AJAX

Co, gdzie, jak, kiedy?

- kiedy
- co
- gdzie
- jak
- jak
- I jeszcze trochę jak

kiedy/gdzie/co

czyli trochę historii

Aaa!!! We're in the 90's!!!

- stare dobre (albo złe… a może jeszcze inne?) www
- Microsoft? That's impossible, Holmes!
- a jednak…

Coś tu się rusza…

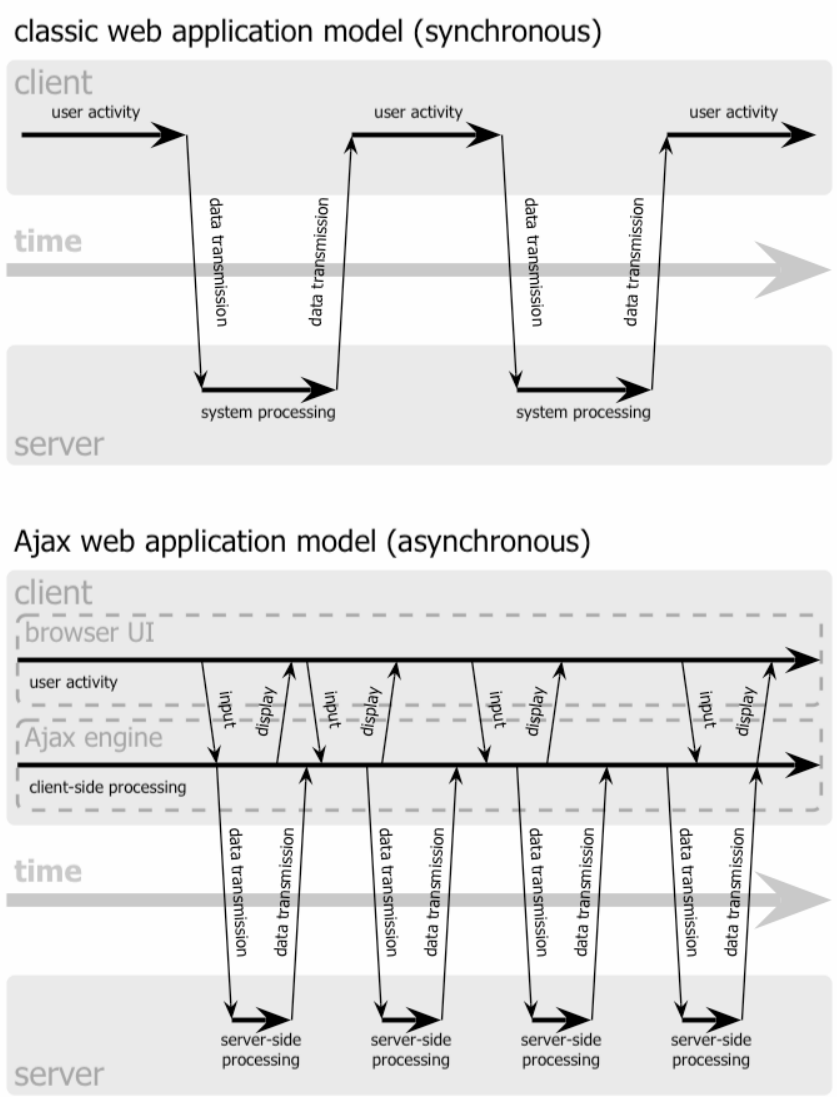
- powolutku, niemrawo pojawiają się kolejne strony
- i wtedy nadchodzi Google (konkretniej GMail, za chwilkę Maps)

AJAX: A New Approach to Web Applications

- 18 luty 2005 (!) – artykuł Jasse Jamesa Garretta, *adaptive path*
- pierwsze zdefiniowanie pojęcia AJAX
- Asynchronous JavaScript + XML

Gdzie jest kot pogrzebany?

- słowo klucz: asynchronous



Więcej...

- http://www.adaptivepath.com/publications/essays/archives/000385.php

jak

cz. 1, czyli czysty JavaScript

You wanna get dirty?

- jak każda fajna technologia, nie daje się w tym pisać!
- write once, run anywhere? akurat
  - `xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");`
  - `xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");`
  - `xmlhttp = new XMLHttpRequest();`
  - `xmlhttp = window.createRequest();`

Dirty, dirty, dirty

```
var xmlhttp=false;
/*@cc_on @*/
/*@if (@_jscript_version >= 5)
// JScript gives us Conditional compilation, we can cope with old IE versions.
// and security blocked creation of the objects.
 try {
  xmlhttp = new ActiveXObject("Msxml2.XMLHTTP");
 } catch (e) {
  try {
   xmlhttp = new ActiveXObject("Microsoft.XMLHTTP");
  } catch (E) {
   xmlhttp = false;
  }
 }
@end @*/
if (!xmlhttp && typeof XMLHttpRequest!='undefined') {
     try {
             xmlhttp = new XMLHttpRequest();
     } catch (e) {
             xmlhttp=false;
     }
}
if (!xmlhttp && window.createRequest) {
     try {
             xmlhttp = window.createRequest();
     } catch (e) {
             xmlhttp=false;
     }
}
```

Mamy XMLHttpRequest…

- jest lepiej

```
xmlhttp.open("GET", "test.txt",true);
xmlhttp.onreadystatechange=function() {
 if (xmlhttp.readyState==4) {
  alert(xmlhttp.responseText)
 }
}
xmlhttp.send(null)
```

- co możemy dostać?
  - normalny tekst – `xmlhttp.responseText`
  - xml – `xmlhttp.responseXML` (ale serwer musi wysłać dane jako `mime: text/xml`)

Info + linki

- http://www.xulplanet.com/references/objref/XMLHttpRequest.html
- http://developer.apple.com/internet/webcontent/xmlhttpreq.html
- http://jibbering.com/2002/4/httprequest.html
- http://www.sitepoint.com/article/remote-scripting-ajax

jak

cz. 2, czyli prototype.js

Welcome back to the XXI century

- w3.org twierdzi, że JavaScript jest językiem obiektowym…
  - bez enkapsulacji?
  - bez dziedziczenia?
- próba zrobienia z JavaScriptu czegoś chociaż zbliżonego do języka obiektowego

Podstawy obiektowości

- Class.create();

```
MyClass = Class.create();
MyClass.prototype = {
    initialize: function(a, b) {
        this.a = a;
        this.b = b;
this.secondfunc();
    },
    secondfunc: function(){
this.t = 'secondlife';
}
}

var mc = new MyClass("foo", "bar");
alert(mc.t)
```

Prawie rodzina

- Object.extend(dst, src)

```
objA = {name: "Joe", age: "12"};
objB = {name: "Tom"};
Object.extend(objA, objB);
alert(objA.name); // "Tom"
```

```
objA = {name: "Joe", age: "12"};
objC = Object.extend({}, objA);
```

```
objA = {name: "Joe", age: "12", car:{make: "Honda"}};
objC = Object.extend({}, objA);
objC.car.make = "Toyota";
alert(objA.car.make); // "Toyota"
```

```
objA = {name: "Joe", age: "12"};
objC = Object.extend(objA, {name: "Tom"});
objD = Object.extend(objA, {name: "Jim"});

alert(objC.name); // "Tom"
alert(objD.name); // "Jim"
alert(objA.name); // "Jim"
```

Czyżby czyżyk? Kolekcje?

- Enumerable, Array, Hash
- `$A(obj); $H(obj);`
- iteratory
  - each, select, invoke, find, etc.

```
// alerts "a is at 0" then "b is at 1" then "c is at 2"
["a", "b", "c"].each(function(item, index) {
 alert(item + " is at " + index);
});
```

```
// [80,50]
[1, 80, 3, 50].select(function(item) {
 return (item > 20);
});
```

AJAX

- Ajax.Request(url, opts)
  - url – normalny adres
  - opts – ogólnie dowolny obiekt, o ile posiada pola:
    - method
    - parameters
    - asynchronous
    - onXXX (XXX – Loaded, Complete, Success, Failure, …)
    - …

Ajax.Request();

```
new Ajax.Request('/foo/bar', {method:'post', postBody:'thisvar=true&thatvar=Howdy'});
```

```
var handlerFunc = function(t) {
    alert(t.responseText);
}

var errFunc = function(t) {
    alert('Error ' + t.status + ' -- ' + t.statusText);
}

new Ajax.Request('/foo/bar', {parameters:'thisvar=true&thatvar=Howdy', onSuccess:handlerFunc, onFailure:errFunc});
```

```
var opt = {
    method: 'post',
    postBody: 'thisvar=true&thatvar=Howdy&theothervar=2112',
    onSuccess: function(t) {
        alert(t.responseText);
    },
    on404: function(t) {
        alert('Error 404: location "' + t.statusText + '" was not found.');
    },
    onFailure: function(t) {
        alert('Error ' + t.status + ' -- ' + t.statusText);
    }
}

new Ajax.Request('/foo/bar', opt);
```

Ajax.Updater

- sam zmienia wskazany element

```
new Ajax.Updater(container, url, options);
```

  - container – id, element lub obiekt z polami success i failure
  - reszta bez zmian

Jak działa

```
new Ajax.Updater('mydiv', '/foo/bar', {asynchronous:true, evalScripts:true});
// this will evaluate any scripts in <script></script> blocks.
```

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

<script language="JavaScript" type="text/javascript">
var ajax;
function mydate() {
 ajax =  new Ajax.Updater(
         'datestr',       // DIV id must be declared before the method was called
         'date.cgi',      // URL
         {                // options
         method:'get'
             });

}
</script>

date is now: <div id="datestr">n/a</div>

<script language="JavaScript" type="text/javascript">
    mydate();
</script>
```

I jeszcze jedna

- Ajax.PeriodicalUpdater(container, url, options)
  - gdzie jest częstotliwość wykonywania???

Działanie

```
new Ajax.PeriodicalUpdater('mydiv', '/foo/bar', {asynchronous:true, frequency:2});
```

- fajna opcja: decay

```
new Ajax.PeriodicalUpdater('mydiv', '/foo/bar', {asynchronous:true, frequency:2, decay:2});
```

I teraz to już ostatnia rzecz...

- Ajax.Responders
  - rejestrowanie funkcji do wszystkich wywołań AJAX.

```
Ajax.Responders.register({
 onCreate : showLoader,
 onComplete : hideLoader
});
```

Przydatne stronki

- http://www.sergiopereira.com/articles/prototype.js.html
- http://wiki.script.aculo.us/scriptaculous/show/Prototype
- http://www.sitepoint.com/article/painless-javascript-prototype
- http://encytemedia.com/blog/articles/2005/12/07/prototype-meets-ruby-a-look-at-enumerable-array-and-hash
- http://www.snook.ca/archives/000531.php
- http://particletree.com/features/quick-guide-to-prototype/

jak

cz. 3, czyli trochę o niczym

script.aculo.us

- po prawdzie bardziej biblioteka efektów niż biblioteka AJAXu
- bazuje na prototype.js (kiedyś była wewnątrz prototype), bardzo często widziane razem

Warto wiedzieć

- test units
  - gotowa strona do pobrania, odpala testy
  - Test.Unit.Runner()
  - Test.Unit.TestCase
- słaba dokumentacja, ale (podobno) duże możliwości

Link

- http://script.aculo.us/

i znowu jak

cz. 4, która znowu nic nowego nie wniesie

Rico

- oparty na prototype.js
- zawiera dużą kolekcję efektów
- inne podejście do AJAXu

3 kroki do AJAXu

- powiedzieć, co zrobić w momencie żądania
- co zrobić z odpowiedzią
- wywołać żądania dla podanych zdarzeń

Czym się różni

- ustalony format odpowiedzi z serwera
- możliwość zawarcia w jednej fizycznej odpowiedzi kilku logicznych.

Tradycyjnie... link

- http://openrico.org/rico/home.page

inne podejście

czyli część, której nie będzie

Ruby on Rails

- pełny zrąb do tworzenia aplikacji webowych
- oparty na prototype
- szybko zdobywający rynek

Co takiego fajnego?

- całkowicie oparty na Rubym – w pełni obiektowy
- wbudowane mechanizmy łączności w bazami danych
- wbudowana możliwość AJAXu
- Convention over Configuration

Convention over Configuration

- z góry ustalone zasady nadawania nazw:
  - nazw plików
  - nazwy klas i metod
  - tabele w bazie danych
  - umiejscowienia plików
- MVC
- brak plików konfiguracyjnych

Must-see

- http://www.rubyonrails.org/
- http://www.onlamp.com/pub/a/onlamp/2005/06/09/rails_ajax.html
- http://poignantguide.net/ruby/

a może jednak Java?

czyli część ostatnia

DWR

- tworzony tylko i wyłącznie z myślą o AJAXie
- mapowanie Javy z serwera do JavaScriptu u klienta (!)
- wymaga kontenera serwletów na serwerze

Co zrobić, żeby działało

- ściągnąć `dwr.jar` i zainstalować w katalogu `WEB-INF/lib`
- zmodyfikować plik `WEB-INF/web.xml` oraz dodać plik `WEB-INF/dwr.xml`
- wejść na adres `[serwer]/[webapp]/dwr`, powinna się pokazać lista udostępnionych klas

No to szybko zrobimy czat :D

- nasza strona:

```
<p>Messages:</p>
<div id="chatlog"></div>
<p>
  Your Message:
  <input id="text"/>
  <input type="button" value="Send"
      onclick="sendMessage()"/>
</p>
```

Co na serwerze?

- dwie klasy:

```
public class Chat
{
    public List addMessage(String text)
    {
        if (text != null &&
            text.trim().length() > 0)
        {
            messages.addFirst(new Message(text));
            while (messages.size() > 10)
            {
                messages.removeLast();
            }
        }

        return messages;
    }

    public List getMessages()
    {
        return messages;
    }

    static LinkedList messages =new LinkedList();
}
```

```
public class Message
{
    public Message(String newtext)
    {
        text = newtext;
        if (text.length() > 256)
        {
            text = text.substring(0, 256);
        }
        text = text.replace('<', '[');
        text = text.replace('&', '_');
    }

    public long getId()
    {
        return id;
    }

    public String getText()
    {
        return text;
    }

    long id = System.currentTimeMillis();
    String text;
}
```

Konfigurujemy serwer

- najlepsza metoda programowania – copy&paste

```
<servlet>
  <servlet-name>dwr-invoker</servlet-name>
  <display-name>DWR Servlet</display-name>
  <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
  <init-param>
    <param-name>debug</param-name>
    <param-value>true</param-value>
  </init-param>
</servlet>
<servlet-mapping>
  <servlet-name>dwr-invoker</servlet-name>
  <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE dwr PUBLIC
    "-//GetAhead Limited//DTD Direct Web Remoting 1.0//EN"
    "http://www.getahead.ltd.uk/dwr/dwr10.dtd">

<dwr>
  <allow>
    <create creator="new" javascript="Chat">
      <param name="class" value="[your.package].Chat"/>
    </create>
    <convert converter="bean" match="[your.package].Message"/>
  </allow>
</dwr>
```

Działaj strono, działaj

- dodajemy JavaScript

```
<script type='text/javascript'
    src='/[YOUR-WEB-APP]/dwr/engine.js'></script>
<script type='text/javascript'
    src='/[YOUR-WEB-APP]/dwr/interface/Chat.js'></script>
<script type='text/javascript'
    src='/[YOUR-WEB-APP]/dwr/util.js'></script>
```

- co siedzi w Chat.js?

```
Chat.addMessage = function(callback, p0) { ... }
Chat.getMessages = function(callback) { ... }
```

Ostatnie szlify

- dwie małe funkcje

```
function sendMessage()
{
    var text = DWRUtil.getValue("text");
    DWRUtil.setValue("text", "");
    Chat.addMessage(gotMessages, text);
}
```

```
function gotMessages(messages)
{
    var chatlog = "";
    for (var data in messages)
    {
        chatlog = "<div>" + messages[data].text +
            "</div>" + chatlog;
    }
    DWRUtil.setValue("chatlog", chatlog);
}
```

Link

- http://getahead.ltd.uk/dwr/
- http://today.java.net/pub/a/today/2005/08/25/dwr.html

koniec