cors(cross origin resource share)常见实现方案：

1. XMLHttpRequest:

预备知识：XMLHttpRequest 跨域请求分为：

a).简单跨域请求：当同时满足下列两个条件时视为简单跨域请求

(1).请求方法是 GET、HEAD 或者 POST，并且当请求方法是 POST 时，Content-Type 必须是 application/x-www-form-urlencoded, multipart/form-data 或着 text/plain 中的一个值

(2). 请求中没有自定义 HTTP 头部。

对于简单跨域请求，浏览器要做的就是在 HTTP 请求中添加 Origin Header，将 JavaScript 脚本所在域填充进去，向其他域的服务器请求资源。服务器端收到一个简单跨域请求后，根据资源权限配置，在响应头中添加 Access-Control-Allow-Origin Header。浏览器收到响应后，查看 Access-Control-Allow-Origin Header，如果当前域已经得到授权，则将结果返回给 JavaScript。否则浏览器忽略此次响应

b).带预检的跨域请求（preflighted request）:不满足简单跨域请求条件的即为 preflighted request.

带预检(Preflighted)的跨域请求需要浏览器在发送真实 HTTP 请求之前先发送一个 OPTIONS 的预检请求，检测服务器端是否支持真实请求进行跨域资源访问，真实请求的信息在 OPTIONS 请求中通过 Access-Control-Request-Method Header 和 Access-Control-Request-Headers Header 描述，此外与简单跨域请求一样，浏览器也会添加 Origin Header。服务器端接到预检请求后，根据资源权限配置，在响应头中放入 Access-Control-Allow-Origin Header,Access-Control-Allow-Methods Header 和 Access-Control-Allow-Headers Header，分别表示允许跨域资源请求的域、请求方法和请求头。此外，服务器端还可以加入 Access-Control-Max-Age Header，允许浏览器在指定时间内，无需再发送预检请求进行协商，直接用本次协商结果即可。浏览器根据 OPTIONS 请求返回的结果来决定是否继续发送真实的请求进行跨域资源访问。这个过程对真实请求的调用者来说是透明的。

XMLHttpRequest 支持通过 withCredentials 属性实现在跨域请求携带身份信息(Credential，例如 Cookie 或者 HTTP 认证信息)。浏览器将携带 Cookie Header 的请求发送到服务器端后，如果服务器没有响应 Access-Control-Allow-Credentials Header，那么浏览器会忽略掉这次响应。

参考：

http://fengchangjian.com/?tag=preflighted

http://www.freebuf.com/articles/web/18493.html

测试环境：

域 1：本地搭建 wamp(window+apache+mysql+php),域名：127.0.0.1,文件名：cors.html

域 2：百度云开发机,服务器是 lighttpd,域名不方便透露,以 www.a.com 代替,文件名：test.php

测试详情：

你可能写的初始代码如下：

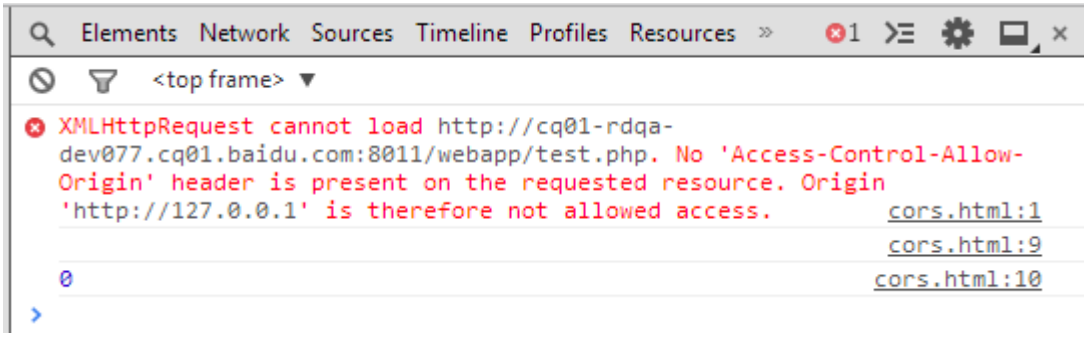
cors.html：

```html
<!DOCTYPE>
<html>
    <body>
        <script type='text/javascript'>
            var url = '[redacted]/test.php';
            var xhr = new XMLHttpRequest();
            xhr.onreadystatechange = function() {
                if(4 === xhr.readyState) {
                    console.log(xhr.responseText);
                    console.log(xhr.status);
                }
            }
            xhr.open('get', url, true);
            xhr.send(null);
        </script>
    </body>
</html>
```

test.php:

```php
<?php
echo 'dddddd';
?>
```

结果：



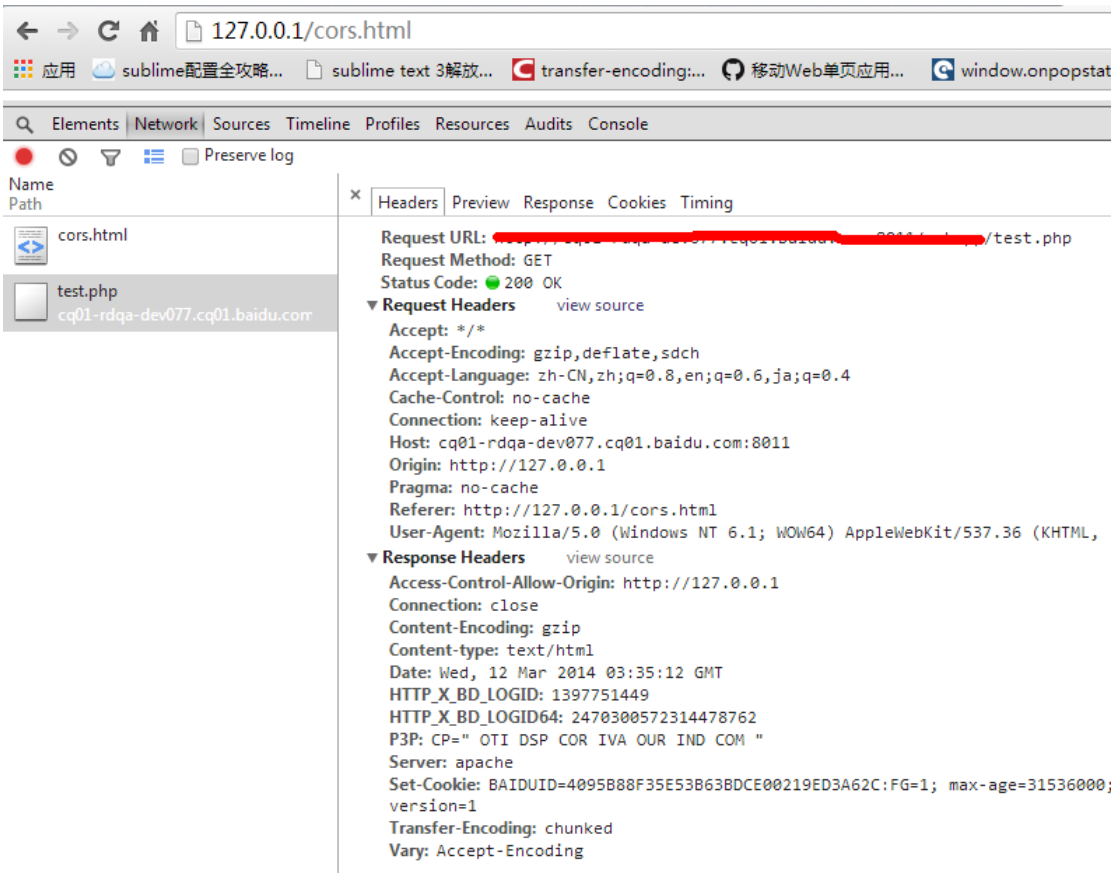
说明：

1. 上述请求为简单跨域请求，服务器端没有配置 Access-Control-Allow-Origin 导致出错

更改 test.php 文件为：

```php
<?php
header('Access-Control-Allow-Origin:http://127.0.0.1');
echo 'dddddd';
?>
```

请求变为：



结果：成功，但是没有带 cookie，若要带上 cookie，需要修改文件如下：

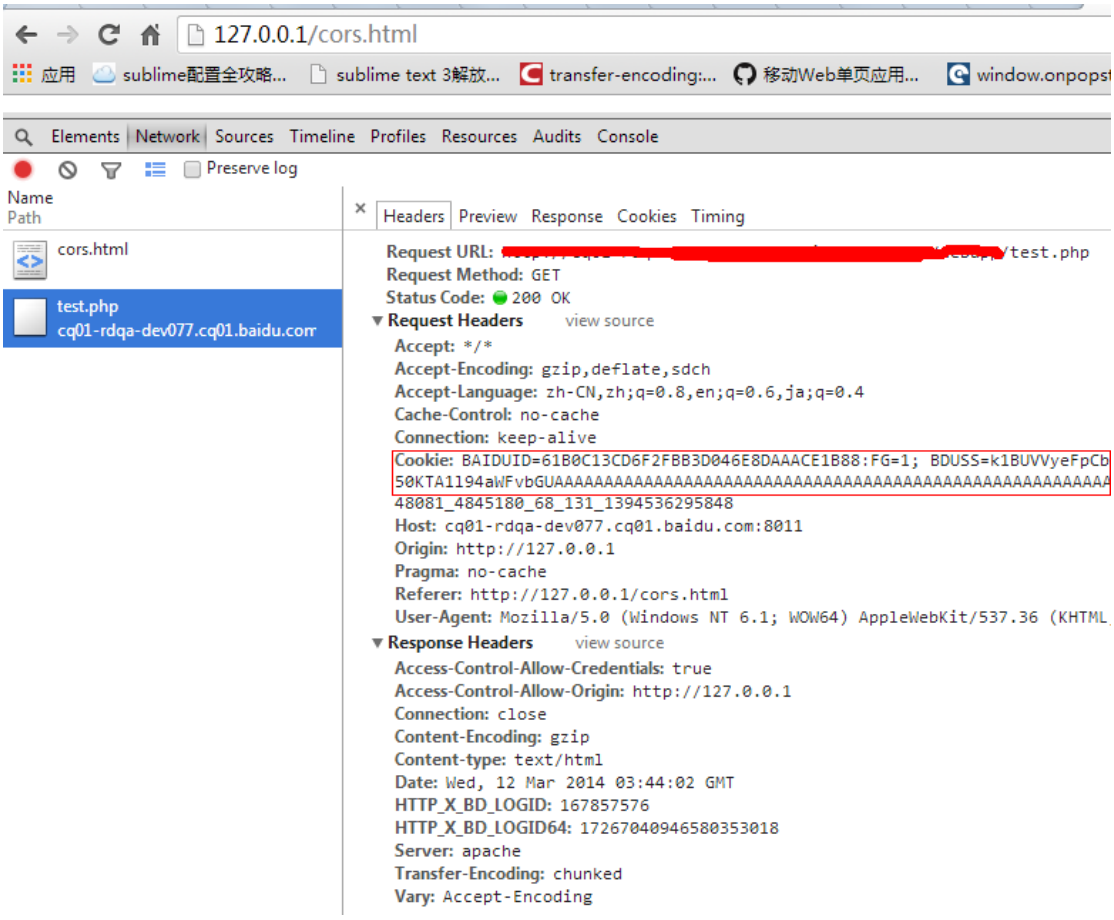
cors.html 改为：

```
<!DOCTYPE>
<html>
    <body>
        <script type='text/javascript'>
            var url = '/test.php';
            var xhr = new XMLHttpRequest();
            xhr.onreadystatechange = function() {
                if(4 === xhr.readyState) {
                    console.log(xhr.responseText);
                    console.log(xhr.status);
                }
            }
            xhr.open('get', url, true);
            xhr.withCredentials = true;
            xhr.send(null);
        </script>
    </body>
</html>
```

Test.php 必须相应修改：

```
<?php
header('Access-Control-Allow-Origin:http://127.0.0.1');
header('Access-Control-Allow-Credentials:true');
echo 'dddddd';
?>
```

结果：



特别说明：如果配置了 xhr.credentials 为 true，此时基于安全考虑,test.php 中的 header Access-Control-Allow-Origin 不能配置为*

非简单跨域请求(Preflighted Request):

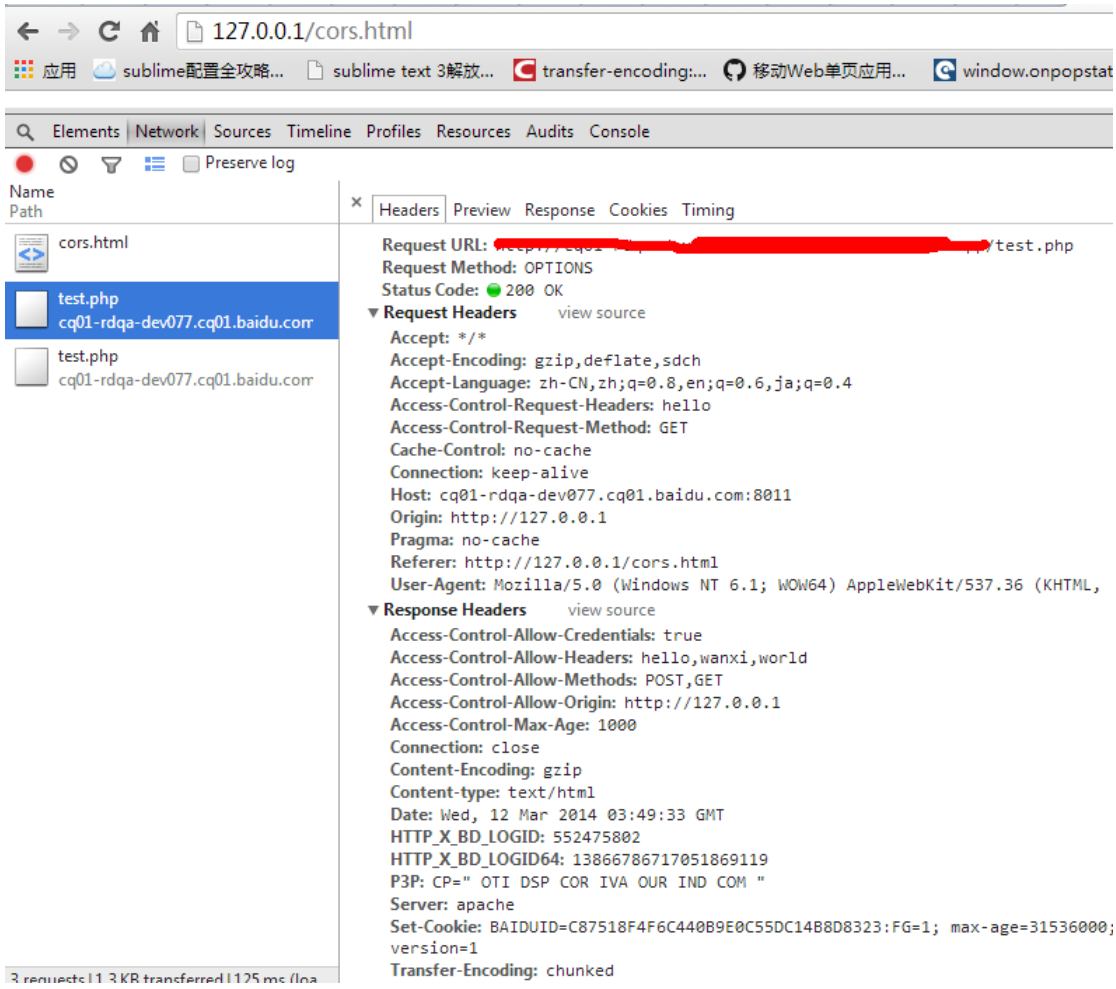
cors.html 改为（添加了自定义头部，不满足简单跨域请求的第二个条件了）：

```
<!DOCTYPE>
<html>
    <body>
        <script type='text/javascript'>
            var url = '[redacted]/test.php';
            var xhr = new XMLHttpRequest();
            xhr.onreadystatechange = function() {
                if(4 === xhr.readyState) {
                    console.log(xhr.responseText);
                    console.log(xhr.status);
                }
            }
            xhr.open('get', url, true);
            xhr.withCredentials = true;
            xhr.setRequestHeader('hello', 'wanxi');
            xhr.send(null);
        </script>
    </body>
</html>
```

test.php 改为：

```php
<?php
header('Access-Control-Allow-Origin:http://127.0.0.1');
header('Access-Control-Allow-Credentials:true');
header('Access-Control-Allow-Methods:POST,GET');
header('Access-Control-Allow-Headers:hello,wanxi,world');
header('Access-Control-Max-Age:1000');
echo 'dddddd';
?>
```

结果：



结论：

test.php 发了两次请求，第一次是 preflighted request,request method 为 OPTIONS,浏览器发现自定义了头部,所以需要先发个请求去服务器端验证下这个自定义头部服务器是否允许，对比返回的结果中 Access-Control-Allow-Headers 发现可以，再发真正 test.php 请求。

2.jsonp,图像 ping

原理：浏览器请求<img />和<script>标签中外链内容时不做扩展名判断，即<script>中 src 是.php 文件也照样请求（亲自测试过），<img>也一样，对返回的数据直接当 js 或图片解析。

3.iframe

限制：主域必须相同，即必须形如 map.baidu.com 和 baike.baidu.com,主域都是 baidu.com