

jQuery



AJAX, JSONP

Aleksander Lamża
ZKSB · Instytut Informatyki
Uniwersytet Śląski w Katowicach

aleksander.lamza@us.edu.pl

- Czym jest AJAX?
- XMLHttpRequest
- jQuery.ajax()
- Same origin policy i komunikacja Cross-Domain
- JSONP

Czym jest AJAX?

AJAX
to skrót od
Asynchronous JavaScript And XML

asynchroniczny

JavaScript

i

XML

czyli nie synchroniczny ;)

to wiadomo...

na szczęście nie musimy
korzystać z XML-a.
Równie dobrze może to być JSON
(można się spotkać ze skrótem AJAJ).

Tak naprawdę można jednak za jego pomocą
realizować też komunikację synchroniczną.

XMLHttpRequest

Asynchroniczną komunikację realizuje się za pomocą obiektu

XMLHttpRequest

Obiekt tworzymy w taki sposób:

```
var xhr = new XMLHttpRequest();
```

Ponieważ z założenia będzie to proces asynchroniczny, ustawiamy funkcję wywoływaną w momencie wykonania operacji (listener):

```
var xhr = new XMLHttpRequest();  
xhr.onload = xhrListener;
```

```
function xhrListener() {  
    console.log(this.responseText);  
}
```



odpowieź wyświetlimy w konsoli;
this w tym przypadku to obiekt XMLHttpRequest


XMLHttpRequest

Teraz musimy określić typ żądania, adres URL i to, czy ma to być żądanie asynchroniczne, czy synchroniczne.

Służy do tego metoda **open(method, url, async)**:

```
xhr.open("get", "resource", true);
```

Kiedy już wszystko ustawimy, trzeba tylko wysłać żądanie co robimy za pomocą metody **send()**.



send() ma tak naprawdę jeden parametr, który służy do przesyłania danych, jeżeli korzystamy z metody POST

Kompletny kod wygląda więc tak:

```
var xhr = new XMLHttpRequest();  
xhr.onload = xhrListener;  
xhr.open("get", "resource", true);  
xhr.send();
```

XMLHttpRequest

A oto efekt działania kodu:

The screenshot shows the Chrome DevTools Network tab with the 'Headers' sub-tab selected. The left sidebar lists three resources: 'test_xhr/' (a folder icon), 'resource' (a document icon), and 'icon_19.png' (a PNG icon). The 'resource' entry is selected. The main pane displays the request details for 'http://localhost/html5/ajax/test_xhr/resource'. The status is 200 OK. The 'Request Headers' section is expanded, showing various headers including Accept, Accept-Charset, Accept-Encoding, Accept-Language, Cache-Control, Connection, Cookie, Host, Referer, User-Agent, and ScreenResolution. The 'Response Headers' section is also expanded, showing headers like Accept-Ranges, Connection, Content-Length, Content-Type, Date, ETag, Keep-Alive, Last-Modified, and Server.

Name	Path
test_xhr/	/html5/ajax
resource	/html5/ajax/test_xhr
icon_19.png	kkelicaakdanhinjdeammilcg

Request Headers

- Request URL: http://localhost/html5/ajax/test_xhr/resource
- Request Method: GET
- Status Code: 200 OK
- Accept: */*
- Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.3
- Accept-Encoding: gzip,deflate,sdch
- Accept-Language: pl-PL,pl;q=0.8,en-US;q=0.6,en;q=0.4
- Cache-Control: max-age=0
- Connection: keep-alive
- Cookie: JSESSIONID.1e85f8cc=62aa5d1c9e603601eac0537c8ad11a98; org.cups.sid=62f102350d6be5df0e15b6494d4954e7; PHPSESSID=mpc3dsk9ro255gaabs40miubm6; JSESSIONID.09906359=3ab9193cd8c6802b9377af27b060ddfd;
- ScreenResolution=1920x1200
- Host: localhost
- Referer: http://localhost/html5/ajax/test_xhr/
- User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.31 (KHTML, like Gecko) Chrome/26.0.1410.63 Safari/537.31

Response Headers

- Accept-Ranges: bytes
- Connection: Keep-Alive
- Content-Length: 24
- Content-Type: application/json
- Date: Sun, 21 Apr 2013 21:54:51 GMT
- ETag: "7a0937-18-4dae5c5b148bb"
- Keep-Alive: timeout=5, max=99
- Last-Modified: Sun, 21 Apr 2013 21:38:25 GMT
- Server: Apache/2.2.22 (Ubuntu)

3 requests | 877 B transferred | 6

This screenshot shows the same Chrome DevTools Network tab, but with the 'Response' sub-tab selected. The left sidebar is identical to the previous screenshot. The main pane displays the response body for the selected resource, which is a JSON object: {msg: "DZIAŁA!!!"}. The response is shown in a syntax-highlighted format.

Name	Path
test_xhr/	/html5/ajax
resource	/html5/ajax/test_xhr
icon_19.png	kkelicaakdanhinjdeammilcg

Response

```
1 {  
2   msg: "DZIAŁA!!!"  
3 }  
4
```

3 requests | 877 B transferred | 6

jQuery.ajax()

Jak pewnie pamiętacie, biblioteka jQuery miała nam we wszystkim pomagać.

Nie inaczej jest w przypadku AJAX-a. Mamy do dyspozycji metodę...

ajax()

Dzięki niej kod:

```
var xhr = new XMLHttpRequest();  
xhr.onload = xhrListener;  
xhr.open("get", "resource", true);  
xhr.send();
```

możemy zastąpić czymś takim:

```
$.ajax({  
  url: "resource"  
}).done(function(data) {  
  console.log(data);  
});
```

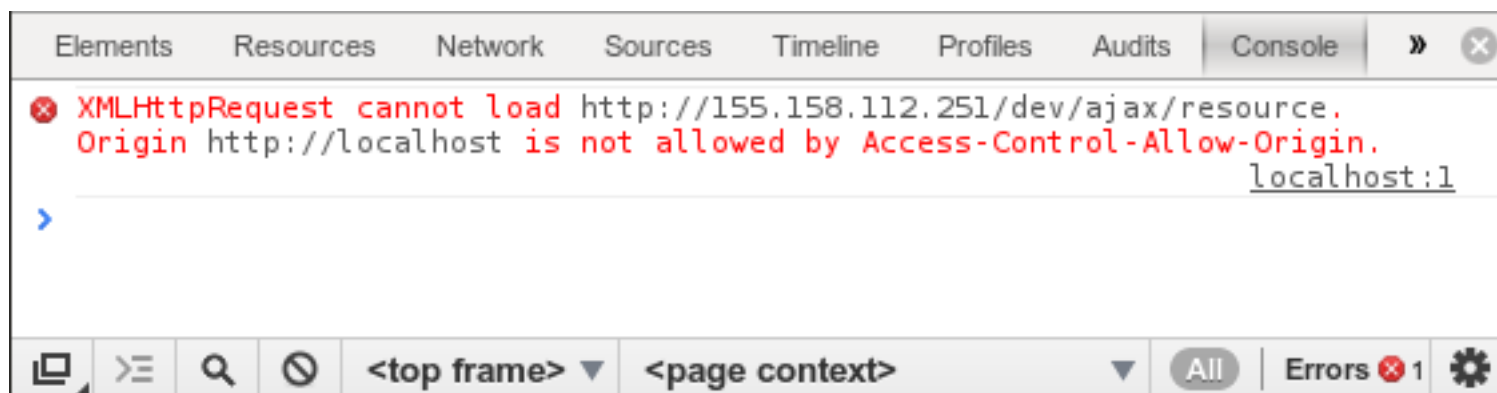


Koniecznie zajrzyjcie do dokumentacji metody `ajax()` na stronie jQuery!

Same origin policy i komunikacja Cross-Domain


No dobrze, a co się stanie, gdy zgłoszę żądanie do innego serwera (innej domeny)?

```
$.ajax({  
  url: "http://155.158.112.251/dev/ajax/resource"  
}).done(function(data) {  
  console.log(data);  
});
```



Same origin policy i komunikacja Cross-Domain

Aby można było otrzymać odpowiedź od innego serwera, musi on wysłać nagłówek `Access-Control-Allow-Origin`.



Do niedawna sprawa była bardziej skomplikowana, ale na szczęście w **XMLHttpRequest 2** istnieje możliwość tego typu komunikacji. Trzeba jednak korzystać z przeglądarki, która obsługuje CORS (Cross-Origin Resource Sharing), a odpowiedź serwera musi posiadać ten nagłówek.

Przykład kodu PHP, który dodaje ten nagłówek:

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



dozwolona jest dowolna domena

Inną możliwością obejścia problemu jest zastosowanie techniki

JSONP

czyli JSON with Padding

W skrócie wygląda to tak:

1. Do dokumentu dodajemy dynamicznie element script.
2. Element ten pobiera skrypt, który zawiera dane.
3. Dane są opakowane w funkcję zwrotną (tzw. callback).
4. Funkcja zwrotna jest wywoływana bezpośrednio po załadowaniu skryptu, dzięki czemu uzyskujemy dostęp do danych (są parametrami funkcji).

Jak to wygląda w praktyce?

Najpierw część serwerowa (PHP):

```
<?php
    //przygotowuję nazwę funkcji zwrotnej
    $callback = 'callback';
    if (isset($_GET['callback'])) {
        $callback = $_GET['callback'];
    }

    //tworzę dane odpowiedzi
    $res = array(
        'msg' => 'DZIAŁA!'
    );

    //odsyłam odpowiedź
    echo ($callback."(".json_encode($res).")");
?>
```

Część kliencka w czystym JavaScriptcie:

```
//tworzę i konfiguruję element script
var script = document.createElement("script");
script.setAttribute("src", "resource.jsonp.php");

//dodaję element script do dokumentu...
var head = document.head;
head.appendChild(script);
//...i od razu go usuwam
head.removeChild(script);
```

W kodzie JS muszę też umieścić funkcję zwrotną:

```
function callback(res) {
  console.log(res);
}
```

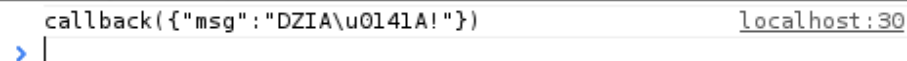
A teraz skorzystamy z metody `jQuery.ajax()`:

```
$.ajax({  
  url: "resource.jsonp.php",  
  dataType: "jsonp"  
}).done(function(data) {  
  console.log(data);  
});
```



A screenshot of a browser's developer console. The top bar shows the object `Object {msg: "DZIAŁA!"}` and the location `localhost:31`. Below the bar, a blue prompt character `>` is followed by a vertical bar `|`, indicating the console is ready for input.

A tak by to wyglądało, gdybyśmy nie określili typu `dataType: "jsonp"`:



A screenshot of a browser's developer console. The top bar shows the error message `callback({"msg":"DZIA\u0141A!"})` and the location `localhost:30`. Below the bar, a blue prompt character `>` is followed by a vertical bar `|`, indicating the console is ready for input.

Mini projekt prezentujący zastosowania AJAX-a



Sosnowiec
Bezchmurnie / słonecznie.

Temperatura	5°C
Ciśnienie	1018 hPa
Wilgotność	65%
Prędkość wiatru	13 km/h