**不受同源策略限制的：**

1、页面中的链接，重定向以及表单提交是不会受到同源策略限制的。

2、跨域资源的引入是可以的。但是js不能读写加载的内容。如嵌入到页面中的

<script src="..."></script>、<img>、<link>、<iframe>等。

**JS实现跨域**

JSONP：通过动态创建script，再请求一个带参网址实现跨域通信。

CORS：服务端设置Access-Control-Allow-Origin即可，前端无须设置，若要带cookie请求，前后端都需要设置。

代理跨域：启一个代理服务器，实现数据的转发

document.domain + iframe跨域：两个页面都通过js强制设置document.domain为基础主域，就实现了同域。

location.hash + iframe跨域：a欲与b跨域相互通信，通过中间页c来实现。 三个页面，不同域之间利用iframe的location.hash传值，相同域之间直接js访问来通信。

window.name + iframe跨域：通过iframe的src属性由外域转向本地域，跨域数据即由iframe的window.name从外域传递到本地域。

postMessage跨域：可以跨域操作的window属性之一。

**JSONP：**

原理是：动态插入script标签，通过script标签引入一个js文件，这个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。

由于同源策略的限制，XmlHttpRequest只允许请求当前源（域名、协议、端口）的资源，为了实现跨域请求，可以通过script标签实现跨域请求，然后在服务端输出JSON数据并执行回调函数，从而解决了跨域的数据请求。

优点是兼容性好，简单易用，支持浏览器与服务器双向通信。缺点是只支持GET请求。

JSONP：json+padding（内填充），顾名思义，就是把JSON填充到一个盒子里

<script>

var script = document.createElement('script');

script.type = 'text/javascript';

// 传参并指定回调执行函数为onBack

script.src = 'http://www.domain2.com:8080/login?user=admin&callback=onBack';

document.head.appendChild(script);

// 回调执行函数

function onBack(res) {

alert(JSON.stringify(res));

}

</script>

**CORS：**

服务器端对于CORS的支持，主要就是通过设置response.setHeader("Access-Control-Allow-Origin", "http://www.domain1.com");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。

**跨域代理(比如nginx反向代理接口跨域):**

跨域原理： 同源策略是浏览器的安全策略，不是HTTP协议的一部分。服务器端调用HTTP接口只是使用HTTP协议，不会执行JS脚本，不需要同源策略，也就不存在跨越问题。

实现思路：通过nginx配置一个代理服务器（域名与domain1相同，端口不同）做跳板机，反向代理访问domain2接口，并且可以顺便修改cookie中domain信息，方便当前域cookie写入，实现跨域登录。

**通过修改document.domain + iframe来跨子域:**

**主域相同 ===> 使用document.domain**

将子域和主域的document.domain设为同一个主域.

前提条件：这两个域名必须属于同一个基础域名!而且所用的协议，端口都要一致，否则无法利用document.domain进行跨域

1.）父窗口：(http://www.domain.com/a.html)

<iframe id="iframe" src="http://child.domain.com/b.html"></iframe> //指向子窗口

<script>

document.domain = 'domain.com';

var user = 'admin';

</script>

2.）子窗口：(http://child.domain.com/b.html)

<script>

document.domain = 'domain.com';

// 获取父窗口中变量

alert('get js data from parent ---> ' + window.parent.user);

</script>

**location.hash + iframe跨域:**

实现原理： a欲与b跨域相互通信，通过中间页c来实现。

三个页面，不同域之间利用iframe的location.hash传值，相同域之间直接js访问来通信。

**使用window.name + iframe 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

**postMessage跨域:**

postMessage是HTML5中的API，且是为数不多可以跨域操作的window属性之一

用法：postMessage(data,origin)方法接受两个参数

data： html5规范支持任意基本类型或可复制的对象，但部分浏览器只支持字符串，所以传参时最好用JSON.stringify()序列化。

origin：协议+域名+端口号，也可以设置为"\*"，表示可以传递给任意窗口，如果要指定和当前窗口同源的话设置为"/"。

1.）a.html：(http://www.domain1.com/a.html)

<iframe id="iframe" src="http://www.domain2.com/b.html" style="display:none;"></iframe>

<script>

var iframe = document.getElementById('iframe');

iframe.onload = function() {

var data = {

name: 'aym'

};

// 向domain2传送跨域数据

iframe.contentWindow.postMessage(JSON.stringify(data), 'http://www.domain2.com');

};

// 接受domain2返回数据

window.addEventListener('message', function(e) {

alert('data from domain2 ---> ' + e.data);

}, false);

</script>

2.）b.html：(http://www.domain2.com/b.html)

<script>

// 接收domain1的数据

window.addEventListener('message', function(e) {

alert('data from domain1 ---> ' + e.data);

var data = JSON.parse(e.data); //将字符串转换为对象

if (data) {

data.number = 16;

// 处理后再发回domain1

window.parent.postMessage(JSON.stringify(data), 'http://www.domain1.com');

}

}, false);

</script>

**postMessage跨域可用于解决以下方面的问题：**

a.） 页面和其打开的新窗口的数据传递

b.） 多窗口之间消息传递

c.） 页面与嵌套的iframe消息传递

d.） 上面三个场景的跨域数据传递