AJAX
“Asynchronous JavaScript and XML”

Alexandru Munteanu

Universitatea de Vest din Timișoara
Facultatea de Matematică și Informatică
Elemente de Web Design

2022

AJAX

Ce este AJAX? Un set de tehnologii web, cu ajutorul cărora putem trimite/încărca date intr-o pagină web, în mod asincron.

De ce? Cu ajutorul AJAX, putem realiza o comunicare între client și server, într-un mod favorabil.

Cum îl putem folosi? Nativ, folosind JavaScript, sau jQuery.

Dar înainte, va trebuie să definim niște noțiuni.

Structura cursului

- Modelul client-server
- Stiva ISO - OSI & TCP/IP
- Protocolul HTTP
- Programarea asincronă
- XML și JSON

Modelul client-server

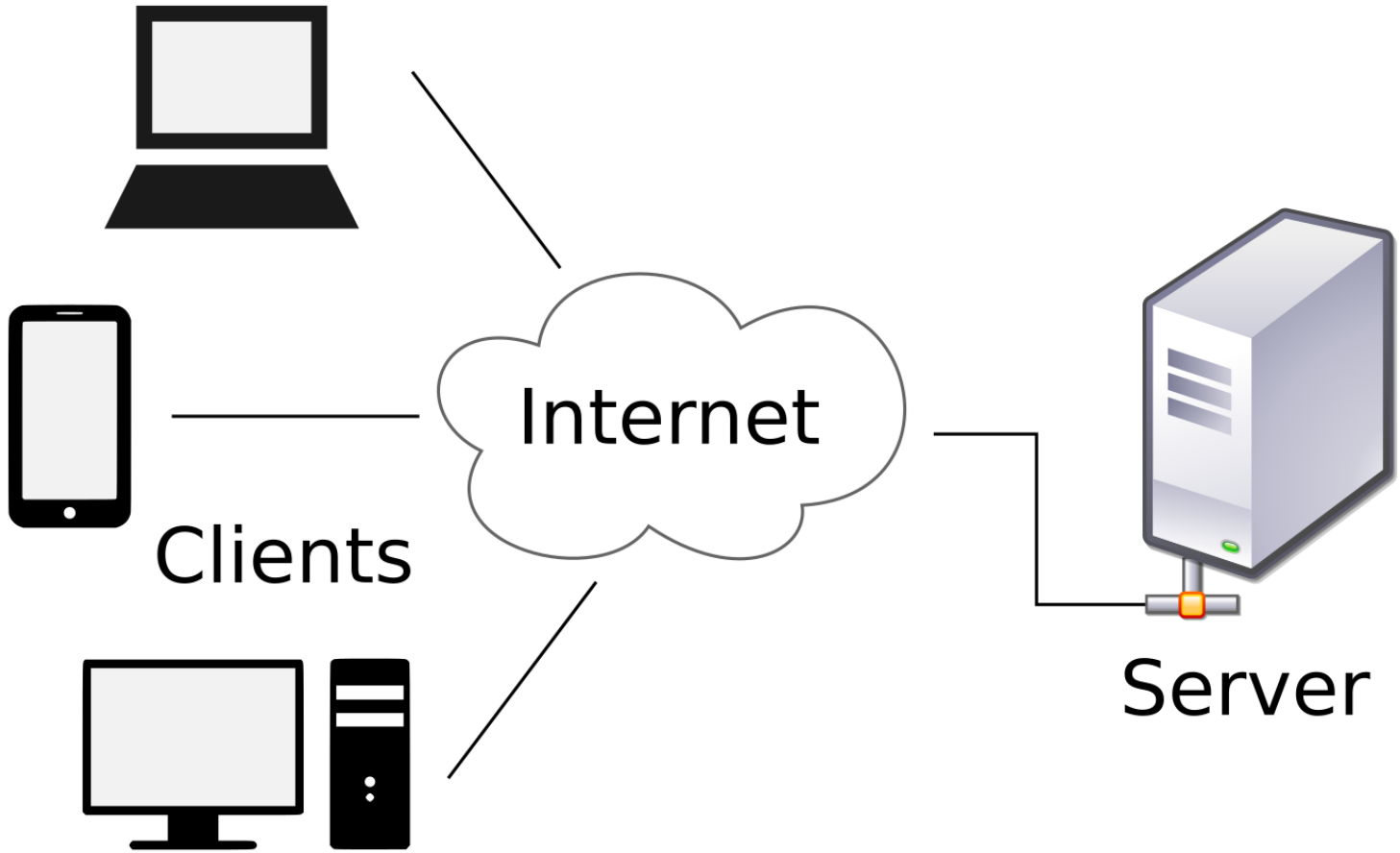

Figure: Arhitectura client-server
https://en.wikipedia.org/wiki/Client-server_model

Exemple[1]

Client	**Server**
Mozilla Firefox	Apache HTTP
Google Chrome	NGINX
cURL	Apache Tomcat
wget	lighttpd

[1] web.stanford.edu/InternetWhitepaper

Stiva ISO - OSI (Open Systems Interconnection)

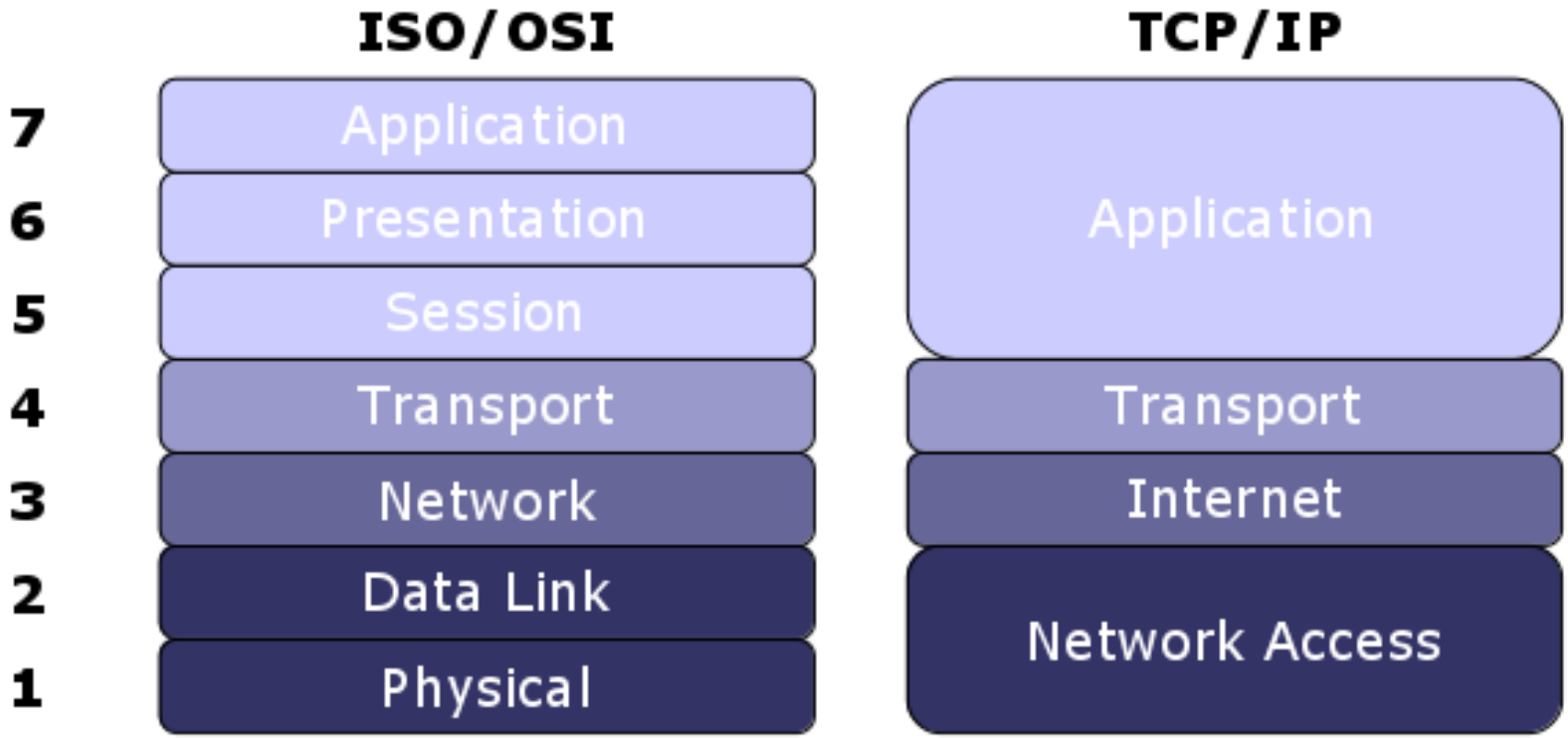

Figure: Stiva OSI & stiva TCP/IP
networkengineering.stackexchange.com

Protocolul HTTP - HyperText Transfer Protocol

- Este un protocol request-response (cerere - răspuns).
- Determină modul în care informațiile sunt transmise în WWW.
- La baza oricărui API web.
- HTTPS - Versiune a HTTP care se foloseste de SSL/TLS pentru a cripta datele ce sunt transmise între client și server.
- Ultima versiune: HTTP/2.0 - 2015.

`https://tools.ietf.org/html/rfc7540`

Request: La ce nivel din stiva ISO - OSI credeți că se află protocolul HTTP?

Request: La ce nivel din stiva ISO - OSI credeți că se află protocolul HTTP?
Response: La nivelul aplicație.

Un exemplu de cerere HTTP

```
alex@beast ~ % curl -v https://www.info.uvt.ro/
*   Trying 85.120.206.130:443...
* Connected to www.info.uvt.ro (85.120.206.130) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*  CAfile: /etc/ssl/certs/ca-certificates.crt
*  CApath: none
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES256-GCM-SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: CN=info.uvt.ro
*  start date: Feb 22 12:29:00 2021 GMT
*  expire date: May 23 12:29:00 2021 GMT
*  subjectAltName: host "www.info.uvt.ro" matched cert's "www.info.uvt.ro"
*  issuer: C=US; O=Let's Encrypt; CN=R3
*  SSL certificate verify ok.
> GET / HTTP/1.1
> Host: www.info.uvt.ro
> User-Agent: curl/7.75.0
> Accept: */*
```

Un exemplu de răspuns HTTP

```
< HTTP/1.1 200 OK
< Date: Mon, 22 Mar 2021 14:05:09 GMT
< Server: Apache/2.4.18
< Link: <http://info.uvt.ro/wp-json/>; rel="https://api.w.org/", ...
< Set-Cookie: pll_language=ro; expires=Tue, 22-Mar-2022 14:05:10 GMT; Max-Age=31536000; path=/;
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html; charset=UTF-8
< X-HA-Timestamp: Mon, 22 Mar 2021 14:05:10 GMT
< X-HA-Frontend: http
< X-HA-Backend: info.uvt.ro
< X-HA-Request-Id: 947451001.1151755277.346013847.1826599631
< X-HA-Session-Id: 1822759947.2845893653.395082039.2445584033
< Set-Cookie: X-HA-Session-Id=1822759947.2845893653.395082039.2445584033; Path=/; Max-Age=2419200
< Content-Security-Policy: upgrade-insecure-requests
< Keep-Alive: timeout=60, max=1000
< X-HA-HTTP-Action: GET://www.info.uvt.ro/?
<
<!DOCTYPE html>
<html lang="ro-RO">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width">
.
.
.
```

Întrebare: Care este clientul folosit pentru efectuarea cererilor HTTP din exemplul anterior?

Întrebare: Care este clientul folosit pentru efectuarea cererilor HTTP din exemplul anterior?

Răspuns: Informația se află in header-ul User-Agent: curl/7.75.0

Scurtă taxonomie a cererii

```
> GET / HTTP/1.1
> Host: www.info.uvt.ro
> User-Agent: curl/7.75.0
> Accept: */*
```

- Verbul HTTP: GET
- Pagina catre care s-a facut cererea: / (care reprezintă rădăcina site-ului, implicit index.html)
- Protocolul: HTTP/1.1
- Adresa: www.info.uvt.ro
- User-Agent: cURL
- Accept: orice mimetype (i.e. orice fel de fișier)

Dar ce este un verb HTTP?

Numele unei metode, care informează serverul care sunt intențiile cererii.

Metoda	**Folosită pentru**	**Observații**
GET	citire	
POST	creare	
PUT	update	prin înlocuire
PATCH	update	prin modificare
DELETE	ștergere	

Coduri de retur HTTP

Orice răspuns HTTP returnează un cod de retur.

- 1.x.x - Informațional
  - 100 - Continue - serverul a primit cererea, așteapt instrucțiuni adiționale
  - 101 - Switching Protocols - cerere pentru schimbarea protocolului acceptată
  - 102 - Processing - serverul proceseaza cererea
- 2.x.x - Success
  - 200 - OK - raspuns standard pentru o cerere HTTP
  - 201 - Created - o cerere a fost îndeplinita
  - 202 - Accepted - cererea a fost inregistrata, dar nu îndeplinită

- 3.x.x - Redirectare
  - 301 - Moved Permanently
  - 304 - Not Modified
- 4.x.x - Eroare client
  - 400 - Bad Request
  - 401 - Not Authorized
  - 403 - Forbidden
  - 404 - Not Found
- 5.x.x - Eroare server
  - 500 - Internal Server Error
  - 501 - Not Implemented
  - 505 - HTTP Version Not Supported

! Important: Folosirea codurilor de retur într-un mod responsabil, astfel încât acestea să reflecte întradevar ceea ce se petrece în implementare.

Programarea asincronă

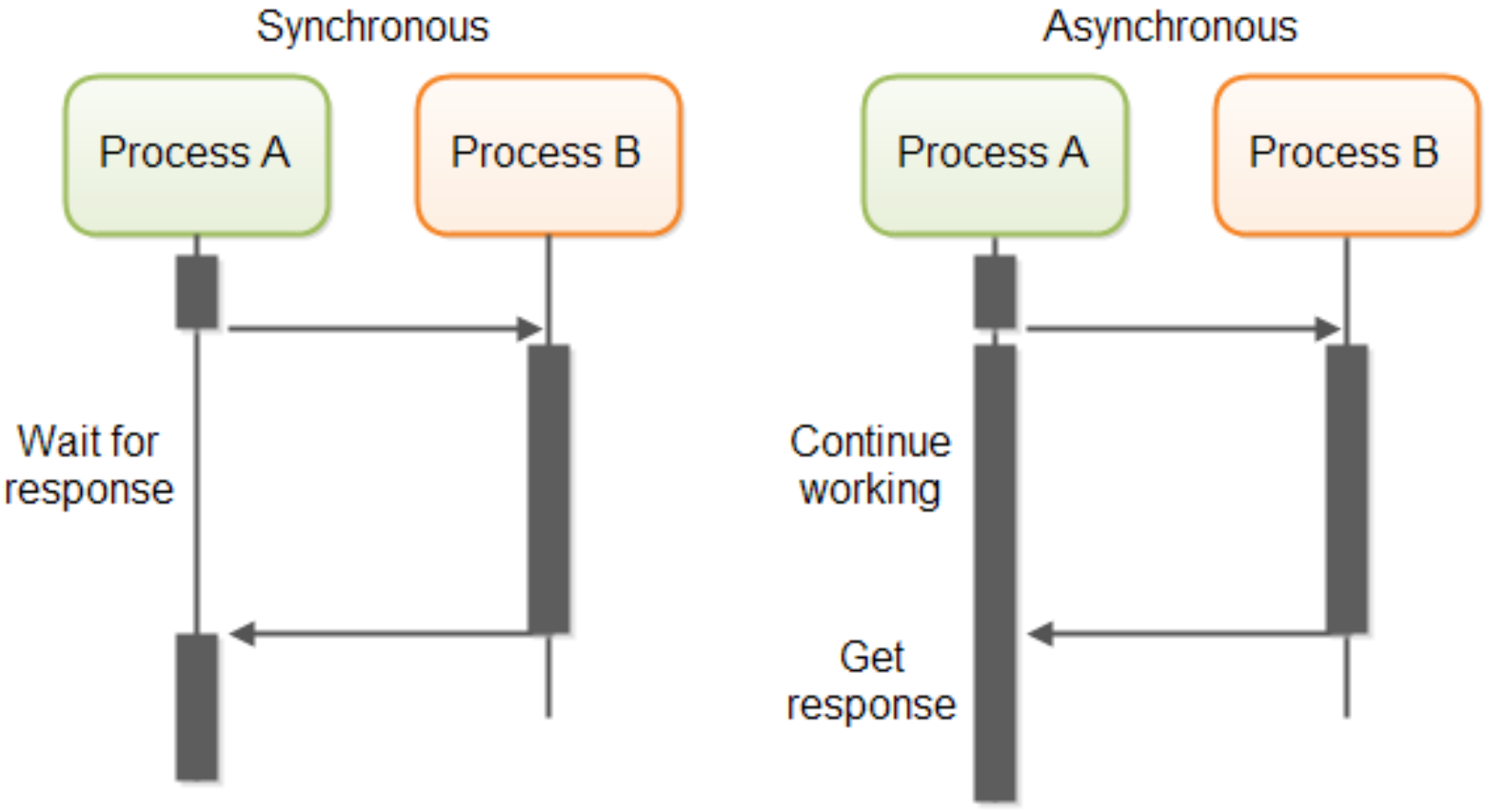

Figure: Programarea sincronă și cea asincronă

De ce asincron?[2]

Scăpăm de secțiuni/rutine de cod blocante, și redăm control browserului.

Mecanisme de programare asincronă în JavaScript:

- async()
- await ¡expr¿
- fetch()
- Promise

[2] https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Asynchronous/Concepts

XML - eXtensible Markup Language[3]

- Folosit pentru stocarea și transferul informațiilor.
- Implementare a SGML.
- Case sensitive.
- Support pentru namespace-uri.
- Structura și informațiile pot fi verificate folosind XML schemas.

[3] https://www.w3.org/TR/REC-xml/

Un fișier XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<breakfast_menu>
    <food>
        <name>Belgian Waffles</name>
        <price>$5.95</price>
        <description>
            Two of our famous Belgian Waffles with plenty of real maple syrup
        </description>
        <calories>650</calories>
    </food>
    <food>
        <name>Strawberry Belgian Waffles</name>
        <price>$7.95</price>
        <description>
            Light Belgian waffles covered with strawberries and whipped cream
        </description>
        <calories>900</calories>
    </food>
</breakfast_menu>
```

JSON - JavaScript Object Notation[4]

- Folosit pentru stocarea și transferul informațiilor.
- Implementare bazată pe standardul ECMA-262.
- Orice obiect JavaScript poate fi reprezentat ca JSON.

[4] https://www.json.org/json-en.html

Un fișier JSON

```
{
  "breakfast_menu": {
    "food": [
    {
      "name": "Belgian Waffles",
      "price": "$5.95",
      "description": "Two of our famous Belgian Waffles with plenty of real maple syrup",
      "calories": 650
    },
    {
      "name": "Strawberry Belgian Waffles",
      "price": "$7.95",
      "description": "Light Belgian waffles covered with strawberries and whipped cream",
      "calories": 900
    }
    ]
  }
}
```

XML vs JSON

- JSON nu are nevoie de deschidere/inchidere de tag-uri.
- JSON este mai scurt.
- Informațiile dintr-un fișier JSON pot fi parsate mai rapid.
- Validarea in JSON este mai dificilă.

AJAX

AJAX se foloseste de programarea asincronă și de metode de parsare a documentelor XML sau JSON pentru trimiterea/obținerea datelor, folosind protocolul HTTP.

AJAX - Plain JavaScript folosind XMLHttpRequest[5]

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "info_text.txt", true);
  xhttp.send();
}
```

[5] developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest

AJAX - Plain JavaScript folosind XMLHttpRequest

- XMLHttpRequest.onreadystatechange
  - Un EventHandler care este apelat atunci când atributul readyState este modificat.
- XMLHttpRequest.readyState
  - Returneaza un număr întreg, starea cererii.
    - 0 - UNSENT - Clientul a fost creat, open() nu a fost apelată.
    - 1 - OPENED - open() a fost apelată.
    - 2 - HEADERS_RECEIVED - send() a fost apelată, header-ele și statusul sunt disponibile.
    - 3 - LOADING - Downloading; responseText conține date parțiale.
    - 4 - DONE - Operație comlpetă.

AJAX - Plain JavaScript folosind XMLHttpRequest

- XMLHttpRequest.status
  - Returneaza un număr întreg cu starea raspunsului la cerere.
- XMLHttpRequest.open()
  - Inițializează o cerere.
- XMLHttpRequest.send()
  - Trimite cererea. Daca cererea este asincronă (implicit), metoda va returna imediat ce cererea este trimisă.

AJAX - jQuery[6]

```
1  $.ajax({
2    url: "info_text.txt",
3    async: true,
4    context: document.body,
5    method: "GET"
6  }).done(function(resp) {
7    $("#demo").html(resp);
8  });
```

[6] https://api.jquery.com/Jquery.ajax/

AJAX - jQuery

- context
  - Obiectul acesta va fi contextul tuturor callback-urilor legate de AJAX. Implicit, contextul este obiectul care reprezintă setările AJAX folosite în apel. ($.ajaxSettings impreună cu setările trimise funcției $.ajax)
- html()
  - Descriere: Seteaza conținutul HTML al fiecărui element în setul de elemente selectate.

XMLHttpRequest vs jQuery

- Ambele se folosesc defapt de XMLHttpRequest.
- jQuery - cod mai scurt.
- Nici o diferență în performanță.
- $.post() sau $.get() - request-uri AJAX mai scurte, dedicate doar metodelor POST sau GET.

Exemplu concret

Extragerea datelor de la un API care ofera date meteorologice.

Un exemplu concret

```
1   function get_weather_data(){
2     $.ajax({
3       url: "http://www.7timer.info/bin/astro.php
4             ?lat=45.760696&lon=21.226788&ac=0
5             &unit=metric&output=json&tzshift=0",
6       async: true,
7       context: document.body,
8       method: "GET"
9     }).done(function(resp) {
10      var data = resp['dataseries'];
11      var new_data = "";
12
13      for(i=0; i<= data.length; i++){
14        new_data += "<p>" + data[i]['temp2m'] + "</p>";
15      }
16      $('#demo').html(new_data);
17    });
18  }
19
20  $( document ).ready(function() {
21    $("#getdata").click(function() {
22      get_weather_data();
23    });
24  });
```

Dar...URL-ul este cam lung, si greu de citit.

Parametrul data

```
1       function get_weather_data(){
2         $.ajax({
3           url: "http://www.7timer.info/bin/astro.php",
4           async: true,
5           context: document.body,
6           method: "GET",
7           data: {
8             'lat': 45.760696,
9             'lon': 21.226788,
10            'ac': 0,
11            'unit': 'metric',
12            'output': 'json',
13            'tzshift': 0
14          }
15        }).done(function(resp) {
16          var data = resp['dataseries'];
17          var new_data = "";
18
19          for(i=0; i<= data.length; i++){
20            new_data += "<p>" + data[i]['temp2m'] + "</p>";
21          }
22          $('#demo').html(new_data);
23        });
24      }
25
26      $( document ).ready(function() {
27        $("#getdata").click(function() {
28          get_weather_data();
29        });
30      });
```

- data - Type: PlainObject or String or Array
  - Datele ce vor fi trimise serverului. Daca metoda HTTP este una ce nu poate avea un corp de mesaj (cum ar fi GET), datele sunt concatenate în URL.

De reținut: În general, HTTP GET va encoda parametrii în URL, în timp ce POST îi va include in corpul mesajului trimis.

Folosirea statusCode

```
1          $.ajax({
2            statusCode: {
3              404: function() {
4                alert("page not found");
5              }
6              200: function() {
7                alert("request successfull")
8              }
9            }
10         });
```

Parametrul statusCode

statusCode - support pentru definirea funcțiilor ce vor fi apelate atunci cand un statusCode anume este returnat.

Recapitulare

- Stiva ISO - OSI & TCP/IP
- HTTP
  - Metode/verbe HTTP
  - Coduri de retur
- XML
- JSON
- XMLHttpRequest()
- $.ajax()