头(HTTPS)的说明，其他都是一样的。

### 不同之处

HTTPS和HTTP的区别主要如下：

* **工作层**：在OSI网络模型中，HTTP工作于**应用层**，而HTTPS工作在传输层。
* **连接端口**：HTTP标准端口是80，而HTTPS的标准端口是443。
* **传输方式**：HTTP是超文本传输协议，信息是明文传输，而HTTPS是SSL加密传输协议。
* **工作耗时**：HTTP耗时=TCP握手，而HTTPS耗时=TCP握手+SSL握手。
* **显示形式**：HTTP的URL以http://开头，而HTTPS的URL以https://开头。
* **费用**：HTTP无需费用，而HTTPS需要到CA申请证书，一般免费证书较少，需要一定费用。
* **安全性**：HTTP的连接很简单，是无状态的；HTTPS协议是由SSL+HTTP协议构建的可进行加密传输、身份认证的网络协议，比HTTP协议安全。

### HTTPS的优点

尽管HTTPS并非绝对安全，掌握根证书的机构、掌握加密算法的组织同样可以进行中间人形式的攻击，但HTTPS仍是现行架构下最安全的解决方案，主要有以下几个有点：

* 使用HTTPS协议可认证用户和服务器，确保数据发送到正确的客户机和服务器；
* HTTPS是现行架构下最安全的解决方案，虽然不是绝对安全，但它大幅增加了中间人攻击的成本。
* 谷歌曾在2014年8月份调整搜索引擎算法，并称“比起同等HTTP网站，采用HTTPS加密的网站在搜索结果中的排名将会更高”。

## 从任务栏输入一个url地址到页面渲染成功的过程？

1. 查找浏览器自身的缓存（[chrome://net-internals/#dns](chrome://net-internals/" \l "dns)）
2. 查找系统本身的缓存（C:\Windows\System32\drivers\etc\hosts）
3. 查看运营商是否有缓存（移动、电信、联通）

若运营商存在缓存，则直接返回ip地址

若运营商不存在缓存，则发起请求

1. 发起http请求
2. 进行三次握手

客户端问服务器端：哥们，你能听见我说话吗？

服务器问客户端：哥们，我能听见你说话，你能听见我说话吗？

客户端对服务器端说：哥们，我能听见，我们开始说话吧！

1. 服务器响应数据到客户端
2. 客户端将响应的数据进行页面渲染（DOM进行渲染）hua

## link和@import的区别

* link属于html标签，而@import是css提供的。
* 页面被加载时，link会同时被加载，而@import引用的css会等到页面加载结束后加载。
* link是html标签，因此没有兼容性，而@import只有IE5以上才能识别。
* link方式样式的权重高于@import的。

## Css优先级别算法

**原则：**

**①　!important声明的样式优先级最高。**

**②　就近原则：如果优先级相同，则最后定义的样式会起作用，但应避免这种情况出现。**

**③　同权重情况下，内联样式表> 嵌入样式表> 外部样式表。**

**④　继承的样式的优先级低于指定的样式的优先级。**

**选择器的优先级：**

**①　在属性后面使用!important会覆盖页面中任何位置定义的元素样式**

**②　作为style属性写在元素内的样式**

**③　id选择器**

**④　类选择器**

**⑤　伪类选择器**

**⑥　标签选择器**

**⑦　通配符选择器**

## 前端性能优化方法

一）内容层面

1、DNS解析优化（DNS缓存、减少DNS查找、keep-alive、适当的主机域名）

  2、避免重定向（/还是需要的）

  3、切分到多个域名

  4、杜绝404

二）网络传输阶段

1、减少传输过程中实体的大小

    1）缓存

    2）cookie优化

    3）文件压缩（Accept-Encoding：g-zip）

2、减少请求的次数

    1）文件适当的合并

    2）雪碧图

3、异步加载（并发,requirejs）

4、预加载、延后加载、按需加载

三）渲染阶段

1、js放底部，css放顶部

2、减少重绘和回流

       3、合理使用Viewport 等meta头部

       4、减少dom节点

      5、BigPipe

四）脚本执行阶段

1、缓存节点，尽量减少节点的查找

2、减少节点的操作（innerHTML）

3、避免无谓的循环，break、continue、return的适当使用

4、事件委托

## 如何让div水平垂直剧中

方法一：

div使用绝对布局，设置margin:auto;并设置top、left、right、bottom的值相等即可，不一定要都是0

.main{

text-align: center; /\*让div内部文字居中\*/

background-color: #fff;

border-radius: 20px;

width: 300px;

height: 350px;

margin: auto;

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

}

方法二：

仍然是绝对布局，让left和top都是50%，这在水平方向上让div的最左与屏幕的最左相距50%，垂直方向上一样，所以再用transform向左（上）平移它自己宽度（高度）的50%，也就达到居中效果了，效果图和上方相同。

.main{

text-align: center;

background-color: #fff;

border-radius: 20px;

width: 300px;

height: 350px;

position: absolute;

left: 50%;

top: 50%;

transform: translate(-50%,-50%);

}

方法三：利用flex布局，实际应用考虑兼容性

Display : flex;

Align – items : center;

Justify – content : enter;

## Position的定位原点

fixed 固定定位，相对于浏览器窗口进行定位，脱离文档流

absolute 绝对定位， 相对于值不为 static的第一个父元素进行定位。脱离文档流

relative 相对定位, 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。

## JS跨域请求方式

通过修改document.domain来跨子域；

通过jsonp跨域

使用window.name来进行跨域

等

## ES6中let与var的区别

通过var定义的变量，作用域是整个封闭函数，是全域的-不论var声明的变量处于当前作用域的第几行，都会提升到作用域的头部 -var 声明的变量会被提升到作用域的顶部并初始化为undefined，而let声明的变量在作用域的顶部未被初始化 。通过let定义的变量，作用域是在块级或是子块中。

变量提升现象：浏览器在运行代码之前会进行预解析，首先解析函数声明，定义变量，解析完之后再对函数、变量进行运行、赋值等。

**但是直接用let声明变量不赋值是会打印undefined，还是初始化了，只是let声明放在赋值之后，let声明会提前但不会初始化。**

只要块级作用域内存在let命令，它所声明的变量就“绑定”这个区域，不再受外部的影响。总之，在代码块内，使用let命令声明变量之前，该变量都是不可用的，尽管代码块外也存在相同全局变量。

let不允许在相同作用域内，重复声明同一个变量。

## 同源策略

同源策略，它是由[Netscape](https://baike.baidu.com/item/Netscape/2778944" \t "_blank)提出的一个著名的[安全策略](https://baike.baidu.com/item/%E5%AE%89%E5%85%A8%E7%AD%96%E7%95%A5/2890163" \t "_blank)。

现在所有支持JavaScript 的浏览器都会使用这个策略。

所谓同源是指，域名，协议，端口相同。

当一个浏览器的两个tab页中分别打开来 百度和谷歌的页面

当浏览器的百度tab页执行一个脚本的时候会检查这个脚本是属于哪个页面的，

即检查是否同源，只有和百度同源的脚本才会被执行。[1]

如果非同源，那么在请求数据时，浏览器会在控制台中报一个异常，提示拒绝访问。