

Tehnologii Web

front-end



back-end



interacțiune Web

suita de tehnologii Ajax + aplicații Web hibride

„Modul în care dăm
face mai mult decât ceea ce dăm.”

Pierre Corneille

Care e modalitatea de a transfera asincron
date între client(i) și server(e) Web?

interacțiune web: **ajax**

Asynchronous JavaScript And XML

(Jeese James Garrett)

permite transferul asincron de date
între un document HTML redat de client (*browser*)
și o aplicație rulând pe un server Web

interacțiune web: ajax

O suită de tehnologii deschise:

limbaje standardizate de structurare – uzual, HTML –
și de prezentare a datelor: CSS

interacțiune web: **ajax**

O suită de tehnologii deschise:

redare + interacțiune la nivel de client (navigator) Web
via standardul **DOM**

interacțiune web: ajax

O suită de tehnologii deschise:

interschimb și manipulare de date reprezentate prin:

JSON

diverse dialecte XML (Atom, RSS, KML,...)

HTML

alte formate – *e.g.*, CSV

interacțiune web: **ajax**

O suită de tehnologii deschise:

transfer (a)sincron de date via HTTP
facilitat de obiectul **XMLHttpRequest**

interacțiune web: **ajax**

O suită de tehnologii deschise:

procesare folosind limbajul **ECMAScript** (JavaScript)

interacțiune web: **ajax**

Componenta de bază: obiectul **XMLHttpRequest**

disponibil la nivelul navigatorului Web via JavaScript

interacțiune web: **ajax**

Componenta de bază: obiectul **XMLHttpRequest**

specificația inițială bazată pe implementarea MSIE

oferită în prezent de (aproape) orice *browser*

www.w3.org/TR/XMLHttpRequest1/

interacțiune web: **ajax**

Componenta de bază: obiectul **XMLHttpRequest**

specificația actuală

HTML5 Living Standard, 15 mai 2019

implementare în toate navigatoarele Web moderne

xhr.spec.whatwg.org

interacțiune web: **ajax**

Componenta de bază: obiectul **XMLHttpRequest**

permite realizarea de cereri HTTP – *e.g.*, GET, POST,... –
dintr-un program rulând la nivel de client (*browser*)
spre o aplicație / un serviciu Web existent(ă) pe server,
în mod **asincron** ori **sincron**

interacțiune web: ajax

Componenta de bază: obiectul **XMLHttpRequest**

paginile Web nu mai trebuie reîncărcate complet,
conținutul lor – structurat via HTML –
fiind manipulat prin DOM în cadrul *browser*-ului,
în conformitate cu datele recepționate de la server

interacțiune web: **ajax**

Metode importante oferite de **XMLHttpRequest**

open ()

inițiază – deschide – o conexiune HTTP cu serverul,
emițând o cerere: GET, POST,...

interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

send ()

transmite (asincron) date – *e.g.*, JSON, XML etc. –,
spre aplicația/serviciul ce rulează pe server

interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

send ()

transmite (asincron) date – *e.g.*, JSON, XML etc. –,
spre aplicația/serviciul ce rulează pe server

orice *listener* (asociat evenimentelor **onloadstart**,
onprogress, **onload**, **onloadend**, **ontimeout**, **onabort**, **onerror**)
trebuie stabilit înainte de a trimite date

interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

abort ()

abandonează transferul de date curent

interacțiune web: ajax

Metode importante oferite de XMLHttpRequest

setRequestHeader ()

specifică anumite câmpuri de antet HTTP

exemple: Cookie, Keep-Alive, User-Agent,...

interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

getResponseHeader ()

furnizează valoarea unui anumit câmp prezent
în antetul mesajului de răspuns HTTP trimis de server

interacțiune web: ajax

Metode importante oferite de **XMLHttpRequest**

getAllResponseHeaders ()

oferă toate câmpurile HTTP trimise de server,
exceptând **Set-Cookie**

interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

readyState

furnizează codul de stare a transferului:

- 0 – **UNSENT**
- 1 – **OPENED**
- 2 – **HEADERS_RECEIVED**
- 3 – **LOADING**
- 4 – **DONE**

interacțiune web: ajax

Proprietăți de bază ale XMLHttpRequest

status

oferă codul de stare HTTP întors de serverul Web:

200 (*Ok*)

404 (*Not Found*)

500 (*Internal Server Error*)

...

interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

statusText

conține mesajul corespunzător codului de stare HTTP

interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

responseText
responseXML

stochează răspunsul (datele) obținut(e) de la server

interacțiune web: **ajax**

Proprietăți de bază ale **XMLHttpRequest**

onreadystatechange

specifică funcția ce va fi invocată la modificările de stare ale transferului de date dintre server și client

handler de tratare a
evenimentelor de transfer

interacțiune web: **ajax**

Excepții ce pot fi emise

AbortError
InvalidAccessError
InvalidStateError
NetworkError
SecurityError
TimeoutError

...

Noutăți:

stabilirea unui **timeout** privind realizarea unei cereri
(la nivel de milisecunde)

o valoare nenulă cauzează realizarea
unei preîncărcări (*fetching*) a resursei

de studiat și **Fetch** (*HTML5 Living Standard*, 3 mai 2019)
fetch.spec.whatwg.org

Noutăți:

datele vehiculate pot fi de mai multe tipuri
(**ArrayBuffer, Blob, Document, DOMString, FormData**)

detalii la xhr.spec.whatwg.org/#interface-formdata

Noutăți:

procesul de transmitere a datelor spre server (*upload*)
poate avea asociat un *handler* specific
via proprietatea **upload**

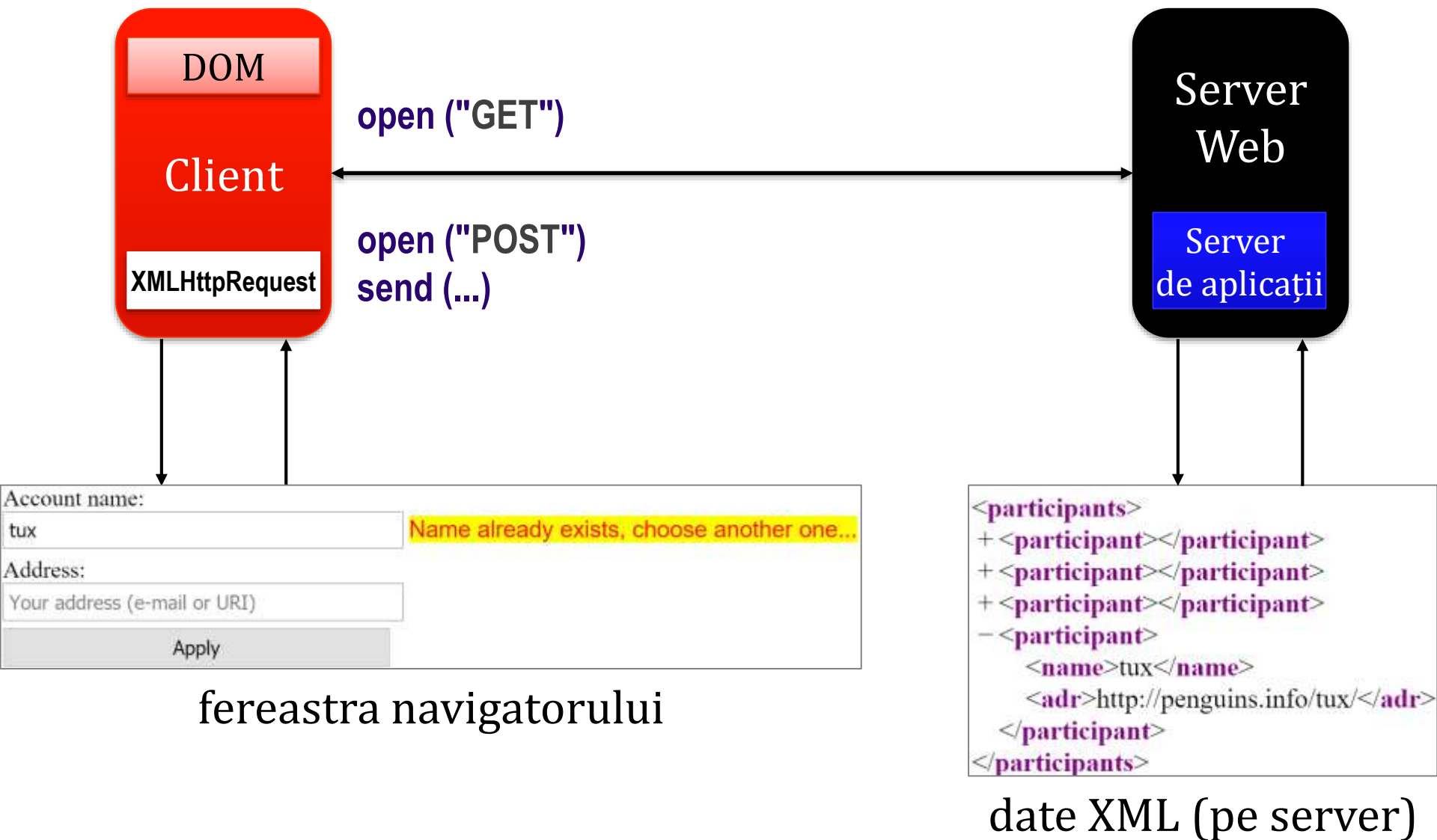
Noutăți:

progresul încărcării poate fi urmărit pe baza funcționalităților specificate de interfața **ProgressEvent**

xhr.spec.whatwg.org/#interface-progresssevent

interacțiune web: **ajax** – studiu de caz

Verificarea existenței unui nume de utilizator
în vederea creării unui cont de autentificare
în cadrul unei aplicații Web



verificarea asincronă a existenței unui cont pe server

interacțiune web: **ajax** – studiu de caz

Verificarea existenței unui nume de utilizator
în vederea creării unui cont de autentificare
în cadrul unei aplicații Web

tratând prin DOM evenimentul **onblur**, putem detecta
– interogând asincron aplicația Web de pe server –
faptul că numele de cont introdus de utilizator
într-un formular Web deja a fost folosit de altcineva

interacțiune web: **ajax** – studiu de caz

Verificarea existenței unui nume de utilizator
în vederea creării unui cont de autentificare
în cadrul unei aplicații Web

aplicația Web de pe server – adoptând stilul REST –
va oferi un document XML modelând răspunsul la
interogarea „există deja un utilizator având un nume dat?”

```
<?php // program PHP, cu rol de serviciu Web, rulat la nivel de server
define('DOCXML', './particip.xml'); // locația documentului XML
// trimitem tipul conținutului; aici, XML
header('Content-type: text/xml');

// funcție care verifică dacă un nume de participant deja există
// returnează 1 dacă numele există, 0 în caz contrar
function checkIfNameExists ($aName) {
    // încercăm datele despre participanți via SimpleXML
    if (!($xml = simplexml_load_file (DOCXML))) { return 0; }
    // parcurgem toți participanții găsiți cu XPath...
    foreach ($xml->xpath('/participants/participant/name') as $name) {
        // comparăm numele, ignorând minusculele de majuscule
        if (!strcasecmp($aName, $name)) { return 1; }
    }
    return 0;
}
?>

<response>
    <result><?php echo checkIfNameExists ($_REQUEST['name']); ?></result>
</response>
```

<!-- Formularul Web preluând date de la utilizator -->

<form action="add.php" method="post">

<div>

<label for="name">Account name:</label>

<input type="text" name="name" id="name"

onblur="javascript:signalNameExists (this.value, ")" />

<!-- mesaj inițial ascuns -->

Name already exists, choose another one...

</div>

<div>

<label for="adr">Address:</label>

<input type="text" name="adr" id="adr" />

</div>

<input type="submit" value="Apply" />

</form>

```
// programul JS executat în cadrul browser-ului
var request; // încapsulează cererea HTTP către serverul Web

function loadXML (url) { // încarcă un document XML desemnat de 'url'
    // verificăm existența obiectului XMLHttpRequest (browser antic?)
    if (window.XMLHttpRequest) {
        request = new XMLHttpRequest (); // există suport nativ
    } else
        if (window.ActiveXObject) { // se poate folosi obiectul ActiveX din MSIE
            request = new ActiveXObject ("Microsoft.XMLHTTP");
        }
    if (request) { // există suport pentru Ajax
        // stabilim funcția de tratare a stării transferului de date
        request.onreadystatechange = handleResponse;
        // preluăm documentul prin metoda GET
        request.open ("GET", url, true);
        request.send (null); // nu trimitem nimic serviciului Web
    }
}
```

// funcția de tratare a schimbării de stare a cererii

function handleResponse () {

// verificăm dacă încărcarea s-a terminat cu succes

if (request.readyState** == 4) {**

// am obținut codul de stare '200 Ok'?

if (request.status** == 200) {**

// procesăm datele recepționate prin DOM

// (preluăm elementul rădăcină al documentului XML)

var response = request.responseXML**.documentElement;**

var res = response.getElementsByTagName**('result')[0].**firstChild**.data;**

// apelăm o funcție ce va modifica arborele DOM al paginii Web

// conform răspunsului transmis de serviciul invocat

...

}

// eventual, se pot trata și alte coduri HTTP (404, 500 etc.)

else {

alert ("A problem occurred:\n" + request.statusText**);**

}

}



vezi
exemplul
din arhivă

A new participant

Account name:

Name already exists, choose another one...

Address:

Apply

Status	Method	File	Domain	Cause	Headers	Cookies	Params	Response
200	GET	verify.php?name=tux	localhost	JS xhr	Response payload			
200	GET	verify.php?name=busaco	localhost	JS xhr	1	<response>		
200	GET	verify.php?name=tux	localhost	JS xhr	2	<result>1</result>		
					3	</response>		

utilizatorul introduce un nume de cont; via Ajax, i se va semnala că deja există, conform răspunsului XML trimis de către serviciul Web

cerere HTTP via URL-ul <http://web.info/verify.php?name=tux>
răspuns XML de forma `<response><result>1</result></response>`

0 = nu există

studiu de caz: RandomAjax

```
<div id="numbers">[Wait, please...]</div>
```

HTML

```
1 #numbers {
2   font-family: monospace;
3   font-size: 1em;
4   width: 12em;
5 }
```

CSS

```
// a JS program that asynchronously get a sequence of numbers generated by random.org
```

JAVASCRIPT

```
const URL = 'https://www.random.org/sequences/?min=1&max=33&col=1&format=plain&rnd=new';
const TIME = 2000;
```

```
try { // trying to instantiate a XMLHttpRequest object
  var xhr = new XMLHttpRequest();
} catch (e) {
  numbers.textContent = 'XMLHttpRequest cannot be instantiated: ' + e.message;
} finally {
  var numbers = document.getElementById('numbers');
  xhr.ontimeout = function () { numbers.textContent = 'Time-out... :('; };
  xhr.onload = function () {
    if (xhr.readyState === 4) { // data arrived
      if (xhr.status === 200) { // response Ok from Web service
        // substituting white spaces with comma and
        // putting the content into the HTML element identified by 'numbers'
        numbers.textContent = xhr.responseText.trim().replace(/\W+/g, ', ');
      } else {
        numbers.textContent = 'An error occurred: ' + xhr.statusText;
      }
    }
  };
  xhr.open("GET", URL, true); // opening connection
  xhr.timeout = TIME; // setting the response time
  xhr.send(null); // sending the HTTP request (no data is provided)
}
```

```
5, 18, 13, 32, 6,
21, 17, 24, 14, 30,
20, 15, 16, 4, 8,
33, 31, 29, 10, 19,
7, 27, 28, 2, 11,
12, 1, 22, 9, 25,
26, 23, 3
```

preia asincron
o secvență de numere
aleatoare generate de
random.org – trimisă ca
text obișnuit
jsfiddle.net/busaco/2254kdqn/

```
var xhr = new XMLHttpRequest ();
var numbers = document.getElementById ('numbers');

// tratarea evenimentului de expirare a timpului de așteptare
xhr.ontimeout = function () { numbers.textContent = 'Time-out... :('; };
// tratarea evenimentului de preluare a datelor solicitate unui serviciu
xhr.onload = function () {
    if (xhr.readyState === 4) { // am primit datele
        if (xhr.status === 200) { // răspuns Ok din partea serverului
            // înlocuim spațiile albe cu virgulă și plasăm conținutul
            // în cadrul elementului HTML identificat prin 'numbers'
            numbers.textContent = xhr.responseText.trim ().replace (/W+/g, ', ');
        } else {
            numbers.textContent = 'An error occurred: ' + xhr.statusText;
        }
    }
};
```

```
// adresa Web a sursei de date
const URL = 'https://www.random.org/sequences/
             ?min=1&max=33&col=1&format=plain';
// timpul maxim de așteptare a răspunsului trimis de server
const TIME = 2000;

// deschidem conexiunea
xhr.open ("GET", URL, true);
// stabilim timpul maxim de așteptare a răspunsului
xhr.timeout = TIME;
// nu expediem date
xhr.send (null);
```

studiu de caz: RandomAjax (Fetch)

```
<div id="numbers">[Wait, please...]</div>
```

HTML

```
// A JS program that asynchronously get (by using the Fetch API) a sequence of numbers by random.org. See also "Introduction to fetch()": https://developers.google.com/web/updates/2015/03/introduction-to-fetch
```

```
const URL = 'https://www.random.org/sequences/?min=1&max=33&col=1&format=plain&rnd=new';
```

```
function status(response) { // using promises to handle the returned HTTP status code
  if (response.status >= 200 && response.status < 300) {
    return Promise.resolve(response)
  } else {
    return Promise.reject(new Error(response.statusText))
  }
}
```

```
fetch(URL)
  .then(status) // checking if data was successfully received
  .then((response) => response.text()) // transforming received data into a string
  .then(function(response) { // processing the number sequence
    // substituting white spaces with comma and
    // putting the content into the HTML element identified by 'numbers'
    var numbers = document.getElementById('numbers');
    numbers.textContent = response.trim().replace(/\W+/g, ', ');
  })
  .catch(function(error) { // an error occurred :(
    numbers.textContent = 'An error occurred: ' + error;
  });
```

```
#numbers {
  font-family: monospace;
  font-size: 1em;
  width: 12em;
}
```

CSS

```
28, 25, 7, 33, 23,
22, 14, 21, 5, 1,
18, 2, 4, 13, 12, 6,
20, 17, 19, 24, 16,
31, 11, 29, 10, 3,
8, 26, 30, 27, 15,
32, 9
```

soluție folosind **Fetch API**
pentru aceeași problemă
jsfiddle.net/busaco/a2q9regd/

Noul API **Fetch** se bazează pe conceptul **promisiune** (***promise***) \equiv rezultat ce ar putea fi oferit în urma execuției unei operații asincrone

*represents an operation that has not completed yet,
but is expected in the future*

exploringjs.com/es6/ch_promises.html

developers.google.com/web/fundamentals/primers/promises

github.com/wbinssmith/awesome-promises

Recurgem la *promises* pentru a realiza procesări
în funcție de codul de stare HTTP primit

```
function status(response) {  
  if (response.status >= 200 && response.status < 300) {  
    // cererea poate fi rezolvată  
    return Promise.resolve(response)  
  } else {  
    // cererea a fost rejectată  
    return Promise.reject(new Error(response.statusText))  
  }  
}
```

fetch (URL)

```
.then (status) // verificăm dacă datele au fost recepționate cu succes
// transformăm obiectul răspunsului în șir de caractere
.then ((response) => response.text ())
// procesăm secvența de numere
.then (function (response) {
    // înlocuim spațiile albe cu virgulă și plasăm conținutul
    // în cadrul elementului HTML identificat prin 'numbers'
    var numbers = document.getElementById ('numbers');
    numbers.textContent = response.trim ().replace (/W+/g, ', ');
})
.catch (function (error) { // a survenit o eroare :(
    numbers.textContent = 'An error occurred: ' + error;
});
```

studiu de caz: PostJSON

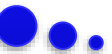
Scenariu – implementare cu JS modern (ES6):
preluăm caractere de la client – introduse în <textarea> –
și le trimitem asincron prin POST unei aplicații Web
rulând pe server care le expediază înapoi

mesaje vehiculate în format JSON
{ "**tasta**": "*caracter*", "**data**": "*secunde*" }

tratăm evenimentul **keypress** pentru a capta tastele
acționate de utilizator


```
// codul programului ES6 interpretat de navigatorul Web
// tratăm evenimentul de apăsare a unei taste
const trateazaEveniment = ev => {
  // mesajul propriu-zis trimis serverului prin POST
  // atunci când survine evenimentul
  let msg = `{ "tasta": "${String.fromCharCode (ev.charCode)}",
               "data": "${Date.now()}" }`;
  // încapsulăm o cerere POST
  let request = new Request ('/ajax/post.php', {
    method: 'POST',
    body: JSON.stringify (msg), // convertim datele JSON în șir de caractere
    headers: {}                 // n-avem câmpuri-antet
  });
```

de consultat
arhiva cu exemple



```
fetch (request) // promitem să executăm codul, transmițând cererea...
.then (response => { // verificăm dacă am primit date JSON de la server
    var contentType = response.headers.get ('Content-Type');
    if (contentType && contentType.includes('application/json')) {
        return response.json (); };
    throw new TypeError ('Datele primite nu-s JSON :('); })
.then (json => {      // procesăm efectiv datele
    // creăm un nod text care indică tasta apăsată
    let elem = document.createTextNode (json.tasta);
    document.getElementById ('tasteApasate').appendChild (elem);
    // raportăm datele primite și la consola browser-ului
    console.log
        (`Date JSON primite: tasta=${json.tasta}, data=${Date(json.data)}`);
    })
};

// via DOM, tratăm evenimentul keypress
document.addEventListener ('keypress', trateazaEveniment);
```

```
// post.php -- program PHP care preia date JSON  
// transmise via POST de client și le trimite înapoi (echo)
```

```
function eJSONValid ($sir) { // verifică dacă datele JSON sunt corecte  
    json_decode ($sir);  
    return json_last_error () == JSON_ERROR_NONE;  
}
```

```
// preluăm de la intrarea standard datele transmise de client (raw data)  
// (aici, cele dintr-o cerere POST)
```

```
$date = trim (file_get_contents ("php://input"));
```

```
if (eJSONValid ($date)) { // trimitem datele JSON înapoi dacă sunt în regulă  
    header ("Content-type: application/json");  
    echo json_decode ($date);  
} else {  
    die ('Date incorecte');  
}
```

aplicația (serviciul) Web invocat(ă) pe server

Redare taste apasate

Web

Taste: Web

Inspector Console Debugger Style Editor Network DOM HTTPS Everywhere

Filter output ☐ Persist Logs

Date JSON primite: taste=W, data=Wed Nov 29 2017 08:58:00 GMT+0200 (EET) post-chars.html:41:6
Date JSON primite: taste=e, data=Wed Nov 29 2017 08:58:01 GMT+0200 (EET) post-chars.html:41:6
Date JSON primite: taste=b, data=Wed Nov 29 2017 08:58:03 GMT+0200 (EET) post-chars.html:41:6

All HTML CSS JS XHR Fonts Images Media Flash WS Other ☐ Persist Logs ☐ Disable cache Filter URLs

Stat...	Met...	File	Domain	Cause	Ty...	Tran...	Size	Headers	Cookies	Params	Response
200	POST	post.php	localhost:88...	JS fetch	json	379 B	41 B				
200	POST	post.php	localhost:88...	JS fetch	json	378 B	41 B				
200	POST	post.php	localhost:88...	JS fetch	json application/json		41 B				

Filter properties

JSON

taste: e
data: 1511938681737

Response payload

1 { "taste": "e", "data": "1511938681737" }

(în loc de) pauză



Ce alte aspecte trebuie considerate
atunci când se recurge la Ajax?

interacțiune web: ajax – utilizări

Reîmprospătarea periodică a conținutului

e.g., știri recepționate în formate ca Atom sau RSS,
mesaje în cadrul aplicațiilor sociale, notificări,...

interacțiune web: ajax – utilizări

Anticiparea *download*-urilor

pre-încărcarea datelor (*e.g.*, imagini) ce vor fi solicitate

interacțiune web: ajax – utilizări

Auto-completarea datelor

auto-completion

sugestii de căutare – exemplu: Google Suggest

interacțiuni web: ajax – utilizări

Validarea în timp-real a datelor introduse
în formulare de către utilizator

exemplificare:

verificarea existenței unui cont sau a unei localități

interacțiune web: ajax – utilizări

Creare de componente de interfață Web (*widgets*)
sau de aplicații Web rulând pe platforme mobile

interacționează cu utilizatorul
pe baza evenimentelor survenite

interacțiune web: ajax – aspecte

Evitarea încărcării întregului document Web

avantaj:

se pot modifica doar fragmente de document

dezavantaj:

bookmarking-ul poate fi compromis
(nu există un URL unic desemnând
reprezentarea resursei curente)

interacțiune web: ajax – aspecte

Oferirea de alternative la Ajax,
atunci când suportul pentru acesta
nu este implementat/activat

graceful degradation

progressive enhancement

interacțiune web: ajax – aspecte

Minimizarea traficului dintre *browser* și server

interacțiune web: ajax – aspecte

Transferul de date poate fi monitorizat
(+interceptat) via instrumente dedicate

la nivel de *desktop*:

instrumentul **WireShark**

www.wireshark.org/docs/wsug_html_chunked/

All

HTML

CSS

JS

XHR

Fonts

Images

Media

Flash

WS

Other

☐ Persist Logs

☐ Disable cache

Status	Method	File	Domain	Cause	Type	Transferr.
200	POST	/_save/	jsfiddle.net	xhr	json	311 B

One request

87 B / 311 B transferred

Finish: 1.71 min

DOMContentLoaded: 2.53 s

load: 2.31 s

Filter output

GET https://docs.google.com/static/forms/client/css/2785006849-formview_embedded_st_ltr.css

GET https://docs.google.com/static/forms/client/js/1918813819-formviewer_prd.js

Use of getPreventDefault() is deprecated. Use defaultPrevented instead.

GET https://togetherjs.com/togetherjs/images/button-share-active.png

GET https://togetherjs.com/togetherjs/images/icon-close-active.png

Send: Object { type: "form-focus", element: "#savenew" }

Send: Object { type: "cursor-click", element: "#savenew", offsetX: 40.366668701171875, offsetY: 31 }

POST XHR https://jsfiddle.net/_save/

Headers

Cookies

Params

Response

Timings

Filter properties

JSON

pastie_url_relative: /busaco/akq5f1ht/

slug: akq5f1ht

shell_id: 317850410

Response payload

1

{"pastie_url_relative": "/busaco/akq5f1ht/", "slug": "akq5f1ht", "shell_id": 317850410}

GET https://jsfiddle.net/busaco/akq5f1ht/embedded/

inspecție a datelor vehiculate
cu instrumentele pentru
dezvoltatorii Web
oferite de *browser*

interacțiune web: ajax – aspecte

Stabilirea unui mod clar
de interacțiune cu utilizatorul

interacțiune HTML clasică

versus

interacțiune „bogată” cu Ajax

versus

interacțiune la nivelul unei aplicații convenționale

interacțiune web: ajax – aspecte

Adoptarea Ajax pentru creșterea utilizabilității,
nu doar de dragul tehnologiei

exemple negative:

distragerea utilizatorului

abuz de resurse (supradimensionarea arborelui DOM)

interacțiune web: ajax

Ajax oferă premisele invocării asincrone de (micro-)servicii Web în stilul REST

folosind diverse reprezentări ale datelor transferate:

POX (Plain Old XML)

JSON (JavaScript Object Notation)

AHAH (Asynchronous HTML and HTTP)

text neformatat

Șabloane de proiectare AJAX

amănunte în cartea Michael Mahemoff,
Ajax Design Patterns, O'Reilly, 2006

www.oreilly.com/library/view/ajax-design-patterns/0596101805/

Șabloane de proiectare AJAX

privind programarea:

invocare de servicii Web
(*RESTful Service, JSON Message*)

Șabloane de proiectare AJAX

privind programarea:

dialog între navigatorul Web și server

(Periodic Refresh, Submission Throttling, Cross-Domain Proxy)

Șabloane de proiectare AJAX

privind programarea:

asigurarea performanței

*(Fat Client, Browser-Side Cache, Guesstimate,
Predictive Fetch, Code Compression, On-Demand JS)*

Șabloane de proiectare AJAX

referitoare la interacțiunea cu utilizatorul:

formulare Web

(Edit-in-place, Rich Text Editor, Live Search)

Șabloane de proiectare AJAX

referitoare la interacțiunea cu utilizatorul:

widget-uri de afișare a conținutului

(Data Grid, Progress Indicator, Suggestion, Slider, Status Area)

Șabloane de proiectare AJAX

referitoare la interacțiunea cu utilizatorul:

acțiuni oferite

(Drag-and-Drop, Popup, Upload/Download Files)

detalii la master,

în cadrul materiei *Human-Computer Interaction*

profs.info.uaic.ro/~busaco/teach/courses/hci/hci-film.html

Șabloane de proiectare AJAX

referitoare la interacțiunea cu utilizatorul:

efecte vizuale

(One-Second Spotlight, One-Second Motion, Highlight)

Șabloane de proiectare AJAX

referitoare la interacțiunea cu utilizatorul:

funcționalitate

*(Lazy Registration, Direct Login, Timeout,
Heartbeat, Autosave, Unique URLs)*

Șabloane de proiectare AJAX

inginerie Web:

monitorizare & diagnoză
(*Logging, Debugging*)

Șabloane de proiectare AJAX

inginerie Web:

inspecție de cod/date
(*DOM Inspection, Traffic Sniffing*)

Șabloane de proiectare AJAX

inginerie Web:

testare

*(Simulation Service, Browser-Side Test,
Service Test, System Test)*

Care e suportul vizând implementarea?

interacțiune web: ajax – programare

La nivel de client

multitudine de biblioteci + *framework*-uri JavaScript
www.javascripting.com/search?q=ajax

micro-biblioteci:
microjs.com/#ajax

interacțiune web: ajax – programare

La nivel de server

biblioteci, module, *framework*-uri – exemple:

Java – **Apache Wicket**, **DWR**, **OpenXava**, **Vaadin** etc.

.NET – **AJAX Control Toolkit**: devexpress.com/act

Node.js – www.npmjs.com/search?q=ajax

PHP – **Cjax**: github.com/ajaxboy/cjax

Perl – **CGI::Ajax**, **Catalyst**, **Mason** etc.

Python – wiki.python.org/moin/WebFrameworks

...

interacțiune web: ajax – programare

API-uri specializate

exemplificări:

acces la resurse cartografice – **HERE JavaScript APIs**
developer.here.com/develop/javascript-api

fotografii de calitate oferite de Unsplash API – **Unsplash.js**
github.com/unsplash/unsplash-js

Ajax în contextul extensiilor **WordPress**
codex.wordpress.org/AJAX_in_Plugins

interacțiune web: comet

Comet

termen propus de Alex Russel

permite ca datele să fie „împinse” (*push*) de către server spre aplicația client, utilizând conexiuni HTTP persistente (*long-lived*) în vederea reducerii latenței

interacțiune web: comet

Șablon de proiectare a aplicațiilor Web
care necesită realizarea de conexiuni persistente,
în stilul *peer-to-peer*

utilizat de aplicațiile Web intensiv interactive,
eventual colaborative – *e.g.*, Mibbit

interacțiune web: comet

Complementar Ajax

long polling

HTTP server push

Reverse Ajax

de studiat M. Carbou, “*Reverse Ajax, Part 1. Introduction to Comet*”, IBM developerWorks, 2011
www.ibm.com/developerworks/web/library/wa-reverseajax1/

interacțiune web: comet

Instrumente software – exemplificări:

Atmosphere – github.com/Atmosphere/atmosphere-javascript

APE (*Ajax Push Engine*) – www.ape-project.org

Axios – github.com/axios/axios

Fermata – github.com/natevw/fermata

Push – pushjs.org

interacțiune web: comet

Soluții alternative, moderne:
adoptarea diverselor tehnologii HTML5

server-sent events
WebSocket

detalii la „Dezvoltarea aplicațiilor Web cu JavaScript”
(curs opțional, anul 3, semestrul I)

profs.info.uaic.ro/~busaco/teach/courses/staw/

mash-ups

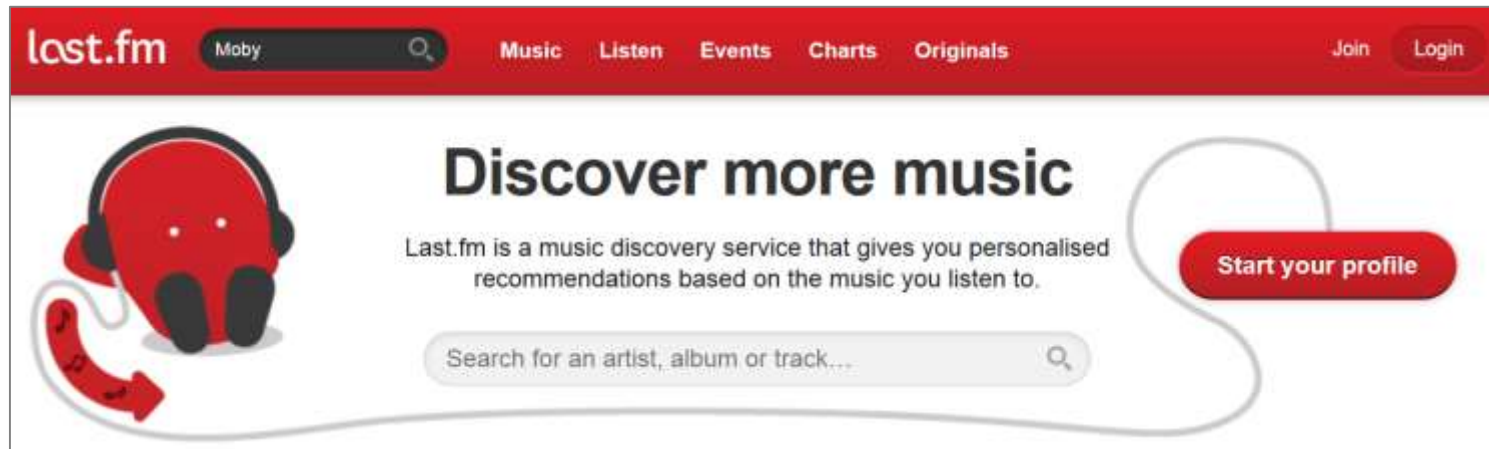
Ajax/Comet oferă suport pentru dezvoltarea de aplicații Web hibride – *mash-ups*

combinarea – la nivel de client și/sau server – a conținutului ce provine din surse (situri) multiple, oferind o funcționalitate/experiență nouă

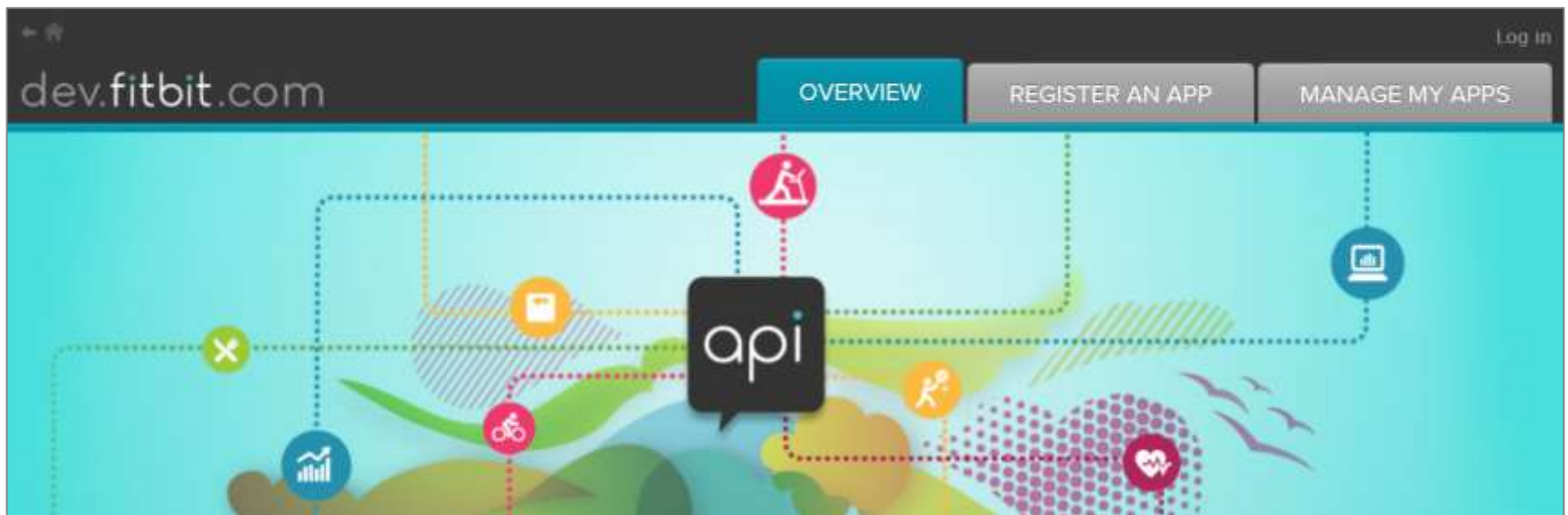
mash-ups

Exemplificare:

dorim să oferim o aplicație ce pune la dispoziție
informații din domeniul muzical
în funcție de activitățile fizice ale utilizatorului,
pe baza unor servicii Web publice



www.last.fm/api/rest



dev.fitbit.com/build/guides/

Account

[Add API account](#)

API Guides

[Introduction](#)

[User Authentication](#)

[Scrobbling](#)

[Radio API](#)

[Playlists API](#)

[Tools](#)

[REST requests](#)

[XML-RPC requests](#)

[Error codes](#)

[Terms of Service](#)

API Methods

Album

[album.addTags](#)

[album.getInfo](#)

[album.getTags](#)

[album.getTopTags](#)

[album.removeTag](#)

[album.search](#)

Artist

[artist.addTags](#)

[artist.getCorrection](#)

[artist.getInfo](#)

[artist.getSimilar](#)

[artist.getTags](#)

[artist.getTopAlbums](#)

[Last.fm Web Services](#)

REST Requests

The API root URL is located at <http://ws.audioscrobbler.com/2.0/>

Generally speaking, you will send a method parameter expressed as 'package.method' along with method specific arguments to the root URL. The following parameters are required for all calls:

api_key : A Last.fm API Key.

method : An API method expressed as *package.method*, corresponding to a documented last.fm API method name.

For example:

```
http://ws.audioscrobbler.com/2.0/?method=artist.getSimilar&api_key=xxx...
```

If you are accessing a *write* service, you will need to submit your request as an HTTP POST request. All POST requests should be made to the root url:

```
http://ws.audioscrobbler.com/2.0/
```

With all parameters (including the 'method') sent in the POST body. In order to perform write requests you will need to authenticate a user with the API. See [authentication](#) for more.

REST Responses

Responses will be wrapped in an lfm status node

```
<lfm status="$status">
  ...
</lfm>
```

acces la serviciile REST
despre formatii + albume
via o cheie de autentificare

Get user's *profile* in the *format* requested using *units* in the *unit system* which corresponds to the *Access Type*.

Access Type: Read

Rate Limited: Yes

OAuth: **oauth_token** is optional, if omitted you should explicitly specify `<user-id>`.

Privacy: Basic profile is always public, **About Me** (Friends or Anyone), **Age and height** (Friends or other user's respective profile fields, considering:

- authenticated owner will receive all values, others will receive the correct values for accessible fields, empty string, "NA" (empty gender), 0 (empty height), default avatar etc., some values revealed

API-ul REST de la FitBit oferă date în formatele JSON și XML

Resource URL

GET `/<api-version>/user/<user-id>/profile.<response-format>`

api-version	The API version. Currently 1.
user-id	User's encoded id or "-" (dash) to indicate user currently authenticated via the app.
response-format	The response format. Currently supported response formats are json and xml .

Examples

GET `/1/user/228TQ4/profile.json`

GET `/1/user/228TQ4/profile.xml`

GET `/1/user/-/profile.json`

API

Fitbit

Service

https://api.

Select an API method

Search methods...

PROFILE

GET	Get User Info	
POST	Update User Info	

BODY

GET	Get Body Measurements	
GET	Get Body Weight	
GET	Get Body Fat	
GET	Get Badges	
GET	Get Time Series	
POST	Log Body Measurements	
POST	Log Body Weight	
POST	Log Body Fat	
DELETE	Delete Body Weight Log	
DELETE	Delete Body Fat Log	

mash-ups

www.last.fm/api/rest



+



dev.fitbit.com



FiLa
aplicație Web hibridă

studiu de caz: DoCa

Crearea unui *mash-up* Web la nivel de client
DoCA (*Dogs 'n' Cats*)

imagini cu câini dog.ceo/dog-api/ – răspuns JSON:
{ "status": "success", "message": "URL_image" }

+

fotografii cu pisici aws.random.cat/meow – răspuns JSON:
{ "file": "URL_image" }

HTML ▼

CSS ▼

avansat

```
1 <div><span id="dog"></span> vs <span id="cat"></span></div>
```

```
1 img {  
2   width: 200px;  
3 }
```

JavaScript + No-Library (pure JS) ▼

```
23  
24 // folosim Dog API pentru a obține o imagine aleatorie  
25 // a unui câine dalmatian  
26 fetch(URLDOGS)  
27   .then(status) // datele au fost recepționate cu succes?  
28   .then(response => response.json())  
29   .then(json => {  
30     // procesăm datele JSON primite...  
31     // proprietatea 'message' stochează URL-ul imaginii câinelui  
32     genImg (json.message, 'dog');  
33   })  
34   .catch(error => { // redăm eroare survenită  
35     document.getElementById('dog').textContent = error;  
36   });  
37  
38 // invocăm random.cat API pentru a obține  
39 // fotografia unei pisici  
40 fetch(URLCATS)  
41   .then(status) // datele au fost recepționate cu succes?  
42   .then(response => response.json())  
43   .then(json => {  
44     // procesăm datele JSON primite...  
45     // proprietatea 'file' stochează URL-ul fotografiei cu pisici  
46     genImg (json.file, 'cat');  
47   })  
48   .catch(error => { // redăm eroare survenită  
49     document.getElementById('cat').textContent = error;  
50   });
```



vs



utilizăm două apeluri fetch()
pentru a prelua date JSON
de la cele două servicii Web
(API-uri publice)

URL-urile obținute sunt
folosite pentru a genera
cu DOM elementele
corespunzătoare

codul-sursă complet la jsfiddle.net/busaco/z2f3vp4m/

Status	Method	Domain	File	Cause	Type	Transferred	Size
200	GET	dog.ceo	random	fetch	json	704 B	89 B
200	GET	aws.random.cat	meow	fetch	json	430 B	66 B

https://aws.random.cat/meow

Headers Cookies Params **Response** Timings

Filter properties

▼ JSON

file: <https://purr.objects-us-east-1.dream.io/i/nhyM9.jpg>

▼ Response payload

1 ▼ `{"file":"https://purr.objects-us-east-1.dream.io/i/nhyM9.jpg"}`

inspectarea datelor obținute prin transfer asincron
cu instrumentele pentru dezvoltatori ale *browser*-ului Web

mash-ups

Se bazează pe fluxuri de știri RSS/Atom,
servicii Web, API-uri publice,...

„curentul” **SaaS** (*Software As A Service*)

implementare la nivelul:
clientului (*browser*-ului) Web și/sau serverului Web

mash-ups

Caratteristici:

combinare

vizualizare

agregare

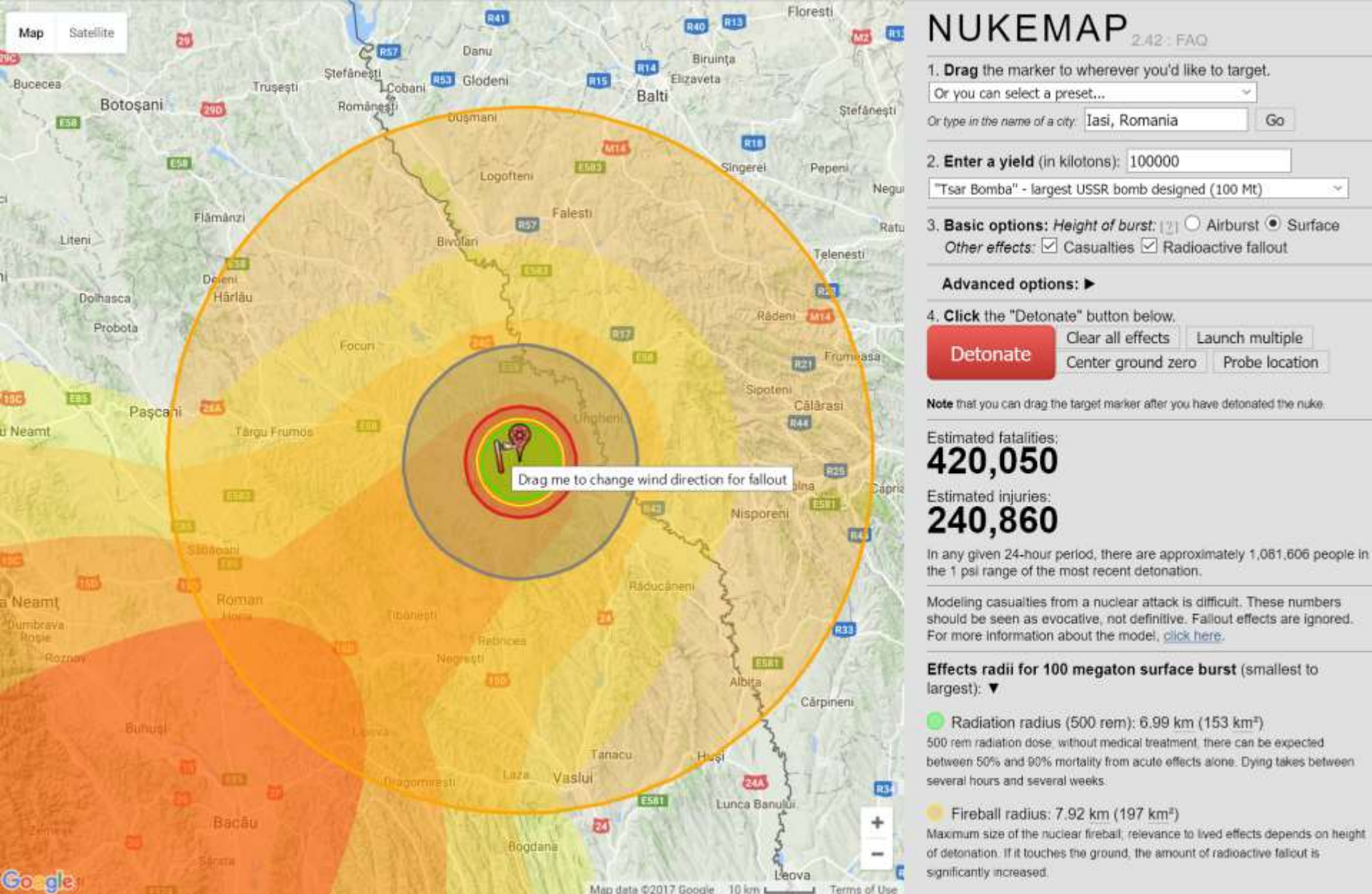
mash-ups

Combinare

utilizarea de surse de date multiple
poate avea caracter multidimensional

de exemplu,
subiect de interes + locație geografică + moment de timp

Yahoo! Music Search + Google Maps + Eventful



un *mash-up* Web de studiere a efectelor detonării bombelor nucleare – nuclearsecrecy.com/nukemap/

mash-ups

Vizualizare

pot fi adoptate diverse tehnici de vizualizare
(prezentare) a datelor:

chart-uri, cartografică, *tag cloud-uri*, tridimensională,...



metode diverse de vizualizare în timp-real
a evoluției cursului monedelor virtuale – **Coinorama**

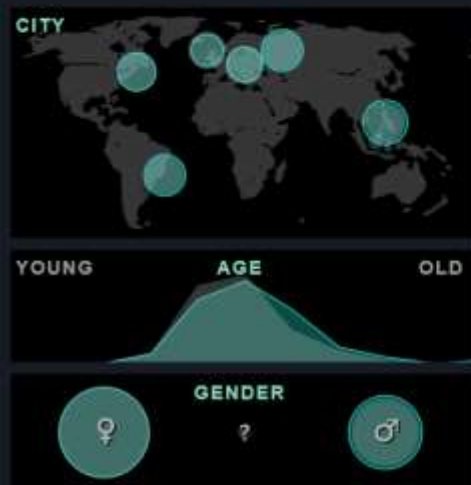
mash-ups

Agregare

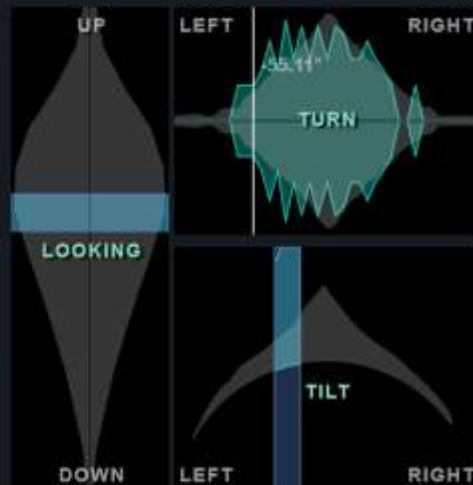
gruparea datelor provenite din mai multe surse și
analizarea lor: statistici, clasificări, predicții,...

e.g., folosind *machine/deep learning* se pot releva
aspecte „ascunse” ale datelor procesate

DEMOGRAPHICS



POSE



FEATURES



MOOD



74 of 3840 selfies.



clasificarea imaginilor expuse de utilizatori
în funcție de vârstă, postură, stare de spirit și altele
Selfieexploratory – selfiecity.net/selfieexploratory/

mash-ups

Surse de date (<i>data feeds</i>)	Atom, RSS, geoRSS, micro-date HTML5, RDFa,...
Interfețe de programare (API-uri)	specifice serviciilor publice și de procesare JSON/XML/RDF
Biblioteci/ <i>framework</i> -uri pentru dezvoltare	<i>framework</i> -uri Web generice sau oferite de organizații
Instrumente interactive (<i>Web tools</i>)	eventual, disponibile în <i>cloud</i>
Platforme (<i>Platform As A Service</i>)	Digital Ocean, Heroku, Google Cloud Platform, MS Azure,...

How we have changed the world

Refresh

A TALE OF NEW CITIES

You are 2 years younger than
Navi Mumbai

India



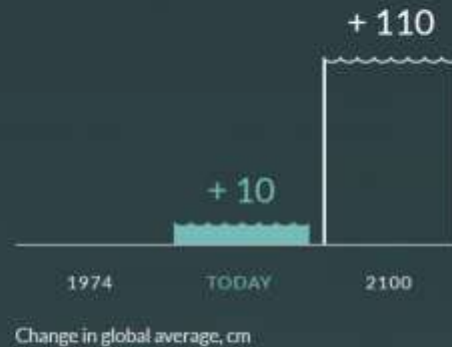
ALL YOU CAN EAT

Amount available per person



SEA CHANGE

Rise of sea level



POWER SUPPLY

Change in energy supply since 1980

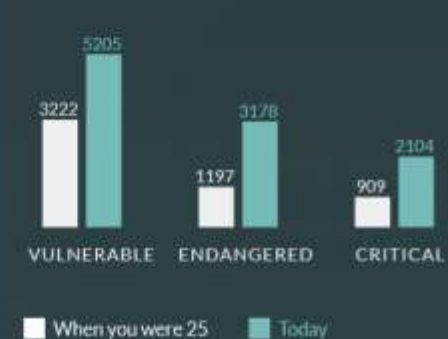


Marine fish caught



LIFE ON THE EDGE

Number of plant species under threat



EMISSION CONTROL

CO₂ emissions in your lifetime



Your Life on Earth (BBC)

www.bbc.com/earth/story/20141016-your-life-on-earth



Search the Largest App and Mashup Directory on the Web

Search

SEARCH MASHUPS

Filter Mashups

Best Practices ✕



Database-as-a-Service ✕

By APIs

☐ Include Deprecated

Mashup Name	Description	Category	Submitted
Honeygain	Honeygain is a mobile application in which users can earn money by allowing application access to...	Monetization	05.13.2019
Petdoption	Android app which allows users to search for adoptable pets in the United States using the...	Animals	04.04.2019
LandedCost.io Consolidated Screening List	The Consolidated Screening List (CSL) is a list of parties for which the United States Government...	eCommerce	04.01.2019
Voice Apps	Here any one can develop voice apps for Amazon Echo,	Voice	03.31.2019



API UNIVERSITY

FEATURED

LATEST

FOR API PROVIDERS

What Are APIs and How Do They Work?

8 Real World API Strategies and the Keys to Their Success

Microservices 101: Understanding and Leveraging Microservices

[More for API Providers >](#)

FOR DEVELOPERS

How to Get Started With Google Actions

How to Build a Monitoring Application With the Google Cloud Vision API

How to Access Any RESTful API Using the R Language

[More for Developers >](#)

lista *mash-up*-urilor: www.programmableweb.com/mashups/directory
 multe alte API-uri publice la github.com/toddmotto/public-apis

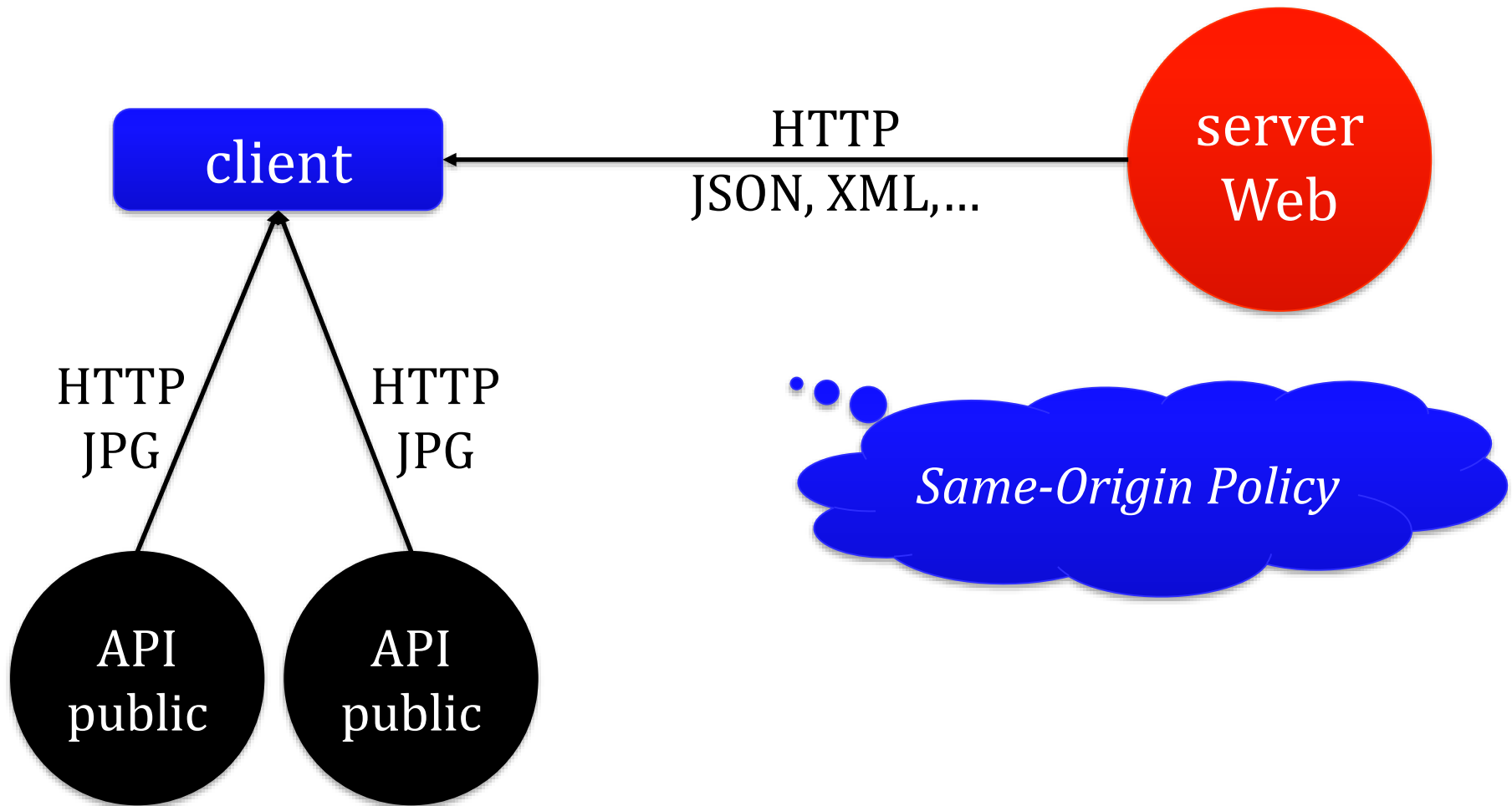
Nu există o problemă de securitate
privind accesul la resurse via JavaScript?

Same-Origin Security Policy

“restricts how a document or script loaded from one origin can interact with a resource from another origin”

astfel, un program JavaScript trebuie să acceseze doar datele aparținând aceleasi origini
– *i.e.*, provenite din același domeniu Internet

avansat



se permit doar transferuri vizând reprezentări de resurse referitoare la imagini, fișiere CSS și alte programe JavaScript aparținând aceleași origini

Same-Origin Security Policy

previne cazurile în care un document/program încărcat dintr-o origine să poată accesa/modifica proprietăți ale unui document aparținând altei origini

developer.mozilla.org/Web/Security/Same-origin_policy


```
var url = "https://profs.info.uaic.ro/~busaco/teach/courses/cliw/";

// realizăm o cerere HEAD pentru a obține meta-date despre o resursă
var client = new XMLHttpRequest ();
client.open ("HEAD", url, true);
client.send ();
client.onreadystatechange = function () {
    // am recepționat câmpurile-antet?
    if (client.readyState == 2) {
        // semnalăm tipul MIME și data ultimei actualizări
        alert ("Resursa de tip " +
            client.getResponseHeader ("Content-Type") + " s-a actualizat la " +
            client.getResponseHeader ("Last-Modified"));
    }
}
```

preluarea cu **HEAD** a unor
meta-date, în mod asincron

URL al altui domeniu ► se încalcă *Same Origin Policy*

The screenshot shows a web browser's developer console with the 'Console' tab selected. The top toolbar includes icons for Elements, Console, Sources, Network, Performance, and Memory. The console shows a red error message: 'Access to XMLHttpRequest at 'https://profs.info.uaic.ro/~busaco/teach/courses/c_head.html:1 liw/' from origin 'null' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.' Below this, a yellow warning message states: 'Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://profs.info.uaic.ro/~busaco/teach/courses/cliw/. (Reason: CORS header 'Access-Control-Allow-Origin' missing). [Learn More]'. The bottom of the console has a blue double arrow icon.

Elements Console Sources Network Performance Memory » 1

top -url:https://profs.info.ua Default levels

✖ Access to XMLHttpRequest at 'https://profs.info.uaic.ro/~busaco/teach/courses/c_head.html:1 liw/' from origin 'null' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.

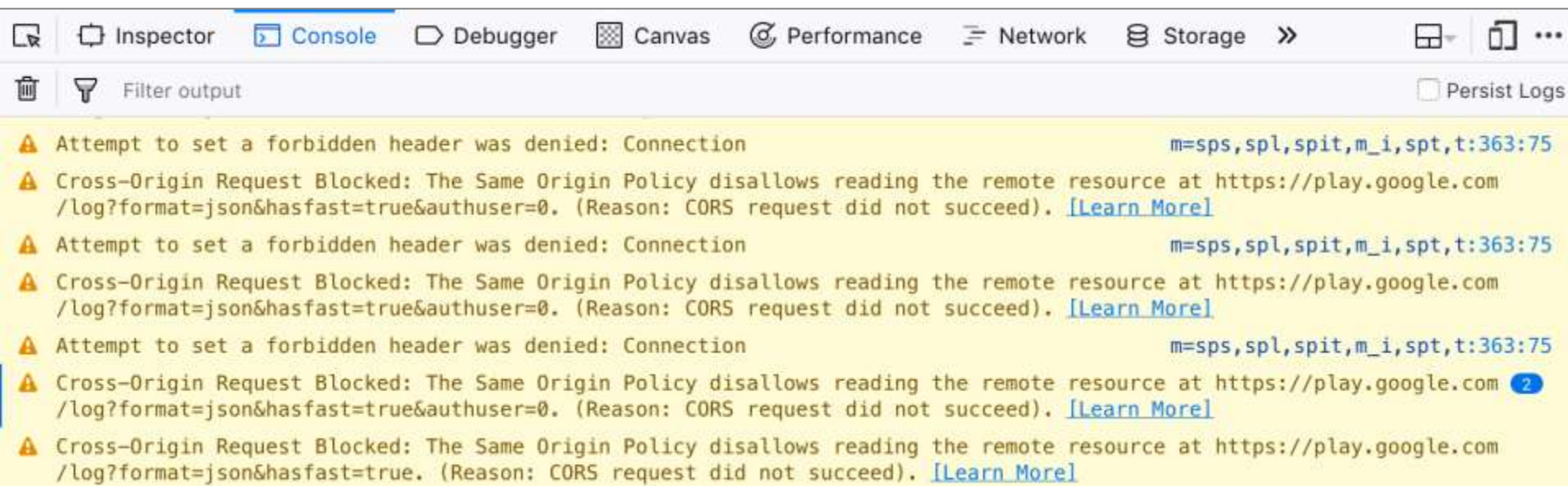
Inspector Console Debugger Canvas Performance Network Storage <> DOM »

Filter output Persist Logs

⚠ Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://profs.info.uaic.ro/~busaco/teach/courses/cliw/. (Reason: CORS header 'Access-Control-Allow-Origin' missing). [\[Learn More\]](#)

»

caz real: Google Mail



The screenshot shows the developer console of a web browser with the 'Console' tab selected. The console displays a series of error messages related to CORS and forbidden headers. The interface includes tabs for Inspector, Console, Debugger, Canvas, Performance, Network, Storage, and a 'Filter output' section. A 'Persist Logs' checkbox is visible in the top right of the console area.

Inspector Console Debugger Canvas Performance Network Storage »

Filter output ☐ Persist Logs

- Attempt to set a forbidden header was denied: Connection m=sps,spl,spit,m_i,spt,t:363:75
- Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <https://play.google.com/log?format=json&hasfast=true&authuser=0>. (Reason: CORS request did not succeed). [\[Learn More\]](#)
- Attempt to set a forbidden header was denied: Connection m=sps,spl,spit,m_i,spt,t:363:75
- Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <https://play.google.com/log?format=json&hasfast=true&authuser=0>. (Reason: CORS request did not succeed). [\[Learn More\]](#)
- Attempt to set a forbidden header was denied: Connection m=sps,spl,spit,m_i,spt,t:363:75
- Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <https://play.google.com/log?format=json&hasfast=true&authuser=0>. (Reason: CORS request did not succeed). [\[Learn More\]](#) 2
- Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at <https://play.google.com/log?format=json&hasfast=true&authuser=0>. (Reason: CORS request did not succeed). [\[Learn More\]](#)

CORS (*Cross-Origin Resource Sharing*)

recomandare a Consorțiului Web (2014)

www.w3.org/TR/cors/

permite partajarea la nivel de client a resurselor
provenind din domenii Internet diferite

astfel, se pot emite cereri între domenii (*cross-origin*)

...Și testul #2?

[A—K] Câți arbori DOM poate crea/manipula o aplicație Web de tip *mash-up*? Discutați și exemplificați.

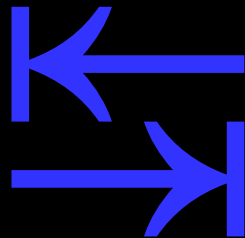
[L—Z] Care e rolul câmpului Content-Type la transferul asincron via Ajax? De cine poate fi stabilit? Argumentați și oferiți minim un exemplu.

[A—L] Câte transferuri (a)sincrone se pot crea via obiectul XMLHttpRequest? Argumentați și oferiți un exemplu.

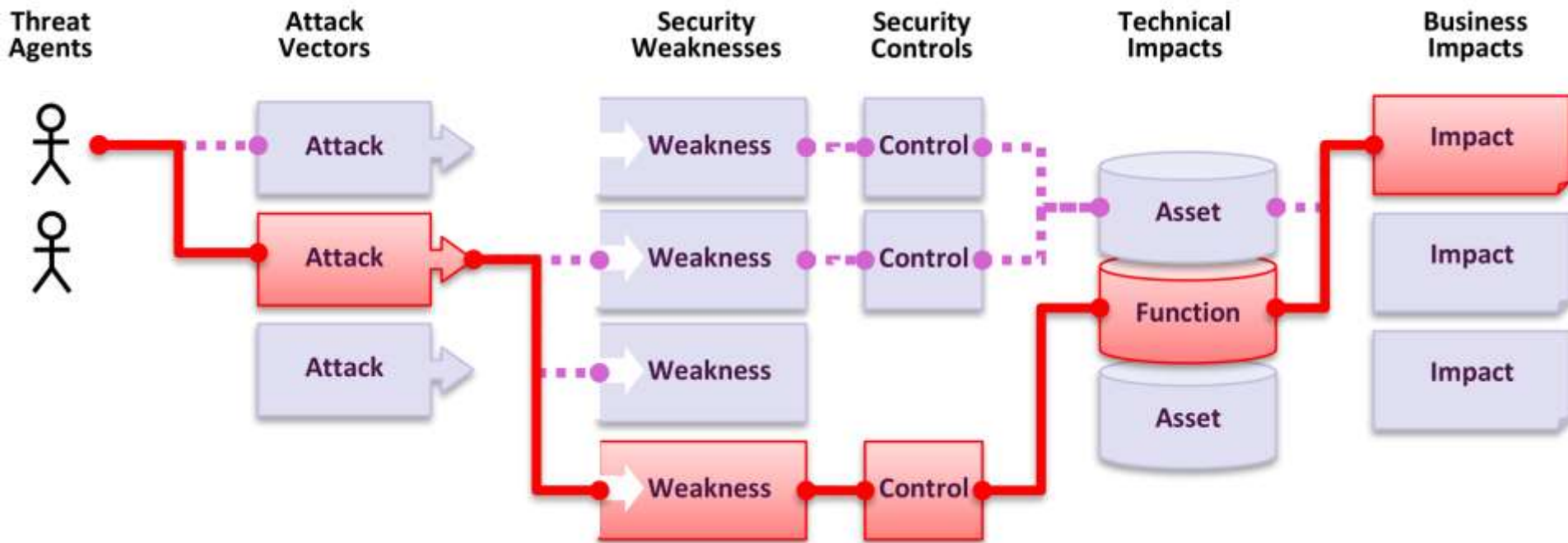
[M—Z] Există situații în care un *mash-up* Web nu necesită stocarea persistentă a datelor la nivel de server și/sau client? Discutați și exemplificați.

rezumat

interacțiune Web



de la interacțiuni asincrone via Ajax
la *mash-up-uri* Web



„ultimul” episod: **securitatea aplicațiilor Web**