

webp 图片格式, tencent: lncp

网速慢, 请求页面时间长

⇒ 如何优化, 减少页面加载时间

请求方式

1. 地址栏

2. 指定有 src 或 href 的元素

<a> <link>

3. 表单提交

以前无法对网络
进行控制

以前刷新页面才能发送请求

利用 JS 发送请求

AJAX 出现

1. GOOGLE SUGGEST

2. GMAIL

AJAX → 浏览器提供的 API, 通过代码控制请求
和响应

网速影响

页面复杂度

network-online

-Custom-adob

自定义网速

network

XHR, AJAX 请求

AJAX (Asynchronous JavaScript And XML) 也是描述数据结构
 各类浏览器 → 打开浏览器输入网址
 → 等待结果

`<student>`
`<name>李三</name>`
`<age>23</age>`
`</student>`

```
var xhr = new XMLHttpRequest(); // 相当于打开浏览器
xhr.open('GET', 'http://day-11.io/time.php'); // 相当于输入网址
xhr.send(); // 相当于提交请求，发送请求

// var response = xhr.send()
// 响应需要时
// 不知道服务器返回何时，时间不确定
// 返回数据，使用事件 因此形式
// 的形式接受数据

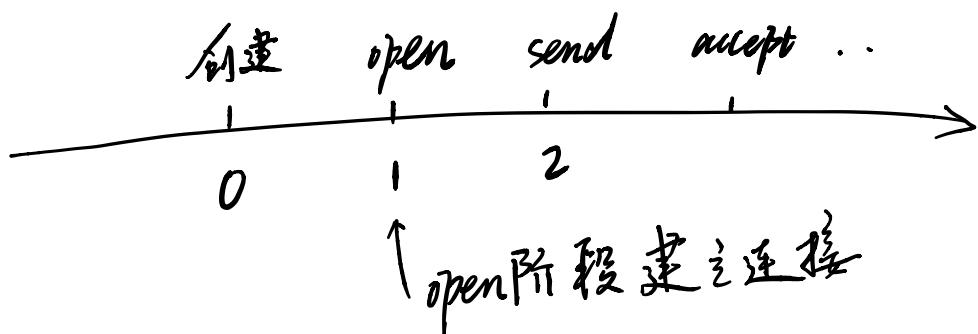
xhr.onreadystatechange = function() {
    console.log(1);
}

```

涉及到 AJAX 的“不能”
 以文件的格式访问

不请求整个页面，而是只获取一部分格式，传输数据有 JSON, XML.

状态改变就触发
 这里显示3次



```
xhr.onreadystatechange = function() {
    console.log(this.readyState);
}
```

这里使用 this
若使用 xhr. 还会不局
部作用域查找

这里显示 为什么没显示 - 1. (事件注册顺序)
1. 2. 3. 4 分别表示什么?

```
xhr.onreadystatechange = function() {
    if (this.readyState != 4) return;
    console.log(this.responseText); ~ 只为4的时候
}
```

执行.

理解的方式记忆.
Ready State . 为什么?

```
var xhr = new XMLHttpRequest();
xhr.open('GET', '～.php');
xhr.send();
xhr.addEventListener('readystatechange', function() {
    console.log(this.readyState);
})
```

0: 初始化代理对象

1: open方法已调用，建立与服务端特定端口连接

2: 接受到响应报文的响应头

3. 正在下载响应报文的响应体

4. 获取得到完整的响应报文
（获取内容不完整）

下载文件先

接受响应头

再慢慢接受体

获取响应头：

obj.getAllResponseHeaders

obj.getResponseHeader
(‘server’)

```
xhr.onload = function() {  
    console.log(this.readyState); //4.  
    console.log(responseText);  
}
```

↑ XMLHttpRequest 2.0 定义

```
var xhr = new XMLHttpRequest;
```

```
xhr.open('GET', 'test.php');
```

设置请求头

设置请求头

```
xhr.setRequestHeader('Foo', 'Bar');
```

设置请求行

```
xhr.send('key1=value1&key2=value2');
```

urlencode

输出

Content-Type 应该与请求体编码

一致。

Request Payload

multipart/form-data .

2.1 to form-data .

`xhr.status` → 200
`xhr.statusText` → 'OK' `xhr.responseText`
`xhr.getResponseHeader('Content-Type')`
`xhr.getAllResponseHeader();`

http约定报文的内容都是字符串
传递给客户端有结构的数据。 返回json格式的字符串
一般是json格式

提供能力。
有输入，有输出

对于返回数据的地址都称之为API(接口)

登陆时点击按钮使用Ajax请求

1. JS中字符串直引变量

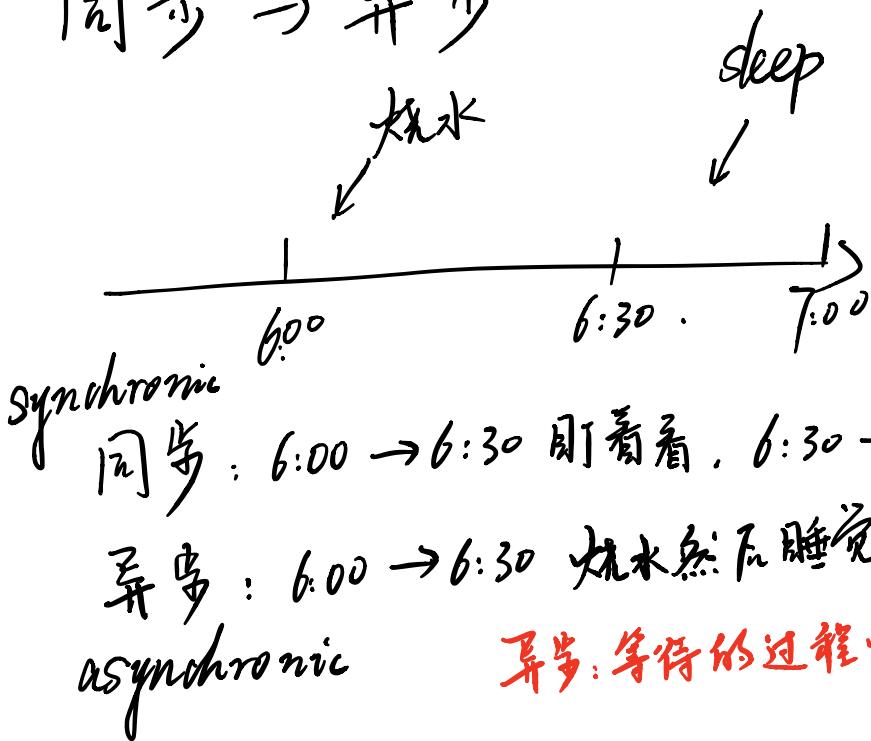
`xhr.send(`username=${username}&password=`

2. urlencode参数必须格式化。 `${password}`);`

请求头

3. 先测服务端，后测客户端

同步与异步



```
console.time('asyn');
var xhr = new XMLHttpRequest();
xhr.open('GET', 'time.php', true);
xhr.send();
console.timeEnd('asyn');
```

```
console.time('sync');
var xhr = new XMLHttpRequest();
xhr.open('GET', 'time.php', false);
xhr.send();
console.timeEnd('sync');
```

Asyn won't wait in "send();"

while Sync will waiting in "send();"

↑
0.29ms
Asyn
No waiting

↑
2047ms
Sync
Waiting

asyn
布尔值，置为 true
Output

JS 把它扔了。

console.time('abc');
↓ Output
Interval
console.timeEnd('abc');

(deprecation) !

如何获取 XML 数据。

PHP 文件返回类型默认为 HTML (Content-Type: text/html)

XHR.responseXML专门用于获取服务器端返回的 XML 数据。

注意客户端要使用 Content-Type: text/html 正确解析数据

console.log(this.responseText.documentElement.children[0].innerHTML)

console.log(this.responseText.documentElement.getElementsByTagName('name')[0])

从服务端获取来的数据类型，可以使用

模板引擎

XMLHttpRequest 在老版本浏览器中有兼容问题

Response 和 ResponseText 的区别

this.response 会根据 responseType 的变化而变化 ↴ 不好

this.responseText 总会翻回请求体的字符串形式

兼容性.

渲染模板地址

script 标签的特点

1. innerhtml 不会显示在页面

2. 类型不为 text/javascript . 内容不会作为 JS 执行

为什么写在 script 标签中？

Ajax 的封装

封装的套路：

1. 写完整的用例

2. 写空函数不带参数

3. 根据使用过程中的

需求写参数

注意 content-type

模块化封装

注意异步的返回值问题

可以使用回调函数

对于普遍使用的功能，通常都会有人封装

jQuery 的封装。

```
$.ajax(`./time.php`, {
    type: 'post', // method 方法
    success: function(res) {
        console.log(res);
    } // 回调函数
});
```

```
$.ajax({
    url: 'time.php',
    type: 'get',
    data: { id: 1, name: 'zs' },
    success: function(res) {
        console.log(res)
    }
})
```

提交到服务器端的
数据。

url 编码。

res 会根据 content-type
自动转换为对象

```

$.ajax({
    url: 'time.php',
    type: 'get',
    data: { id: 1, name: 'zs' },
    dataType: 'json', ← 设置响应体的类型
    success: function(res) {
        console.log(res)
    }
})

```

客户端解读响应体的方式

Success: 状态码为 200 时执行

```

error: function() {};
complete: function() {};
beforeSend: function() {} // 在状态码为 1 之前

```

```

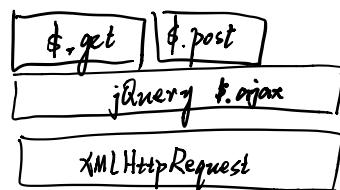
$.get('time.php', { id: 1 }, function(res) {
    console.log(res);
})

$.post('time.php', { id: 1 }, function(res) {
    console.log(res);
})

$.getJSON - $.postJSON

```

写得最多。



```

$(function ($) {
    $('[list-group-item]').on('click', function() {
        var url = $(this).attr('href');
        $('#main').load(url + '#main > *');
        return false;
    });
}

```

将全局变量转为局部变量
直接在内部作用域链
查找变量
load() 只加载页面的某一部分

封装开始结束的全局事件

```

$(document).ajaxStart(function() {
    $('.loading').fadeIn();
});
$(document).ajaxStop(function() {
    $('.loading').fadeOut();
});

```

进度条的库 NProgress.js

进度条是按照一定算法计算的
并非实际的进度

同源策略

不同源地址之间，默认不能相互发送 ajax 请求

请求不同源就是跨域请求

发送请求

img —— 都是跨域请求

link —— 但是无法获取内容

script

iframe

link 用于引入文件

rel 指明文件与引入文件的关系

通过 script 标签发送跨域请求

通过 script 标签可以接受服务器的数据

服务器将数据编写进 js，然后返回 js 文件

客户端会执行该文件，然后便可获取数据

(利用变量获取存在异步问题，请求后未接受到
就要调用)

客户端

约定

服务器

声明函数 foo() ←→ 调用函数 foo(data)

AJAX (CORS)

```
$.get('~/php', {}, function(res){  
    console.log(res);  
})
```

} 正常无法请求

服务端加入

```
header('Access-Control-Allow-Origin: *');
```

设置允许请求的
原

JSONP — JSON with Padding

写函数，然后在引入的地方中引用

通过 script 标签引入 PHP 文件。PHP 文件生成一个 js 文件。
带有函数调用，设置 content-type