

JavaScript / DOM / Ajax

1. Progressive Enhancement – Hijax
2. Prototype – “making JavaScript suck less”
3. Scriptaculous – efecte vizuale
4. Transport de date server–client: XML, JSON, HTML, plain–text
5. Studiu de caz: Ajax shopping cart

1. Progressive Enhancement – Hijax

* Introducere

Definește scopurile și grupurile de utilizatori ale site-ului sau aplicației pe care o crezi.

Un site web ar trebui să fie utilizabil și fără JavaScript.

Uneori alegem să nu suportăm tehnologiile vechi (GMail, Google Maps)

1. Progressive Enhancement – Hijax

* Cum se face?

1. Construiește aplicația în stilul clasic, cu request-uri HTTP obișnuite.
2. Interceptează (*hijack*) link-urile clasice cu JavaScript și folosește XMLHttpRequest pentru a transmite datele către server.

1. Progressive Enhancement – H Ajax

* Metodologie

1. Conținutul (XHTML)
2. Prezentarea (CSS)
3. Comportamentul (JavaScript)

1. Progressive Enhancement – Hajax

* K.I.S.S.

Folosește obiectul XMLHttpRequest doar pe post de transport al datelor între client și server.

Business–logic își are locul pe server. Păstrează codul JavaScript simplu.

1. Progressive Enhancement – Hijax

* De ce?

Hijax te ajută să economisești timp – nu ai de construit variantă cu Ajax și variantă fără Ajax.

D.R.Y. – Don't Repeat Yourself – Codul de business logic rămâne într-un singur loc (pe server) și nu e duplicat (pe client).

Dimensiuni reduse ale fișierelor .js.

Produsul final este utilizabil și cu tehnologii mai vechi.

Întrebări?

2. Prototype

* Intra direct in hora

Intre <head> si </head>:

```
<script type="text/javascript" src="prototype.js"></script>
```

Unul din cele mai puternice toolkit-uri JavaScript. Inclus in distributia Ruby On Rails si folosit din plin acolo.

Alte toolkit-uri JavaScript: jQuery si Dojo

2. Prototype

* Un pic de DOM

```
<div id="header">Geek Meet</div>
```

JavaScript simplu:

```
document.getElementById("header")
```

Prototype:

```
$("#header")
```

2. Prototype

* Arrays & Loops

```
var orase = ["Oradea", "Timișoara", "Brașov", "București"];
```

```
for (i = 0; i < ... las-o baltă!
```

```
orase.each(function(oras) {  
  alert(oras.value);  
})
```

2. Prototype

* Forms

```
<form id="contact" action="sendmessage.php" method="post">  
  <input type="text" name="nume" id="nume" />  
  <textarea name="mesaj" id="mesaj"></textarea>  
</form>
```

```
Form.serialize($("#contact")); // nume=JOHNNY&mesaj=MSG
```

2. Prototype

* Manipulare DOM

```
<div id="header">Geek Meet</div>
```

```
Element.hide($("#header"));
```

```
// in loc de document.getElementById("header").style.display = "none";
```

```
Element.show($("#header"));
```

```
// in loc de document.getElementById("header").style.display = "block";
```

```
Element.update($("#header"), "Geek Meet – 10 iunie 2006");
```

```
// in loc de document.getElementById("header").innerHTML = "Geek Meet – 10  
// iunie 2006";
```

2. Prototype

* Inserții DOM!

```
<div id="header">Geek Meet</div>
```

```
new Insertion.Before("id", "<h1>Titlu</h1>");  
// <h1 id="id">Titlu</h1><div id="header">Geek Meet</div>
```

```
new Insertion.Top("id", "<h1>Titlu</h1>");  
// <div id="header"><h1 id="id">Titlu</h1>Geek Meet</div>
```

```
new Insertion.Bottom("id", "<h1>Titlu</h1>");  
// <div id="header">Geek Meet<h1 id="id">Titlu</h1></div>
```

```
new Insertion.After("id", "<h1>Titlu</h1>");  
// <div id="header">Geek Meet</div><h1 id="id">Titlu</h1>
```

2. Prototype

* Iar acum, puțin Ajax

```
<form id="contact" action="hello.php" method="post">
  <input type="text" name="nume" id="nume" />
  <textarea name="mesaj" id="mesaj"></textarea>
  <input type="submit" name="submit" value="Trimite mesajul"
    onclick="return sendMessage()" />
</form>
```

```
new Ajax.Request("hello.php", {
  parameters : Form.serialize($("contact")),
  onSuccess : function(resp) {
    alert(resp.responseText);
  }
});
return false;
```

2. Prototype

* Iar acum, puțin Ajax

hello.php

```
<?php
```

```
if (isset($_SERVER['HTTP_X_REQUESTED_WITH']) &&  
    $_SERVER ['HTTP_X_REQUESTED_WITH'] == 'XMLHttpRequest')
```

```
{  
    echo 'Salut ' . $_POST['nume'] . ' – request Ajax';
```

```
}  
else  
{  
    echo 'Salut ' . $_POST['nume'] . ' – request normal, sincron';
```

```
}  
?>
```

Oare ce apare în alert(resp.responseText)? :)

2. Prototype

* Ajax în continuare

```
<div id="onlineUsers">2</div>
```

```
new Ajax.Updater("onlineUsers", "getUsers.php");
```

getUsers.php

```
<?php
```

```
// luăm numărul de utilizatori online din baza de date...
```

```
$users = 4;
```

```
echo $users;
```

```
?>
```

```
<div id="onlineUsers">4</div>
```

Întrebări?

3. Scriptaculous – efecte vizuale

* “Nu, nu e Flash”

Între `<head>` și `</head>`:

```
<script type="text/javascript" src="prototype.js"></script>
```

```
<script type="text/javascript" src="scriptaculous.js"></script>
```

Construit pe Prototype.

Prototype + Scriptaculous = JavaScript dream team

3. Scriptaculous – efecte vizuale

* Fade & Highlight

```
<div id="header">Geek Meet</div>
```

```
Effect.Fade("header");
```

```
Effect.Appear("header");
```

```
new Effect.Highlight(" header ", {  
  startcolor : '#999999',  
  endcolor : '#777777',  
  restorecolor : '#777777'  
});
```

3. Scriptaculous – efecte vizuale

* Slide & Blind

```
<div id="header">Geek Meet</div>
```

```
Effect.SlideUp("header");
```

```
Effect.SlideDown("header", { duration : 1.5 });
```

```
Effect.BlindUp("header");
```

```
Effect.BlindDown("header", { duration : 1.5 });
```

3. Scriptaculous – efecte vizuale

* Drag & Drop

```
<div id="dragMe">Geek Meet</div>
```

```
<div id="dropZone"></div>
```

```
new Draggable("dragMe", { revert : true });
```

```
Droppables.add("dropZone", {  
  onDrop: function(element) {  
    alert("la-l pe " + element.id + " de pe mine!");  
  }  
});
```

Întrebări?

4. XML, JSON, HTML, plain-text

* Unde le folosim?

În urma unui request Ajax, server-ul trimite un răspuns, care este apoi preluat de client.

4. XML, JSON, HTML, plain-text

* plain-text

Cea mai rapidă soluție când avem de trimis răspunsuri booleane sau string-uri simple.

```
if ($happyDay)
    echo "1";
else
    echo "0";
```

4. XML, JSON, HTML, plain-text

* plain-text

Clientul acționează în funcție de răspunsul server-ului.

```
new Ajax.Request("server.php", {
  onSuccess : function(resp) {
    if (resp.responseText == "1")
    {
      alert("OK, se face");
    }
    else
    {
      alert("Problema...");
    }
  }
});
```

4. XML, JSON, HTML, plain-text

* HTML

Uneori avem nevoie să construim bucăți de cod HTML dinamic, în funcție de răspunsul server-ului.

```
<div>  
  <h1>Titlu 1</h1>  
  <p>Descrierea cărții</p>  
</div>  
<div>  
  <h1>Titlu 2</h1>  
  <p>Descrierea cărții</p>  
</div>
```

4. XML, JSON, HTML, plain-text

* HTML

Clientul primește codul HTML ca șir de caractere și îl injectează într-un element HTML:

```
new Ajax.Request("server.php", {  
    onSuccess : function(resp) {  
        Element.update($("#container"), resp.responseText); }  
});
```

4. XML, JSON, HTML, plain-text

* XML

XML este lizibil și cunoscut.

```
<books>  
  <book>  
    <title>Titlu 1 </title>  
    <description>Descrierea cărții</description>  
  </book>  
  <book>  
    <title>Titlu 2</title>  
    <description>Descrierea cărții</description>  
  </book>  
</books>
```

4. XML, JSON, HTML, plain-text

* XML

JavaScript + XML = leeeent

```
new Ajax.Request("server.php", {
  onSuccess : function(resp) {
    var books = resp.responseXML.getElementsByTagName("book");
    for (i = 0; i < books.length; i++)
    {
      var theDiv = document.createElement("div");
      var theH1 = document.createElement("h1");
      theH1.appendChild(document.createTextNode(
        books[i].getElementsByTagName("title")[0].firstChild.nodeValue));
      theDiv.appendChild(theH1);
      // iar in continuare acelasi lucru pentru <p> - descriere
    }
  }
});
```

4. XML, JSON, HTML, plain-text

* JSON

JSON = JavaScript Object Notation

```
{ "books" : [ {  
    "book" : {  
        "title" : "Titlu 1",  
        "description" : "Descrierea cărții"  
    }  
},  
    "book" : {  
        "title" : "Titlu 2",  
        "description" : "Descrierea cărții"  
    }  
}  
]
```

4. XML, JSON, HTML, plain-text

* JSON

JavaScript + XML = leeeent

```
new Ajax.Request("server.php", {
  onSuccess : function(resp) {
    var myBooks = eval('(' + resp.responseText + ')');
    for (i = 0; i < books.length; i++)
    {
      var theDiv = document.createElement("div");
      var theH1 = document.createElement("h1");
      theH1.appendChild(document.createTextNode(
        myBooks.books[i].book.title));
      theDiv.appendChild(theH1);
      // iar in continuare acelasi lucru pentru <p> - descriere
    }
  }
});
```

4. XML, JSON, HTML, plain-text

* Concluzii

plain-text e recomandat pentru mesaje scurte. E rapid.

Bucățile de HTML injectate în innerHTML (sau Element.update) nu sunt agreate de puriștii JavaScript/DOM. Pentru structuri simple, însă, care nu conțin formulare, HTML e varianta optimă.

4. XML, JSON, HTML, plain-text

* Concluzii

XML este ușor de generat, dar relativ urât de parcurs în JavaScript. Este de asemenea lent.

JSON are o sintaxă puțin mai greu de prins, dar parcurgerea lui se face rapid în JavaScript, fiind o facilitate nativă și este, astfel, de preferat față de XML.

Întrebări?

5. Ajax shopping cart demo

* Download

www.exigo.ro/work/ajax-cart/