AJAX

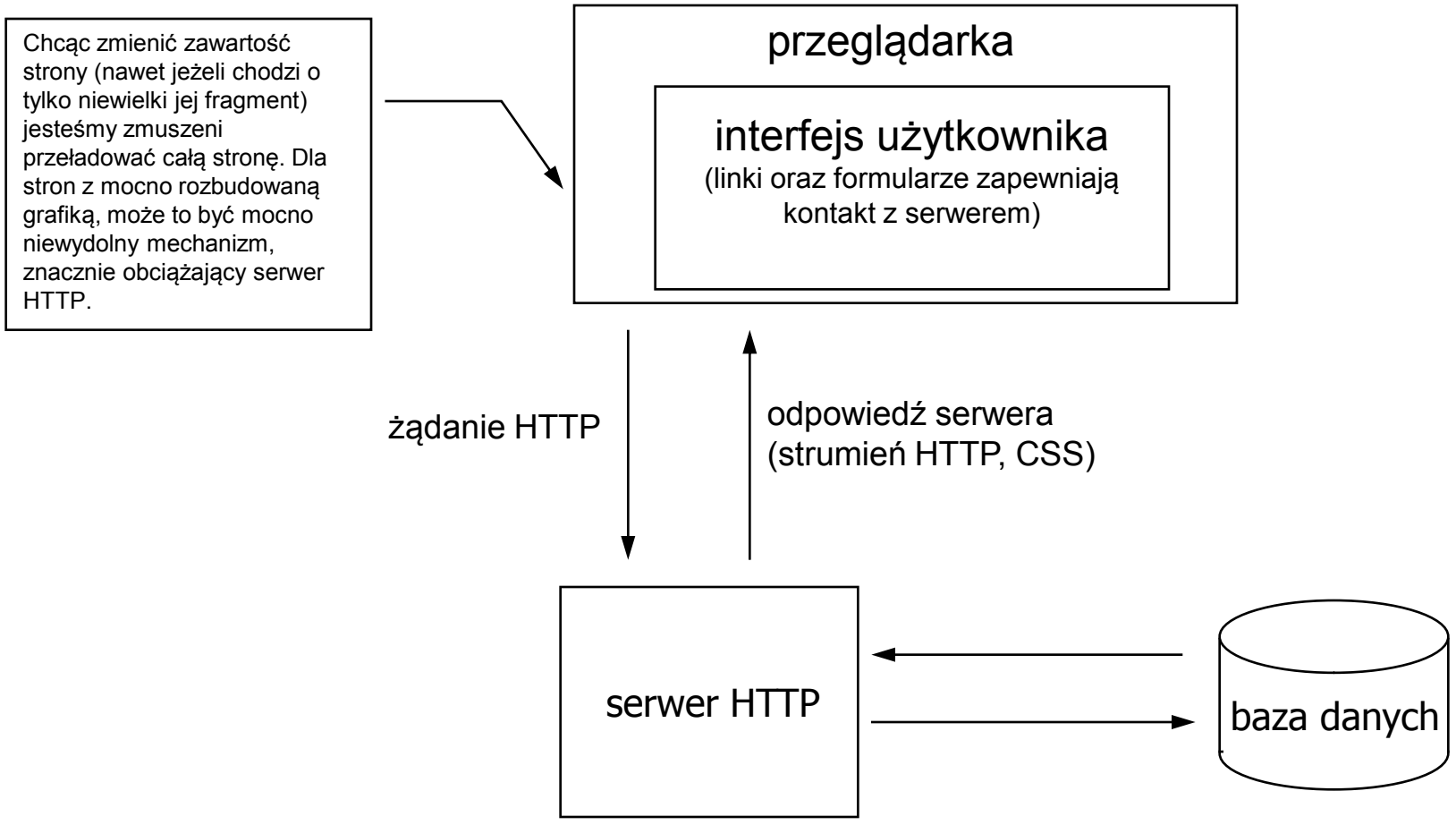
Asynchronous **J**avaScript **A**nd **X**ML

Artur Gramacki

Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych
A.Gramacki@issi.uz.zgora.pl

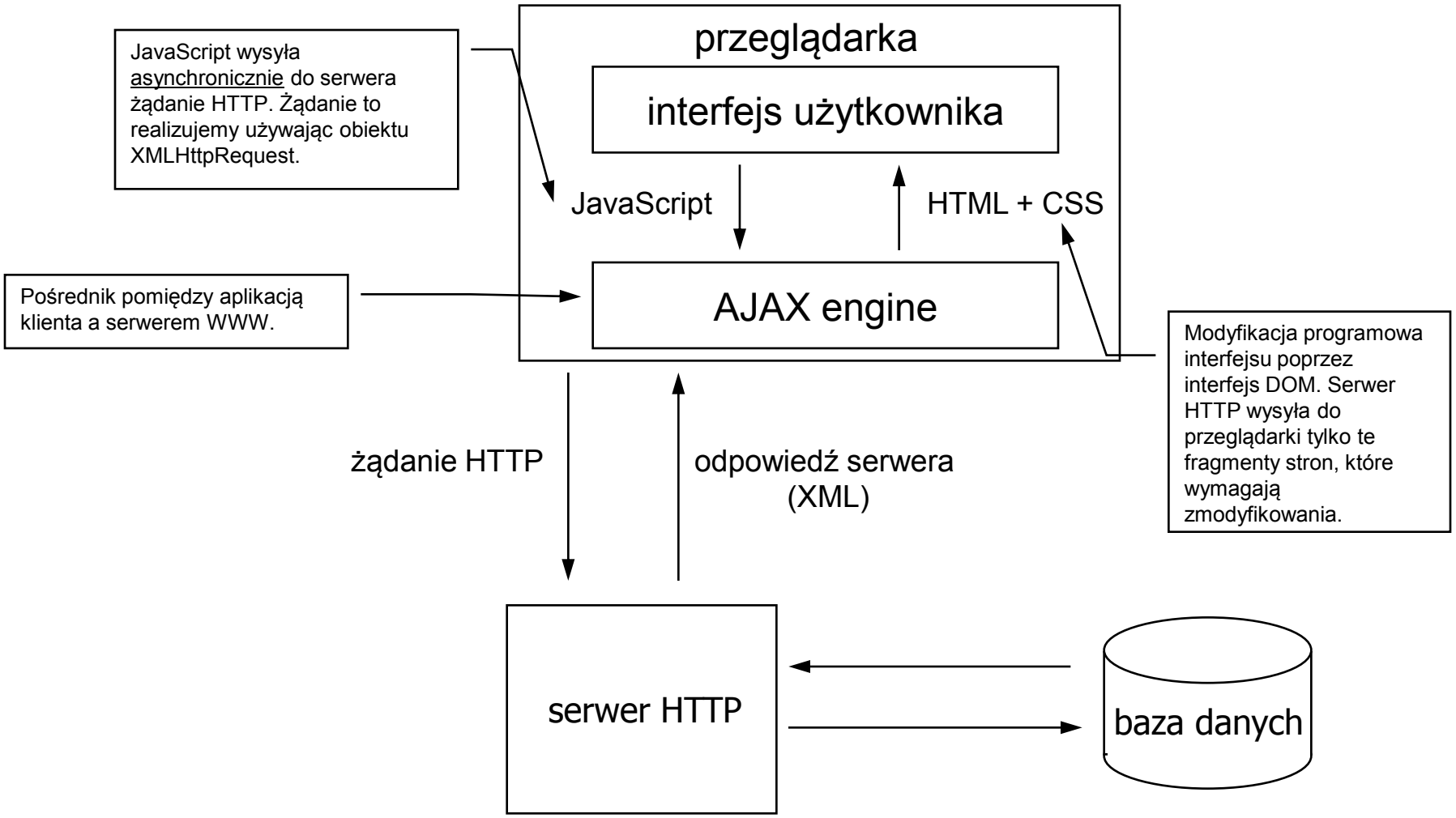
korzystano z opracowania: *http://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_aplikacje_internetowe*

Klasyczna aplikacja internetowa



korzystano z opracowania: *http://wazniak.mimuw.edu.pl/index.php?title=Zaawansowane_aplikacje_internetowe*

Aplikacja internetowa oparta o AJAX

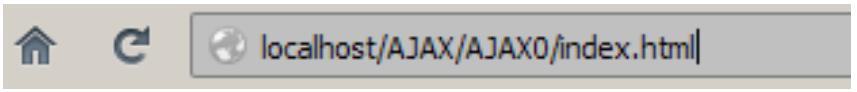


Technologie wchodzące w skład AJAX

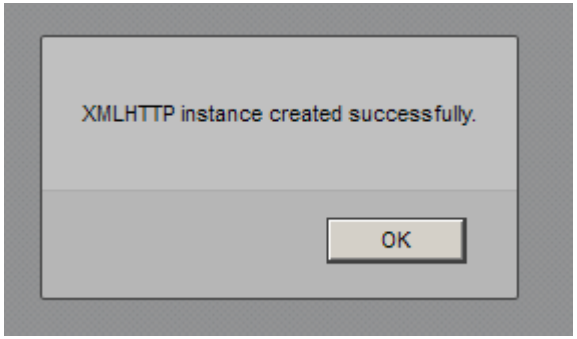
- HTML + CSS (prezentacja danych)
- JavaScript (do implementacji części logiki aplikacji po stronie klienta)
- DOM (Document Object Model; do programowej modyfikacji elementów HTML w przeglądarce)
- XML (jako format wysyłania danych)
  - alternatywnie można używać formatu JSON lub plików z czystym tekstem
- Obiekt XMLHttpRequest (do wysyłania asynchronicznych żądań do serwera HTTP)
  - żądania te wysyła odpowiedni kod w języku JavaScipt

Przykład 1

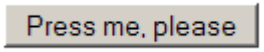
- Wpisujemy adres w przeglądarce



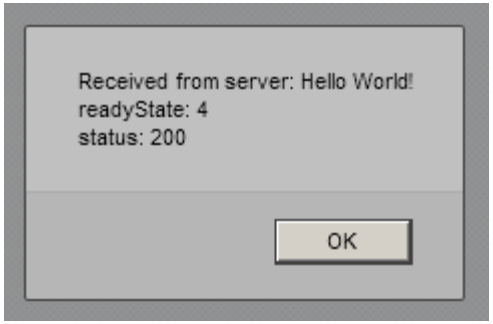
- Tworzony jest obiekt XMLHttpRequest, który zapewnia asynchroniczna komunikację aplikacji z serwerem HTTP



- Interfejs aplikacji (tylko przycisk)



- Asynchronicznie wysyłamy żądanie (pobieramy z serwera plik i odsyłamy go do klienta, gdzie jego zawartość jest wyświetlana)



Przykład 1, kod

```
<html><head><title>AJAX demo (c) by Artur Gramacki</title>
<script type="text/javascript" language="javascript">
```

```
function createXMLHttpRequestObject() {
  var xhr = false;
  if (window.XMLHttpRequest) { // Mozilla, Safari, Chrome, Opera
    xhr = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) { // IE 8 and older
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
  }
  if (!xhr) {
    alert('Cannot create an XMLHTTP instance.');
    xhr = false;
  }
  else {
    alert('XMLHTTP instance created successfully.');
    return xhr;
  }
}
```

```
var xhrRequest = createXMLHttpRequestObject();

w tym miejscu jest reszta kodów, pokazano je na następnym slajdzie

</script>
```

```
</head><body>
<button type="button" onclick="sendRequest()">Press me, please</button>
</body></html>
```

W przeglądarce musi być włączona obsługa JavaScript.

Funkcja, której zadaniem jest utworzenie obiektu XMLHttpRequest . W IE obiekt ten dostępny jest w postaci obiektu ACtiveX. Trzeba pamiętać, aby w przeglądarce IE włączyć możliwość uruchamiania komponentów ActiveX.

Alerty tylko w celach demonstracyjnych. Normalnie raczej nie będziemy ich włączać w aplikacjach (mogą denerwować użytkowników).

Powołanie obiektu XMLHttpRequest do życia.

Po kliknięciu nastąpi asynchroniczne wysłanie żądania do serwer.

Przykład 1, kod

```
function sendRequest() {
  xhrRequest.onreadystatechange = handleResponse;
  xhrRequest.open("GET", "file.txt", true);
  xhrRequest.send();
}
```

Funkcja realizująca asynchronicznego wysłania żądania HTTP.
Wywołujemy dwie metody utworzonego wcześniej obiektu XMLHttpRequest: open() oraz send().

Wskazanie, która funkcja będzie prowadziła nasłuch obiektu XMLHttpRequest.

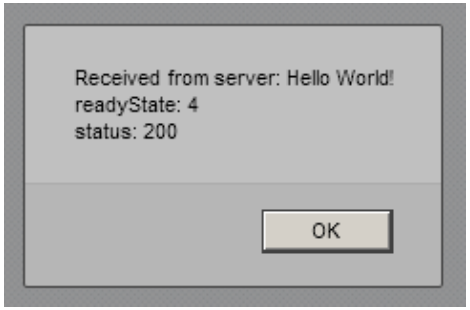
Inicjalizacja żądania.

Aktywacja połączenia i realizacja żądania (tu: pobranie z serwera zawartości pliku tekstowego i wysłanie go asynchronicznie do aplikacji klienta).

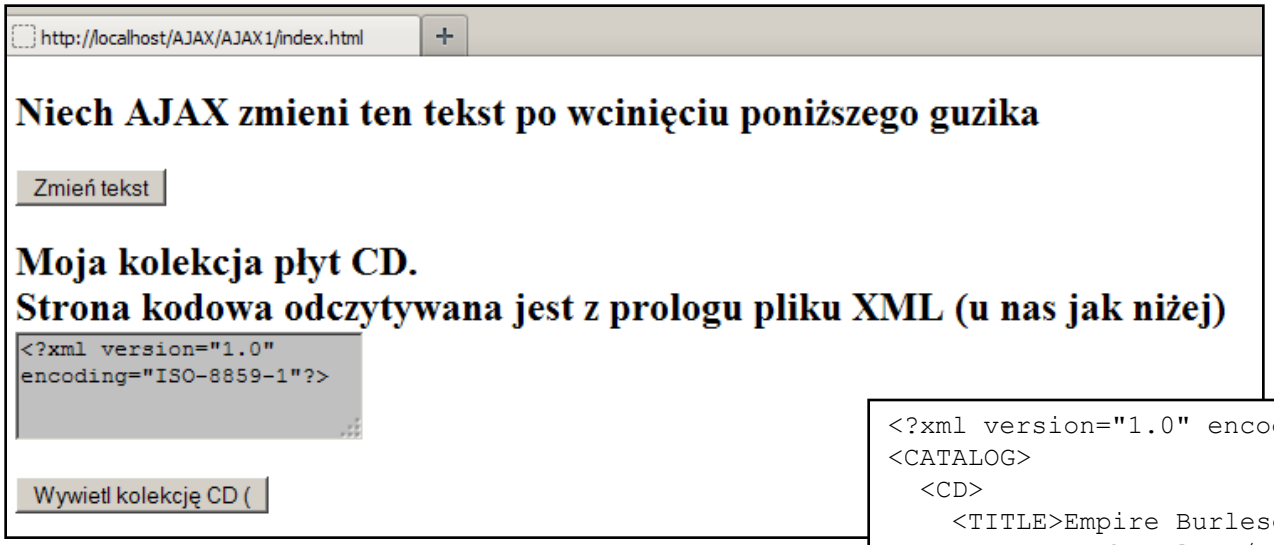
```
function handleResponse() {
  if (xhrRequest.readyState == 4) {
    if (xhrRequest.status == 200) {
      alert(
        "Received from server: " +
        xhrRequest.responseText +
        "\nreadyState: " +  xhrRequest.readyState +
        "\nstatus: " + xhrRequest.status);
    }
    else {
      alert("There was a problem with the request.");
    }
  }
}
```

Funkcja realizująca nasłuch stanu obiektu XMLHttpRequest. Konieczna, aby aplikacja mogła zareagować na zakończenie pobierania danych z serwera.

4 – completed
200 – OK.



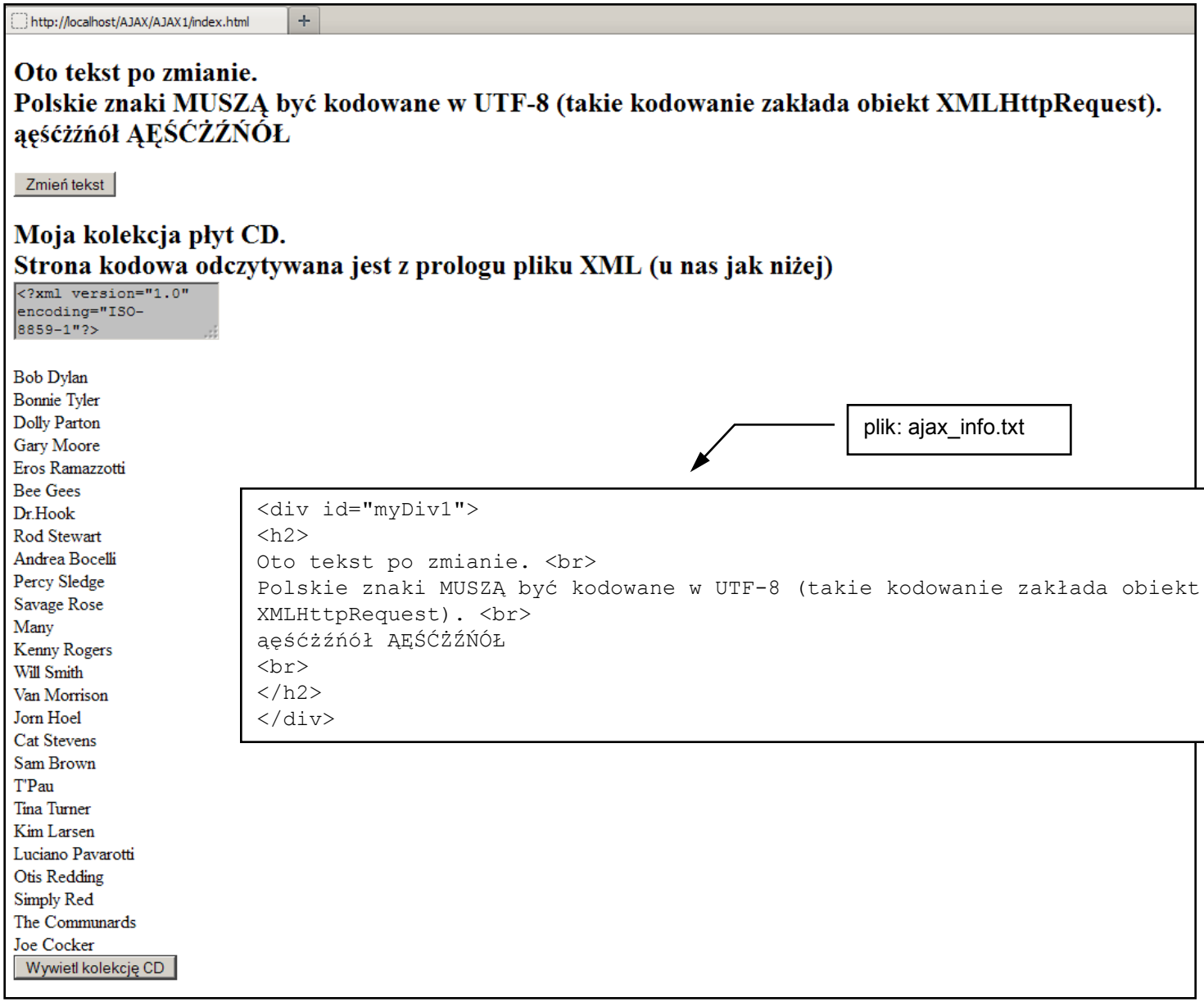
Przykład 2



plik: cd_catalog.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>

...

</CATALOG>
```

Przykład 2


http://localhost/AJAX/AJAX1/index.html
Oto tekst po zmianie.
Polskie znaki MUSZĄ być kodowane w UTF-8 (takie kodowanie zakłada obiekt XMLHttpRequest).
ąęśćżźńół ĄĘŚĆŻŹŃÓŁ
Zmień tekst
Moja kolekcja płyt CD.
Strona kodowa odczytywana jest z prologu pliku XML (u nas jak niżej)
<?xml version="1.0" encoding="ISO-8859-1"?>
Bob Dylan
Bonnie Tyler
Dolly Parton
Gary Moore
Eros Ramazzotti
Bee Gees
Dr.Hook
Rod Stewart
Andrea Bocelli
Percy Sledge
Savage Rose
Many
Kenny Rogers
Will Smith
Van Morrison
Jorn Hoel
Cat Stevens
Sam Brown
T'Pau
Tina Turner
Kim Larsen
Luciano Pavarotti
Otis Redding
Simply Red
The Communards
Joe Cocker
Wywietl kolekcję CD
plik: ajax_info.txt

<h2>
Oto tekst po zmianie.
Polskie znaki MUSZĄ być kodowane w UTF-8 (takie kodowanie zakłada obiekt XMLHttpRequest).
ąęśćżźńół ĄĘŚĆŻŹŃÓŁ

</h2>

Przykład 2, kod

```
function createXMLHttpRequestObject() {
  var xhr = false;
  if (window.XMLHttpRequest) {
    xhr = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) {
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
  }
    return xhr;
  }
}
```

Identycznie jak w przykładzie 1.

```
var http = createXMLHttpRequestObject();
```

```
// Get text file
function sendRequest_1() {
  http.open("GET", "ajax_info.txt", true);
  http.onreadystatechange = handleResponse_1;
  http.send();
}

// Get XML file
function sendRequest_2() {
  http.open("GET", "cd_catalog.xml", true);
  http.onreadystatechange = handleResponse_2;
  http.send();
}
```

Plik tekstowy.

Plik w formacie XML.

Przykład 2, kod

```
// responseText
function handleResponse_1() {
  if (http.readyState == 4 && http.status == 200) {
    document.getElementById("myDiv_1").innerHTML=http.responseText;
  }
}

// responseXML
function handleResponse_2() {
  if (http.readyState == 4 && http.status == 200){
    xmlDoc = http.responseXML;
    txt = "";
    x = xmlDoc.getElementsByTagName("ARTIST");
    for (i=0; i<x.length; i++) {
      txt = txt + x[i].childNodes[0].nodeValue + "<br>";
    }
    document.getElementById("myDiv_2").innerHTML=txt;
  }
}
```

Wstawienie do struktury dokumentu DOM zmodyfikowanej zawartości.

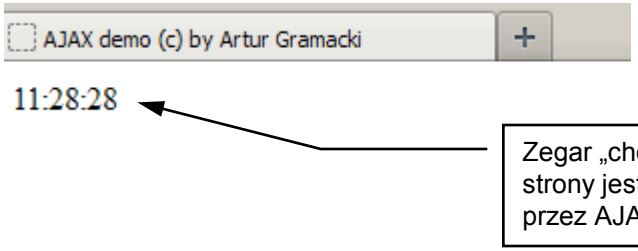
```
<div id="myDiv1">
<h2>
Oto tekst po zmianie. <br>
Polskie znaki MUSZĄ być kodowane w UTF-8
(takie kodowanie zakłada obiekt XMLHttpRequest).
<br>
ąęśćżźńół ĄĘŚĆŻŹŃÓŁ
<br>
</h2>
</div>
```

„Wstrzyknięcie” do struktury dokumentu DOM nowej zawartości.

```
<body>
<div id="myDiv_1"><h2>Niech AJAX zmieni ten tekst po wciśnięciu poniższego guzika</h2></div>
<button type="button" onclick="sendRequest_1()">Zmień tekst</button>
<h2>Moja kolekcja płyt CD.<br>
Strona kodowa odczytywana jest z prologu pliku XML (u nas jak niżej)<br>
<TEXTAREA><?xml version="1.0" encoding="ISO-8859-1"?></TEXTAREA>
</h2>
<div id="myDiv_2"></div>
<button type="button" onclick="sendRequest_2()">Wyświetl kolekcję CD</button>
</body>
</html>
```

Znacznik służy do strukturalizacji dokumenty HTML.

Przykład 3



- Aplikacja składa się z 4 plików
  - ajax.js
  - getTime.php
  - index.html
  - time.js

```
function createXMLHttpRequestObject() {
  if (window.XMLHttpRequest) {
    xhr = new XMLHttpRequest();
  }
  else if (window.ActiveXObject) {
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
  }
}
```

```
<?php echo date('H:i:s'); ?>
```

```
<html>
<head>
<title>AJAX demo (c) by Artur Gramacki</title>

<script type="text/javascript" src="ajax.js"> </script>
<script type="text/javascript" src="time.js"> </script>

</head>
<body>
<span id="currentTime"></span>
</body>
</html>
```

Przykład 3, kody (time.js)

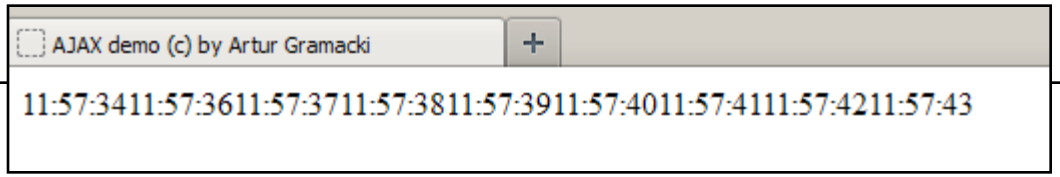
```
xhrRequest = createXMLHttpRequestObject();
```

```
function getTime() {
  // Bypassing the cache
  // https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started
  // Note 2: If you do not set header Cache-Control: no-cache the browser will cache
  // the response and never re-submit the request, making debugging "challenging."
  // You can also append an always-diferent aditional GET parameter, like the timestamp or a random number
  xhrRequest.open("GET", "getTime.php?temp=" + new Date() . getTime(), true);
  xhrRequest.onreadystatechange = updateTime;
  xhrRequest.send(null);
}
```

```
window.onload = getTime;
```

```
function updateTime() {
  if (xhrRequest.readyState == 4) {
    var resp = xhrRequest.responseText;
    var curTime = document.getElementById("currentTime");

    if (curTime.firstChild) {
      curTime.removeChild(curTime.firstChild);
    }

    curTime.appendChild(document.createTextNode(resp));
    var tmId = setTimeout(getTime, 1000);
  }
}
```

Ten IF zapobiega dodawaniu kolejnych elementów potomnych do struktury dokumentu DOM. Gdy go usuniemy strona będzie wyglądała jak poniżej.



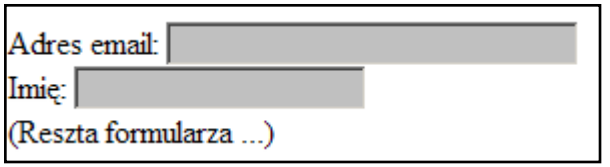
Przykład 4

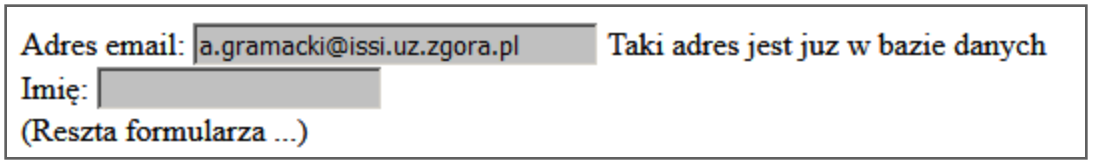
na podstawie: http://www.oracle.com/technetwork/articles/ullman-ajax-096274.html

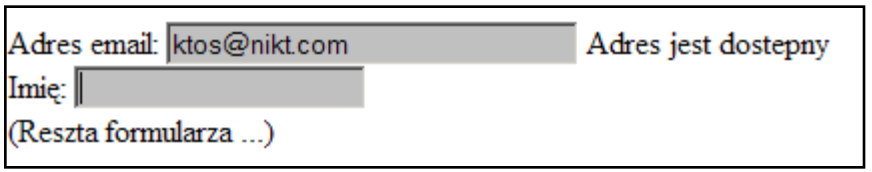
- Aplikacja korzysta z bazy danych (tu Oracle, ale może być dowolna inna)

```
SQL> select * from users;

EMAIL
--------------------------------
a.gramacki@issi.uz.zgora.pl
j.gramacki@ck.uz.zgora.pl
a.gramacki@gmail.com

SQL>
```

- Żądanie AJAX wysyłane po zmianie zawartości pola formularza (OnChange)
- Aplikacja składa się z 2 plików
  - ajax.php
  - index.html







Przykład 4, kody (index.html)

```
function createXMLHttpRequestObject() {
  var xhr;
  if (navigator.appName == "Microsoft Internet Explorer") {
    xhr = new ActiveXObject("Microsoft.XMLHTTP");
  } else {
    xhr = new XMLHttpRequest();
  }
  return xhr;
}
```

```
var http = createXMLHttpRequestObject();
```

```
function sendRequest(email) {
  http.open('get', 'ajax.php?email=' + encodeURIComponent(email));
  http.onreadystatechange = handleResponse;
  http.send(null);
}
```

Funkcja jako parametr dostaje adres email, który ma podlegać weryfikacji. Adres pobierany jest z pola formularza.

Przekształca adres email do bezpiecznej dla URL-a postaci.

```
function handleResponse() {
  if (http.readyState == 4 && http.status == 200) {
    document.getElementById('email_label').innerHTML = http.responseText;
  }
}
```

Gdy zmieni się zawartość pola następuje wysłanie żądania AJAX. Wywoływany jest skrypt PHP (ajax.php).

Zawartość tego znacznika będzie modyfikowana.

```
<form action="somepage.php" method="post">
Adres email:
<input name="email" type="text" size="30" maxlength="60"
       OnChange="sendRequest(this.form.email.value)" />
<span id="email_label"></span>
<br>
Imię: <input name="first_name" type="text" size="20" maxlength="20" />
<br>
(Reszta formularza ...)
</form>
```

Przykład 4, kody (ajax.php; wersja dla ORACLE)

```
<?php
// No HTML required by this script! Validate that the page received $_GET['email']:
if (isset($_GET['email'])) {
  $c = oci_connect ('artur', 'artur', 'localhost:1531/ee')
  or die('Unable to connect to the database. Error: <pre>' .
          print_r(oci_error(),1) . '</pre>');

  // Define the query.
  $q = "SELECT COUNT(*) AS num_rows FROM users WHERE email LIKE'{$_GET['email']}'";

  // Parse the query.
  $s = oci_parse($c, $q);

  // Initialize the PHP variable:
  $rows = 0;

  // Bind the output to $rows:
  oci_define_by_name($s, "NUM_ROWS", $rows);

  // Execute the query.
  oci_execute($s);

  // Fetch the results.
  oci_fetch($s);

  // Close the connection.
  oci_close($c);

  // Return a message indicating the status.
  if ($rows > 0) {
    echo 'Taki adres jest juz w bazie danych';
  } else {
  echo 'Adres jest dostepny';
  }
}
?>
```

Tylko, gdy w polu formularza coś wpisano.

Parametry połączeniowe dobrze jest „wynieść” poza ten plik do innego pliku, odpowiednio dobrze chronionego.

W Oracle musi być z dużych liter.

Typowa procedura pobierania danych z bazy (tu Oracle):
- zainicjowanie sesji z bazą danych
- tworzenie zapytania
- parsowanie zapytania
- związanie zapytania ze zmienną
- wykonanie zapytania
- pobranie wyników
- zamknięcie połączenia

Przykład 4, kody (ajax.php; wersja dla MySQL)

```
<?php
// No HTML required by this script!
// Validate that the page received $_GET['email']:
if (isset($_GET['email'])) {

  // Connect to the database.
  $c = mysqli_connect ('localhost', 'artur', 'artur')
       or die ('Nie można połączyć się z MySQL.');

  mysqli_select_db ($c, 'artur')
       or die ('Nie można połączyć się z bazą danych.');

  // Define the query.
  $q = mysqli_query ($c, "SELECT COUNT(*) AS num_rows
       FROM users WHERE email LIKE '{$_GET['email']}'")
       or die ('Błąd w zapytaniu do bazy.');

  // Fetch the results.
  $f = mysqli_fetch_array ($q, MYSQLI_NUM);
  // Return a message indicating the status.
  if ($f[0] > 0) {
    echo 'Taki adres jest juz w bazie danych';
  } else {
    echo 'Adres jest dostepny';
  }

  // Close connection.
  mysqli_close($c);
}
?>
```

Tylko, gdy w polu formularza coś wpisano.

Parametry połączeniowe dobrze jest „wynieść” poza ten plik do innego pliku, odpowiednio dobrze chronionego.

Typowa procedura pobierania danych z bazy (tu MySQL):
- zainicjowanie sesji z bazą danych (mysqli_connect)
- wybór właściwej bazy danych (mysqli_select_db)
- tworzenie zapytania (mysqli_query)
- wykonanie zapytania i pobranie wyników (mysqli_fetch_array)
- zamknięcie połączenia (mysqli_close)