jsonp

原理

使用XHR不能跨域，前面实验看到了，除非做跨域的处理或者preflight等。

但是`<img><link><script>`等标签可以使用**不同域**的文件。

VsCode中使用Live Server启动一个站点测试

index.html

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <title>jsonp test</title>
</head>
<body>
    <div id="root">mageud.com</div>
</body>
</html>
<script src="index.js"></script>
```

index.js

```
console.log(document.getElementById('root').innerText = 'mageud.net');
```

启动Live Server，访问http://127.0.0.1:5500/index.html

修改`<script src="http://localhost:5500/index.js"></script>`，访问依然成功运行了，说明可以跨域。

使用Flask或者Django等，构建一个视图函数，该视图函数返回

```
from flask import Flask
from flask import request, Response


app = Flask('web')

@app.route('/jsonp')
def jsonp():
    print(request.query_string)
    print(request.args)
    cb = request.args.get('cb')
    if cb:
        content = "{}({{a:1000, b:2000}})".format(cb) # 带数据回去
    else:
        content = ''

    response = Response(content)
```

```
    response.content_type = 'application/javascript'
    return response
```

```
from webapp import app


if __name__ == '__main__':
    app.run(debug=True)
```

```
<script src="http://localhost:5000/jsonp?cb=jsonpCallback"></script>
<script>
    function jsonpCallback(data) {
        console.log(data, '+++');
    }
</script>
```

这样看似访问是一个js，实际上是动态的视图函数。实现了跨域访问，还带回了客户端脚本。

为了便于客户端使用数据，逐渐形成了一种非正式传输协议，人们把它称作JSONP，该协议的一个要点就是允许用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹住JSON数据，这样客户端就可以随意定制自己的函数来自动处理返回数据了。

Ajax访问jsonp

能否把上例中的约定的回调函数jsonpCallback也省略了？jQuery的Ajax就提供了这样内建的处理方式

- dataType:'jsonp'，使用JSONP处理返回数据
- jsonp:'cb'，指定使用jsonp查询字符串的参数名，默认callback
- jsonpCallback:'jsonpcallback'，`cb=xxx`替代xxx的函数名，如果不指定将使用随机函数名

```
<!DOCTYPE html>
<html lang="zh-CN">

<head>
    <meta charset="UTF-8">
    <title>jsonp test</title>
    <script src="./jquery-3.6.0.min.js"></script>
</head>

<body>
    <div id="root">mageud.com</div>
    <button id="testJsonp">Jsonp测试</button>
</body>

</html>
<script>
    // function jsonpCallback(data) {
    //     console.log(data, '+++');
    // }
    $(function () {
        $('#testJsonp').click(function () {
            $.ajax({
```

```
                type: 'GET',
                dataType:'jsonp',
                jsonp:'cb',
                // jsonpCallback:'jsonpcallback', //指定返回后调用的函数名
                url: 'http://localhost:5000/jsonp',
                success:function(data) {
                    console.log(data, '!!!!!!');
                }
            })
        })

    })

</script>
```

注意：Jsonp**只能使用GET方法**，因为它内部使用了script标签的src发起http请求，这不能是POST方法的请求。

Ajax使用XHR组件通信，出于安全，一般要求同源策略。

JSONP实际上是网页内部动态增加了script标签，并利用了使用src引用静态资源时不受跨域限制的机制。事先写好一个回调函数放在网页中，并在发起HTTP请求时，将回调函数名告知服务器端。服务器端使用这个回调函数名，按照JavaScript语法把数据放在回调函数中。然后将这些内容作为脚本的内容返回浏览器端，浏览器端就可以运行该脚本。jQuery提供了JSONP语法糖，简化了操作。