Berkenalan dengan AJAX

Jerry Peter
Jerry.peter@gmail.com
http://www.ruangkecil.or.id

Istilah AJAX (Asynchronous JavaScript And XML) dalam pengembangan web menjadi populer pada beberapa tahun belakangan ini. AJAX ini sendiri bukan merupakan bahasa pemrograman baru, AJAX hanya merupakan sebuah teknik pemanfaatan object XMLHttpRequest dengan javascript untuk berkomunikasi dengan server secara Asynchronous, dengan pemanfaatan object XMLHttpRequest ini kita dapat membuat proses berjalan secara background atau bekerja dibelakang layar sementara user dapat tetap berinteraksi dengan halaman web yang ada. Pemanfaatan tehnik Asynchronous ini jugalah yang telah mendorong pengembangan web menjadi lebih kaya atau banyak yang menulisnya dengan istilah pengembangan Rich Internet Application (RIA) atau WEB 2.0

Pendahuluan

AJAX pertama kali di perkenalkan oleh Jesse James Garrett pada tulisannya yang berjudul AJAX: A New Approach To Web Applications. Jesse James memberi istilah AJAX untuk singkatan dari Asynchronous JavaScript And XML, namun pada perkembangannya data yang dikomunikasikan secara Asynchronous tidaklah harus berupa XML data, kita mengunakan format data lain untuk dikomunikasikan secara Asynchronous dengan server seperti PLAIN TEXT FILE, HTML DATA atau juga berupa SWF data (Flash file).

Pada artike ini kita akan membahas dan bekenalan dengan pemanfaatan tehnik Asynchronous dengan mengunakan object XMLHttpRequest yang dibuat dengan Javascript.

Sejarah singkat AJAX

Tehnik komunikasi Asynchronous dengan server sendiri pertama kali dikembangkan oleh microsoft pada tahun 1997, kemudian pertama kali XMLHttpRequest Object diperkenalkan pada IE5 (circa 1998) dan kemudian dipergunakan secara luas pada Outlook web access. Jesse James Garrett's kemudian memberikan istilah AJAX untuk tehnik Asynchronous ini pada salah satu tulisannya di tahun 2005. Dan hingga saat ini telah banyak yang mengimplementasikan tehnik ini dalam pengembangan web, sebagai contoh pengunaan AJAX dalam web application bisa dilihat pada beberapa link berikut:

- Google Map
- Flickr.com
- gmail.com
- google sugest
- yahoo.com

dan masih banyak lagi contoh lainnya yang dapat ditemukan pada aplikasi web saat ini.

TIPS

Sebelum memulai membaca lebih lanjut sebaiknya pastikan untuk mendownload file **contoh-source.zip** yang juga disertakan dalam ilmukomputer.com. File tersebut adalah file-file source contoh yang akan kita gunakan dalam arttikel ini.

Setelah selesai mendownload file tersebut buatlah sebuah folder LATIHAN didalam komputer anda dan extract isi file contoh-source.zip tersebut kedalamnya, nantinya kita akan mengunakan file-file dalam folder LATIHAN tersebut untuk mencoba contoh-contoh dalam artikel ini.

*Berikutnya setelah selesai menyiapkan folder LATIHAN sebaiknya mulailah membuat kopi panas sebagai teman selama membaca dan mencoba artikel ini

**langkah ini bersifat optional*

BERKENALAN DENGAN AJAX

Sebelum membahas panjang lebar tentang scripting dan coding, kita berkenalan dahulu dengan apa yang disebut dengan AJAX, siapa sebenarnya AJAX tersebut dan bagaimana

kelakuannya.

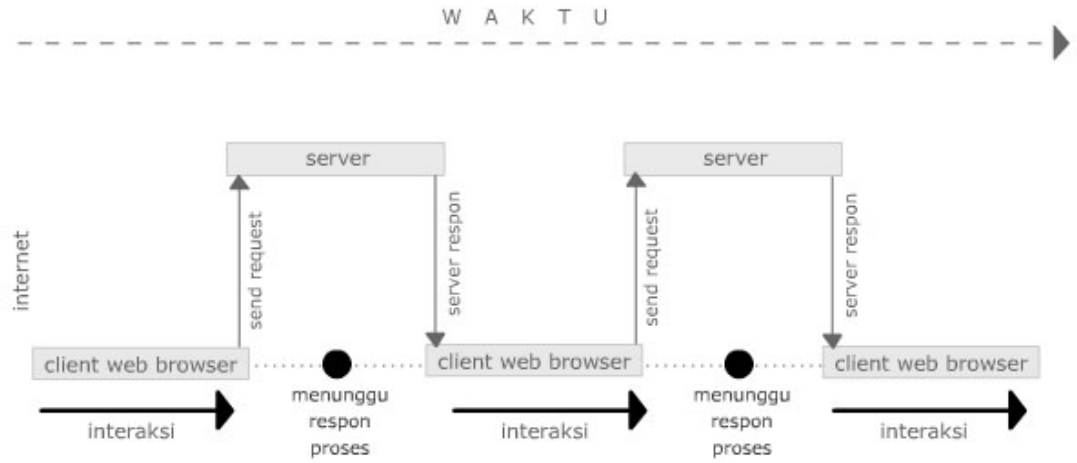
AJAX bukan merupakan bahasa pemrograman baru, namun hanya sebuah istilah untuk tehnik pemanfaatan Javascript dalam mengontrol class object XMLHttpRequest untuk berkomunikasi dengan server kemudian merefresh/mengupdate content yang ada dalam halaman web tanpa melakuan Reload keseluruhan halaman web seperti pada metode Tradisional sebelumnya, AJAX sendiri merupakan singkatan dari "Asynchronous JavaScript And XML".

Asynchronous & Synchronous

Bagaimana AJAX bekerja? dan mengapa pemanfaatan AJAX ini membawa istilah baru pada pengembangan web sehingga muncul istilah WEB 2.0? dan ada beberapa istilah lain yang juga dapat ditemukan dengan menanfatan AJAX seperti Rich internet aplication (RIA).

Dari singkatan AJAX sebelumnya ditemukan istilah “Asynchronous”, sekarang kita bahas sedikit dahulu tentang istilah tersebut dalam pengembangan web. Jika ada tehnik *Asynchronous* maka juga terdapat tehnik *Synchronous*, apa sih perbedaan kedua tehnik tersebut dalam pengembangan web? Berikut sedikit gambaranya.

Agar lebih mudah dipahami saya bahas dengan sedikit gambar berikut untuk proses SYNCHRONOUS dalam pengembangan web yang telah lama digunakan sebelumnya.



Gambar 1
(Synchronous proses)

Saat user berinteraksi dengan sebuah halaman web terdapat banyak pilihan link yang dapat ditemukan, dan saat link tersebut di click maka halaman web tersebut akan melakukan

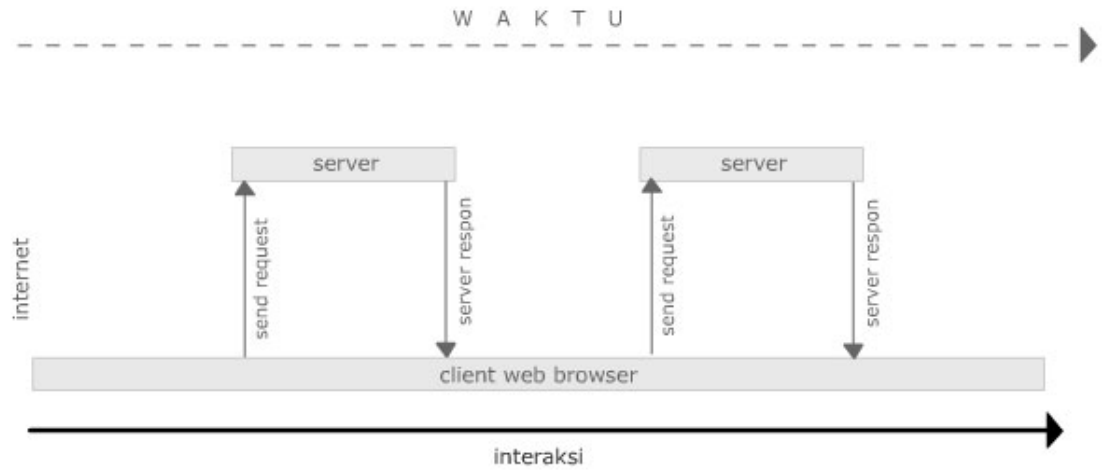
komunikasi dengan server melalui internet untuk meminta proses ke server (send Request)

Selama server melakukan proses, user akan menunggu hasil proses tersebut.

Setelah server selesai melakukan proses yang diminta maka server akan mengembalikan hasil proses yang akan ditampilkan pada client webbrowser.

Dalam proses synchronous ini user melakukan permintaan proses (request process) ke server dan menunggu hingga server mengembalikan hasil proses yang di minta, umumnya hasil proses dari server merupakan sebuah halaman web baru yang ditampilkan pada webbrowser user.

Pemanfaatan tehnik Asynchronous dengan Javascript ini dalam pengembangan web ini lebih dikenal dengan istilah WEB 2.0 (baca: web two point O). Dengan tehnik proses Asynchronous kita dapat membuat sebuah aplikasi web lebih kaya atau banyak yang menulis dengan istilah *Rich internet aplication (RIA)*, kita dapat membiarkan user untuk tetap berinteraksi dengan halaman web yang ada selama proses request dilakukan dan selama server belum mengembalikan hasil proses yang diminta. Dan saat hasil proses telah selesai kita hanya perlu mengupdate data halaman web yang telah ada, berikut gambaran dari proses kerja Asynchronous tersebut:



Gambar 2
(Asynchronous proses)

User berinteraksi dengan link-link yang ada untuk meminta proses ke server, proses yang diminta akan dikirimkan secara background ke server, selama server belum mengembalikan data hasil proses, user dapat tetap bekerja dengan halaman web yang ada sebelumnya.

Setelah server selesai melakukan proses, hasil proses tersebut akan dikirmkan kembali kepada

web browser, saat data yang dikirimkan server telah diterima oleh webbrowser user maka data tersebut ditampilkan pada halaman web yang telah ada sebelumnya.

Disini terlihat semua proses komunikasi dengan server dilakukan secara background atau mungkin bisa dikatakan semua proses dilakukan Behind the Scene :)

XMLHttpRequest Object

Untuk dapat mengembangkan aplikasi web dengan tehnik AJAX ini kita perlu meng-kombinasikan beberapa hal berikut:

- Javascript untuk membuat object XMLHttpRequest yang kita gunakan untuk berkomunikasi dengan server secara behind the scene.
- DOM (Document Object Model), hasil proses yang diterima akan kita tampilkan dengan memanipulasi object DOM yang telah kita persiapkan sebelumnya untuk menampilkan data hasil proses yang diberikan server.
- XML (eXtensible Markup Language) format data yang dikembalikan oleh server, data XML ini siap dibaca dan ditampilkan untuk mengupdate content pada halaman web.

secara sederhana kita akan memanfaatkan beberapa kombinasi diatas untuk membuat aplikasi web dengan tehnik AJAX ini, namun kita masih dapat mengembangkannya lebih lanjut setelah mengetahui proses kerja dari tehnik AJAX ini.

Kelebihan utama AJAX sendiri terletak pada pemanfaatan class object XMLHttpRequest untuk berkomunikasi dengan Web Server secara background dalam melakukan request.

Saat ini terdapat banyak aplikasi web browser, dan yang agak sedikit menyebalkan adalah karena masing-masing browser mempunyai standart dan cara berbeda untuk membuat object XMLHttpRequest ini.

Untuk dapat berjalan dengan baik pada semua browser kita perlu menyiapkan beberapa kondisi untuk mendeteksi web browser yang digunakan oleh user, dan kemudian membuat object XMLHttpRequest tersebut sesuai web browser yang digunakan.

Saat ini setidaknya ada 5 web browser yang banyak digunakan (Firefox, Mozilla, IE7, IE sebelum versi 7, Opera dan juga Safari), berikut pembuatan object XMLHttpRequest dengan javascript pada masing-masing web browse tersebut:

- Untuk membuat class object pada browser Firefox, Safari, Mozila, Opera.

```
oAJAX = new XMLHttpRequest();
```

- Untuk membuat class object pada browser IE7

```
oAJAX = new ActiveXObject('MSXML2.XMLHTTP');
```

- Untuk membuat class object pada browser IE versi lama

```
oAJAX = new ActiveXObject('Microsoft.XMLHTTP');
```

Dari 3 kemungkinan diatas kita akan menyiapkan sebuat sebuah function untuk pembuatan object XMLHttpRequest dengan mendeteksi web browser yang digunakan oleh user, berikut function javascript lengkapnya:

```
function createRequest(){

    var oAJAX = false;

    try {
        oAJAX = new ActiveXObject("Msxml2.XMLHTTP");
    } catch (e) {
        try {
            oAJAX = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e2) {
            oAJAX = false;
        }
    }


    if (!oAJAX && typeof XMLHttpRequest != 'undefined') {
     oAJAX = new XMLHttpRequest();
    }

    if (!oAJAX){
        alert("Error saat membuat XMLHttpRequest!");
    }else{
        alert("XMLHttpRequest sukses dibuat!");
    }
    return oAJAX;
}
```

Function createRequest() diatas akan mengembalikan object XMLHttpRequest jika berhasil dibuat. Pada bagian awal function akan dibuat deklarasi sebuah variable oAJAX dengan nilai awal FALSE.

```
var oAJAX = false;
```

Setelah membuat deklarasi object oAJAX, script berikutnya mencoba membuat object XMLHttpRequest untuk IE7.

```
try {
    oAJAX = new ActiveXObject("Msxml2.XMLHTTP");
}
```

Jika terjadi kegagalan atau error dalam pembuatan object, maka persiapan antisipasi error akan dijalankan dan mencoba membentuk object dengan script IE versi sebelumnya.

```
catch (e) {
    try {
        oAJAX = new ActiveXObject("Microsoft.XMLHTTP");
    } catch (e2) {
        oAJAX = false;
    }
}
```

Jika pembuatan object masih gagal maka setelah oAJAX akan tetap dalam kondisi FALSE, dan dapat dipastikan selesai block TRY ... CACTH diatas web browser yang digunakan bukan IE. Maka selanjutnya kita mencoba membuat object XMLHttpRequest untuk Firefox, Mozilla, Opera atau Safari.

```
if (!oAJAX && typeof XMLHttpRequest != 'undefined') {
  oAJAX = new XMLHttpRequest();
}
```

Bagian script diatas akan memeriksa terlebih dahulu status object oAJAX, jika masih dalam kondisi FALSE atau undefined maka proses pembuatan akan coba dilakukan dengan pembuatan object untuk Firefox, Mozilla, Opera dan Safari.

Bagian script berikutnya hanya akan menampilkan message Alert apakah object XMLHttpRequest telah berhasil terbentuk atau tidak.

```
if (!oAJAX){
    alert("Error saat membuat XMLHttpRequest!");
}else{
    alert("XMLHttpRequest sukses dibuat!");
}
```

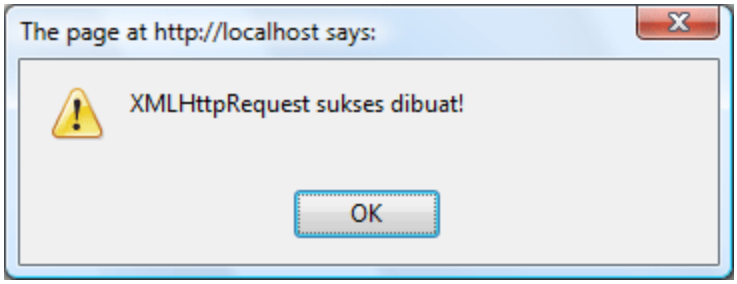
Jika hingga akhir function object XMLHttpRequest belum terbentuk, kemungkinan besar web browser yang digunakan bukan kelima yang kita sebutkan sebelumnya.

Sekarang mari kita coba function diatas dalam sebuah file HTML untuk mencoba membuat object XMLHttpRequest. Pertama buatlah sebuah struktur file HTML dengan editor yang biasa anda gunakan dan copy paste semua script dibawah ini untuk mengantikan script yang ada didalamnya.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />

<title>AJAX - Demo 1</title>
<script language="javascript">
    function createRequest(){
      var oAJAX = false;
      try {
        oAJAX = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
        try {
          oAJAX = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e2) {
          oAJAX = false;
        }
      }

      if (!oAJAX && typeof XMLHttpRequest != 'undefined') {
            oAJAX = new XMLHttpRequest();
      }
      if (!oAJAX){
            alert("Error saat membuat XMLHttpRequest!");
      }else{
            alert("XMLHttpRequest sukses dibuat!");
      }
      return oAJAX;
    }

</script>
</head>
<body onload="javascript:createRequest();">
Halaman ini memangil function pembuatan class object XMLHttpRequest
<br />Jika gagal atau pun berhasil akan di tampilkan Alert message.
</body>

</html>
```

Contoh script diatas juga disertakan dalam file source-code.zip yang dapat didownload dari ilmukomputer.com.

Setelah selesai membuat file tersebut cobalah buka file tersebut pada sebuah web browser maka akan ditampilkan ALERT message jika object XMLHttpRequest telah berhasil dibentuk seperti gambar1 dibawah ini.



Gambar 3
(Membuat object XMLHttpRequest)

Setelah berhasil membuat object XMLHttpRequest, selanjutnya kita coba memanfaatkan object tersebut untuk berkomunikasi secara Asynchronous dengan server.

Untuk berkomunikasi dengan server dan merespon hasil yang dikembalikan oleh server kita akan menfaatkan bebera Method, Event dan Property dari object XMLHttpRequest, berikut beberapa hal tersebut yang akan kita gunakan:

- Open (method)
- onreadystatechange (event)
- readyState (property)
- responseText (property)
- Send (method)

OPEN Method
Method OPEN ini akan kita gunakan untuk membuka komunikasi request kepada server untuk melakukan permintaan proses, syntax lengkapnya adalah sebagai berikut:

```
(XMLHttpRequest Object).open(<Type_request>,<url_file>,<async_status>);
```

Type_status = status proses yang dilakukan (GET/POST)
url_file = alamat url/file yang akan direquest ke server
async_status = status asynchronous (TRUE/FALSE)

ONREADYSTATECHANGE Event
Event berikut dapat kita manfaatkan untuk mengetahui status status hasil request kita kepada server, saat terjadi perubahan status request yang kita minta event ini akan dijalankan.

ONREADYSTATE Property

Property berikut akan berisi flag status request, untuk mengetahui perubahan status disini kita memanfaatkan event *Onreadystatechange* yang dijelaskan sebelumnya, setiap kali property ini berubah event *Onreadystatechange* akan dijalankan, sehingga kita hanya perlu memeriksa setiap perubahan status pada pada event *Onreadystatechage.* Berikut beberapa status yang ada dalam property ini:

- 0: uninitialized --> Open status belum dijalankan
- 1: loading --> Status request belum di jalankan
- 2: loaded --> Status request sudah di kirimkan, dan menunggu respon dari server.
- 3: interactive --> Respon dari server sedang dalam proses download.
- 4: completed --> Respon data dari server telah selesai di download.

RESPONSETEXT Property

Property berikut akan berisi data hasil pengembalian dari server jika telah diterima oleh browser.

SEND method

Method inilah yang kita gunakan untuk mengirimkan request data kepada web server, method OPEN sebelumnya hanya untuk membuka jalur komunikasi dengan server. Pada proses SEND inilah komunikasi request dikirimkan kepada server, berikut sintaks lengkapnya:

```
(XMLHttpRequest Object).send();
```

Berikut contoh function untuk melakukan komunikasi kepada server dengan memanfaatkan object XMLHttpRequest.

```
function requestContent() {
        oRequest = createRequest();
        var url = "dataLoad.html";

        // Buka komunikasi dengan server
        oRequest.open("GET", url, true);

        // menunggu respon dari server
        // jika sudah di dapat respon dari server, maka hasil respon di tampilkan
        oRequest.onreadystatechange = function () {
                document.getElementById("divContent").innerHTML=
                "<div align='center'>Menunggu Respon server</div>";

                if (oRequest.readyState == 4) {
                    // baca data respon dari server
                    var response = oRequest.responseText;
                    document.getElementById("divContent").innerHTML = response;
```

```
                }
        }
        oRequest.send(null);
}
```

Pada bagian awal function akan coba dibuat object XMLHttpRequest dengan memanfaatkan function **createRequest()** yang kita bahas sebelumnya, jika berhasil maka hasil object tersebut akan ditampung kedalam sebuah variable object yang diberi nama **oRequest**.

```
        oRequest = createRequest();
```

Bagian script berikutnya adalah deklarasi alamat URL yang akan direquest ke server, disini kita akan meminta server memproses file dataLoad.html.

```
        var url = "dataLoad.html";
```

Bagian script berikutnya adalah proses request yang kita lakukan kepada server dengan memanfaatkan method OPEN pada object XMLHttpRequest.

```
        oRequest.open("GET", url, true);
```

Selanjutnya kita menyiapkan sebuah function yang akan dijalankan setiap kali nilai variable *ReadyState* berubah, kita dapat memeriksa perubahan tersebut melalui event *onreadystatechange* karena event ini akan selalu dijalankan setiap kali terjadi perubahan pada variable *ReadyState*.
Seperti dijelaskan sebelumnya nilai pada variable ReadyState dapat berisi antara (0,1,2,3 atau 4), saat nilai variable *readyState=4* berarti semua data hasil respon yang dikirimkan oleh server telah diterima dengan selamat sampai di web browser, selanjutnya adalah tugas kita untuk memanfaatkan data tersebut. Pada contoh diatas kita memampilkan data hasil respon tersebut kedalam DIV AREA yang kita telah persiapkan.

```
        oRequest.onreadystatechange = function () {
                document.getElementById("divContent").innerHTML=
                "<div align='center'>Menunggu Respon server</div>";

                if (oRequest.readyState == 4) {
                    // baca data respon dari server
                    var response = oRequest.responseText;
                    document.getElementById("divContent").innerHTML =
                response;
                }
        }
```

Untuk menampilkan data hasil respon server ini kita memanfaatkan *innerHTML* dari DIV AREA yang telah kita beri ID=divContent sebelumnya.

INNER HTML INFO

berikut sedikit penjelasan tentang innerHTML yang digunakan diatas, innerHTML fnction untunk mengambil nilai/value yangg berada diantara TAG PEMBUKA & TAG PENUTUP, contoh:

innerHTML disini
innerHTML disini

Untuk dapat menuliskan data pada innerHTML suatu tag kita perlu memberi ID pengenal pada TAG tersebut, dan kemudian memangil ID tersebut dan menuliskan data didalamnya.

Selama status readyState belum bernilai 4 atau completed, maka status innerHTML pada DIV AREA yang kita persiapkan sebelumnya diisi dengan pesan “Menunggu Respon server”.

Contoh sederhana ini kita hanya berikan status menunggu tersebut berupa flat text, namun sebenarnya kita dapat mengantinya dengan images file, flash file atau sebagainya disini sesuai keperluan agar lebih fancy dan menarik.

Pada contoh disini kita juga hanya memeriksa status readyState==4, atau status dimana server telah memberikan respon dan respon tersebut telah selesai diterima oleh web browser, jika diperlukan kita dapat membuat pemeriksaan status readyState dan menampilkan pesan berbeda untuk masing-masing status readyState.

Untuk memcoba penjelasan panjang lebar diatas buatlah file dan beri nama sesuai dengan keinginan anda, dan kemudian isikan script didalamnya seperti contoh berikut ini:

(Pada contoh source code yang disertakan file ini diberi nama contoh2-responText.html)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>AJAX - Demo 1</title>
<script language="javascript">
function createRequest(){
	var oAJAX = false;

	try {
```

```
        oAJAX = new ActiveXObject("Msxml2.XMLHTTP");
      } catch (e) {
        try {
          oAJAX = new ActiveXObject("Microsoft.XMLHTTP");
        } catch (e2) {
          oAJAX = false;
        }
      }


      if (!oAJAX && typeof XMLHttpRequest != 'undefined') {
         oAJAX = new XMLHttpRequest();
      }

      if (!oAJAX){
        alert("Error saat membuat XMLHttpRequest!");
      }
      return oAJAX;
}

function requestContent(){
 oRequest = createRequest();
 var url = "dataLoad.html";

 // Buka komunikasi dengan server
 oRequest.open("GET", url, true);

 // menunggu respon dari server
 oRequest.onreadystatechange = function () {
    document.getElementById("divContent").innerHTML=
    "<div align='center'>Menunggu Respon server</div>";

    if (oRequest.readyState == 4) {
      // baca data respon dari server
                                                  var response = oRequest.responseText;
      document.getElementById("divContent").innerHTML = response;
    }
 }

 // Send the request
 oRequest.send(null);
}
</script>
```

```
<style type="text/css">
<!--
body,td,th {
	font-family: Verdana, Arial, Helvetica, sans-serif;
	font-size: 11px;
	color: #006699;
}
-->
</style>

</head>

<body>
<table width="100%" border="0" cellspacing="0" cellpadding="3">
<tr>
<td>
 Pemanfaatan class object XMLHttpRequest untuk
 berkomunikasi dengan server <br />
 <a href="#" onclick="javascript:requestContent();">
 Click disini     untuk load Data</a>
</td>
</tr>
<tr>
<td>
    <div id="divContent"></div>
</td>
</tr>
</table>
</body>
</html>
```

Dan buat juga sebuah file lain dan beri nama sesuai keinginan anda **,** file ini merupakan file data yang akan direquest dari server. Isikan script berikut pada file dataLoad.html tersebut:

(Pada contoh source code yang disertakan file ini diberi nama dataLoad.html)

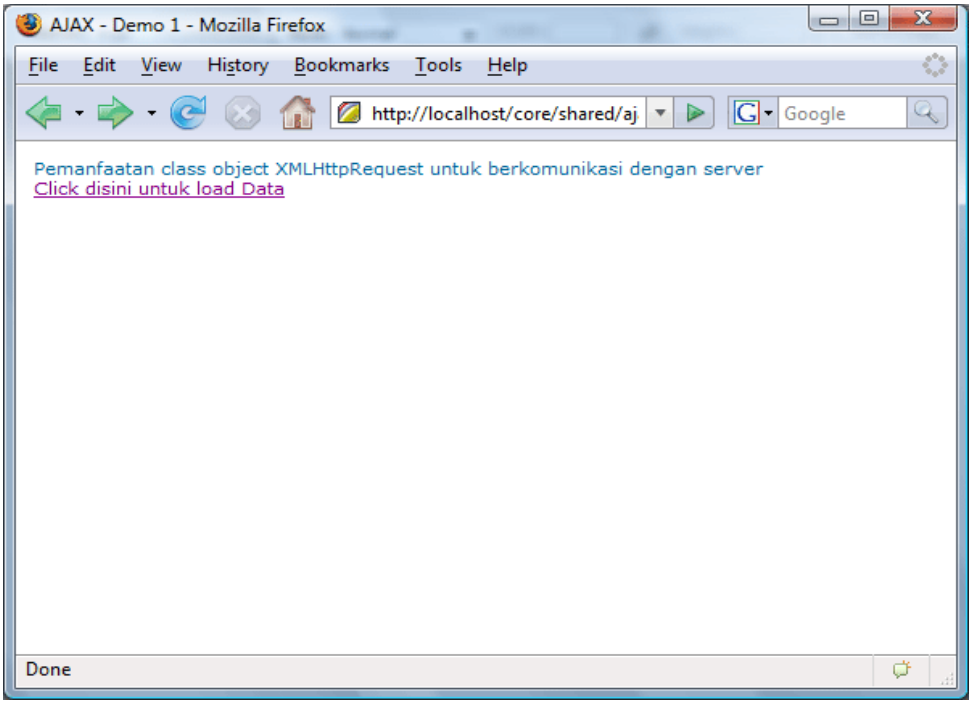
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
<style type="text/css">
<!--
body,td,th {
```

```
      font-family: Verdana, Arial, Helvetica, sans-serif;
      font-size: 11px;
      color: #006699;
}
.table {
      font-family: Verdana, Arial, Helvetica, sans-serif;
      font-size: 11px;
      font-style: normal;
      line-height: normal;
      font-weight: normal;
      font-variant: normal;
      text-transform: none;
      color: #006699;
      text-decoration: none;
      background-color: #FFFFEA;
      border: 1px solid #0099CC;
}
-->
</style>
</head>

<body>
<table width="500" border="0" cellpadding="3" cellspacing="0" class="table">
 <tr>
   <td><strong>File dataLoad.html</strong><br />
   Apapun isi content dari file ini akan di load dan di tampilkan pada lokasi DIV content yang di tuju. <br />
   <br />
   File yang di load ini bisa berupa file statis biasa, ataupun file berisi content yang di proses terlebih dahulu oleh server. </td>
 </tr>
</table>
</body>
</html>
```

Contoh script diatas juga disertakan dalam file source-code.zip yang dapat didownload dari ilmukomputer.com.

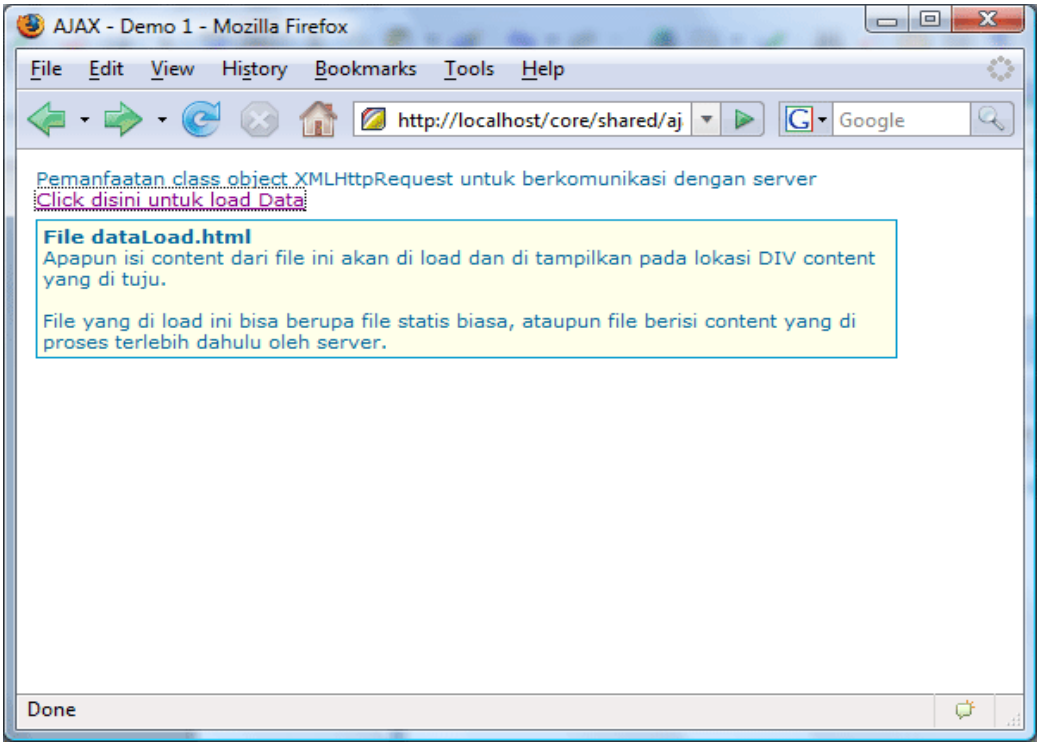
File dataLoad.html yang ditampilkan disini hanya nampilkan sebuah pesan sederhana saya sebagai contoh awal, pada contoh ini kita hanya sedikit mendemonstrasikan pengunaan object XMLHtmlRequest dalam melakukan proses Asynchronous dengan server.

Untuk mencoba contoh diatas letakan kedua file tersebut pada satu folder yang sama, kemudian buka file demo1.html pada web browser hasilnya akan seperti gambar2 dibawah ini.



Gambar 4
(Request data dengan object XMLHttpRequest)

Setelah data diatas tampil pada web browser cobalah click link yang ada didalamnya, dan setelah di click proses request akan dilakukan untuk memproses file dataLoad.html. Saat hasil proses selesai diterima oleh web browser maka secara automatis data tersebut akan ditampilkan pada DIV AREA yang kita persiapkan dibawahnya. Berikut gambar hasil proses setelah web browser menerima data dari server.



Gambar 5
(Hasil respon yang ditampilkan)

Semua contoh file yang ada dalam tulisan ini juga dapat didownload dari link yang disertakan pada ilmukomputer.com.

XML DATA FORMAT

Setelah berkenalan dengan AJAX dan mengetahui proses kerjanya dalam melakukan request Asychronous dan juga menerima respon dari server, sekarang kita akan membahas tentang XML data format yang dijadikan salah satu bagian dalam singkatan AJAX.

Sedikit penjelasan singkat XML

- XML merupakan singkatan dari eXtensible Markup Language.
- XML merupakan Tag base sintaks.
- Masing-masing tag dalam XML diawali dengan tag pembuka dan diakhiri dengan tag penutup, beberapa artikel atau tulisan juga biasa menyebut TAG sebagai ELEMENT. Namun dalam artikel ini saya akan mengunakan istilah TAG dalam menyebutkannya.
  contoh <data> ... </data>
- Single line tag dapat dibuat dengan memberikan closing tag pada bagian akhir "/>"
  contoh: <data atrr="value" />
- Attribut harus dituliskan pada bagian tag pembuka dan diapit oleh tanda " .. " atau '.. '
  contoh: <tagName attr="value">
- Comment dalam XML dbuat didalam tanda <!-- komentar disini -->, bagian comment

ini tidak akan diproses sebagai data dalam XML

- Data XML dapat sebagai Datasource kecil dalam melakukan pertukaran informasi.
- Lebih lengkap tentang XML bisa di lihat pada link berikut ini : http://www.w3.org

Data-data XML akan berisi TAG, ATTRIBUT, ID dan sebagainya, berikut contoh sebuah data dalam format XML struktur.

```
<xml id=”contactList”>
    <contact>
        <id>1</id>
        <nama>Jerry Peter</nama>
        <phone type='mobile>08137777777</phone>
        <phone type='rumah'>(021) 777777777</phone>
        <phone type='kantor'>(021) 8888888888</phone>
        <email>jerry.peter@gmail.com</email>
        <blog>www.ruangkecil.or.id</blog>
    </contact>
</xml>
```

Semua data yang ada didalamnya disimpan dalam sebuah TAG, dan beberapa TAG diatas memili ATTRIBUT. Pada contoh-contoh berikutnya kita akan mencoba membuat pengembalian data server mengunakan format XML, dan kemudian kita akan membuat sebuah proses data yang dikembalikan tersebut

Untuk membaca data dalam format XML tersebut kita akan mengunakan bantuan DOM (Document Object Model), dengan mengetahui tentang struktur dari sebuat Document kita akan lebih mudah membacanya.

DOCUMENT OBJECT MODEL (DOM)

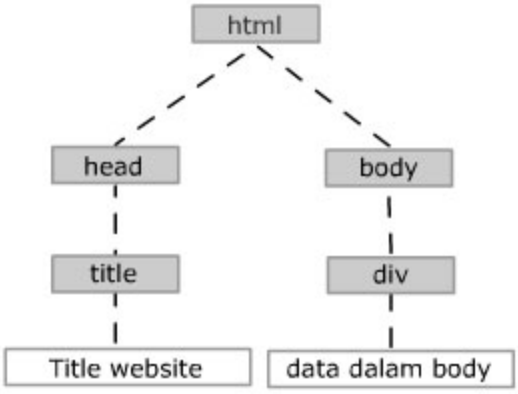
Untuk membaca data-data dalam format XML kita perlu sedikit berkenalan konsep Document Object Model atau biasa disingkat dengan istilah DOM, pada artikel ini saya tidak akan menjelaskan terlalu detail tentang DOM, kita hanya akan membahas sedikit saja tentang DOM ini dan juga bagaimana membaca dan mempergunakannya untuk keperluan kita, mungkin nanti akan saya coba buat artikel terpisah tentang pembahasan DOM yang lebih detail.

DOM disini adalah struktrur object dari sebuah document XML yang berisi TAG, ATTRIBUT, ID dan sebagainya. Dengan mengetahui struktur DOM nantinya kita dapat melakukan pembacaan data-data TAG dalam XML tersebut.

Jika terdapat document dengan tag HTML berikut:

```
<html>
    <head>
        <title>Title web</title>
    </head>
    <body>
        <div>data dalam body</div>
    </body>
</html>
```

Maka struktur Document Object Model (DOM) dari data tersebut adalah seperti diagram berikut ini:



Sebuah data dalam format XML akan memiliki TAG, ATTRIBUT atau juga ID, untuk membaca data dalam format tersebut kita dapat mengunakan fasilitas yang ada dalam DOM berikut:

- getElementsByTagName : digunakan untuk membaca tag berdasarkan NAMA
- getElementById : digunakan untuk membaca tag berdasarkan ID yang diberikan
- getAttribute : digunakan untuk membaca tag berdasarkan ATTRIBUTE
- firstChild : membaca Node array pertama dari data tag yang kita baca
- lastChild : membaca Node array terakhir dari data tag yang kita baca.
- data : membaca data text dari tag atau node yang kita inginkan
- innerHtml : membaca/menuliskan data kedalam bagian tag, innerHtml berbeda dengan property data sebelumnya karena innerHTML dapat berupa sebuah struktur data yang ada didalam TAG yang ambil, dialamnya mungkin akan dapat berisi tag-tag baru lagi, sedangkan property data hanya dapat digunakan jika data yang diambil berupa Text.

Contoh pengunaannya untuk membaca data XML sebagai berikut:

```
<xml id=”contactList”>
    <contact>
        <id>1</id>
        <nama>Jerry Peter</nama>
        <phone type='mobile>08137777777</phone>
        <phone type='rumah'>(021) 777777777</phone>
        <phone type='kantor'>(021) 8888888888</phone>
        <email>jerry.peter@gmail.com</email>
        <blog>www.ruangkecil.or.id</blog>
    </contact>
</xml>
```

Untuk membaca data tersebut pertama kita baca struktur XML tersebut kedalam sebuah variable dengan cara berikut:

```
var dataXML = document.getElementById(“contactList”);
```

Pada script diatas kita gunakan *getElementById* untuk membaca XML data yang telah kita beri ID=CONCACTLIST, setelah script diatas maka variabel dataXML kita tersebut akan berisi semua data dari XML dengan id concatList.

Berikutnya kita gunakan variable **dataXML** tersebut untuk membaca *tag contac*t yang ada didalamnya, cara membacanya kita gunakan *getElementsByTagName* seperti berikut:

```
var contact = dataXML.getElementsByTagName("contact")[0];
```

Pengunaannya sama dengan *getElementById* sebelumnya, namun pada contoh diatas kita tambahkan **[0]** pada bagian belakang script, hal ini adalah untuk mengacu kepada tag concact pertama yang ditemukan didalam array.
(Catatan: urutan array dalam DOM dimulai dari 0 sebagai index pertama data)

Setelah proses sintaks tersebut maka variabel **concact** kita akan berisi dengan semua data diantara *<concact> ... </concact>*.

Proses selanjutnya adalah proses membaca data yang ada yang kita tuju, proses pembacaannya masih mengunakan *getElementsByTagName*, namun sekarang kita gunakan *firstChild.data* untuk membaca data text yang ada didalam masing-masing tag.

Proses script untuk membacanya adalah sebagai berikut:

```
var id   = "ID: " + contact.getElementsByTagName("id")[0].firstChild.data;
var nama = "NAMA: " + contact.getElementsByTagName("nama")[0].firstChild.data;


/* ---- Membaca data dalam tag-tag Phone ---- */
var phone1 = "PHONE1: " + contact.getElementsByTagName("phone")[0].firstChild.data;

var phone2 = "PHONE2: " +  contact.getElementsByTagName("phone")[1].firstChild.data;
var phone3 = "PHONE3: " + contact.getElementsByTagName("phone")[2].firstChild.data;


var email  = "EMAIL: " + contact.getElementsByTagName("email")[0].firstChild.data;

var blog = "BLOG: " + contact.getElementsByTagName("blog")[0].firstChild.data;
```

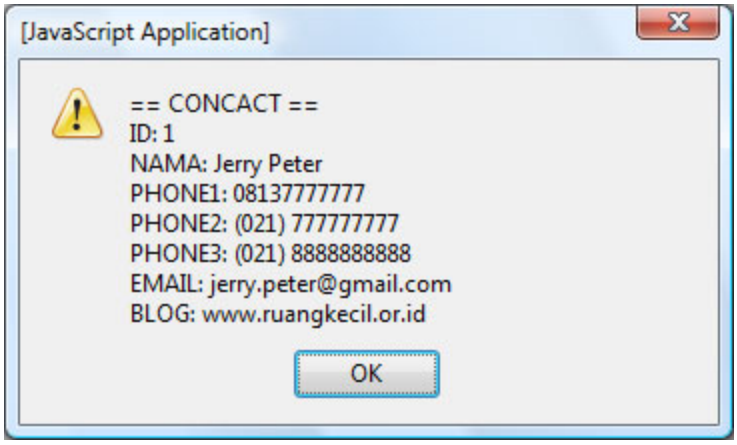
firshChild.data : digunakan untuk mengambil data text yang ada didalam masing-masing tag

Pada contoh diatas semua proses pengambilan data hampir sama, yang berbeda sedikit adalah pada proses pengambilan data dari *tag <phone>*, dimana digunakan index data yang berbeda mulai dari 0 sampai 2. Hal ini karena *tag <phone>* terdapat 3 buah dari dalam data yang kita baca, sehingga data tersebut akan berisi array yang memiliki index 0 hingga 2.
Sebagai tambahan pada awal masing-masing data yang di baca saya coba tambahkan String yang berisi keterangan masing-masing data.

Sampai disini semua data TEXT dari dalam TAG yang kita tuju telah kita simpan kedalam variabel-variabel, namun data tersebut belum terlihat tampil pada layar document karena semuanya masih disimpan didalam memori komputer, jika ingin menampilkan data tersebut kita bisa mencoba mengunakan ALERT javascript untuk menampilkannya sebagai berikut:

```
alert("== CONCACT ==\n" + id + "\n" + nama + "\n" + phone1 +
      "\n" + phone2  + "\n" + phone3  + "\n" + email + "\n" + blog);
```

Character *n* yang digunakan diatas adalah untuk menambahkan baris baru untuk masing-masing data yang akan ditampilkan, dengan alert tersebut data akan ditampilkan dalam sebuah Popup windows seperti gambar berikut:



Gambar 6
(Hasil pembacaan data XML)

Setelah berkenalan singkat dengan XML dan DOM berikutnya kita akan coba kombinasikan hal tersebut dalam proses komunikasi dengan server dalam proses request Asynchronous.

MENGUNAKAN XML FORMAT

Pengunaan XML dalam proses request dan komunikasi dengan server sebenarnya tidak berbeda dengan sebelumnya, yang menjadi perbedaan hanya pada saat membaca respon data XML yang dikirimkan oleh server, jika sebelumnya kita mengunakan property *responText* dalam menerima respon data dari server, sekarang kita akan mengunakan property *responXML* dalam menerima data XML respon dari server.

Dengan mengunakan responXML ini maka data yang diterima dapat kita baca sebagai data format XML dan kita dapat memperlakukan data tersebut sebagai data XML dan membaca masing-masing tag dengan mengunakan *getElementsByTagName* atau *getElementsById*.

Agar lebih mudah dimengerti saya coba bagi contoh pembacaan tersebut menjadi 3 contoh program, berikut :

1. MEMBACA FORMAT DATA XML
   Pada contoh pertama kita hanya akan mencoba membaca data dari file XML yang ada tanpa melakukan formating data

2. MEMBACA DATA XML + MANIPULASI DOM
   Pada contoh kedua kita akan coba membaca data XML kemudian melakukan formating data tersebut sesuai keperluan

3. MEMBACA DATA XML + MANIPULASI DOM + FORMATING DENGAN CSS
   Pada contoh terakhir kita akan mencoba membaca data XML, menampilkan data dan juga sedikit contoh formating data dengan CSS

CATATAN:

Untuk contoh2 program berikutnya saya telah memisahkan block function pembuatan Object XMLHttpRequest kedalam file terpisah yang saya beri nama createObject.js, dan file tersebut akan saya include kedalam masing-masing contoh yang akan dibuat dengan mengunakan script berikut:

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

Hal ini dilakukan agar function tersebut tidak dituliskan berulang-ulang pada setiap program yang kita buat.

CONTOH1: MEMBACA FORMAT DATA XML

Sekarang kita akan mencoba contoh sederhana membuat request kepada server dan membaca data respon XML yang dikembalikan oleh server, kemudian menampilkannya pada halaman web yang telah ada.
Pada 3 contoh pembacaan format XML ini kita akan mencoba membaca data XML yang berisi data photoBook, dan setelah kita membaca format data tersebut kita mencoba membuat menjadi sebuah photo thumbnail dan melakukan formating dengan CSS.
Data XML yang akan di request dari server adalah seperti dibawah ini:

```
<xml id="photoBook">
	<photoList>
		<photo>
			<id>1</id>
			<nama>MOBIL</nama>
			<file>photo/car.jpg</file>
			<keterangan>keterangan MOBIL disini</keterangan>

		</photo>
		<photo>
			<id>2</id>
			<nama>MENARA EIFFEL</nama>
			<file>photo/eiffel.jpg</file>
			<keterangan>keterangan MENARA EIFFEL</keterangan>
		</photo>
		<photo>
			<id>3</id>
			<nama>BUKIT</nama>
			<file>photo/hill.jpg</file>
			<keterangan>keterangan BUKIT disini</keterangan>
		</photo>
		<photo>
```

```
                    <id>4</id>
                    <nama>KACAMATA</nama>
                    <file>photo/kacamata.jpg</file>
                    <keterangan>keterangan KACAMATA disini</keterangan>
                </photo>
                <photo>
                    <id>5</id>
                    <nama>LONCENG</nama>
                    <file>photo/lonceng.jpg</file>
                    <keterangan>keterangan LONCENG disini</keterangan>

                </photo>
        </photoList>
</xml>
```

Pada format data XML diatas terdapat 5 data photo yang masing-masing memiliki ID, NAMA, FILE PHOTO dan juga keterangan masing-masing photo.

Secara singkat proses yang akan dilakukan adalah sebagai berikut:

1. Membuat object XMLHttpRequest dengan menanfaatkan function **createObject**
2. Membuka komunikasi dengan server mengunakan **OPEN** method
3. Mengirimkan request dengan **SEND** method
4. Menunggu respon dengan menafaatkan **ONREADYSTATECHANGE** event
5. Menerima data yg dikirimkan dengan mengunakan r**esponseXML**
6. Mengubah data *tag <photo>* pada file XML yang diterima menjadi array mengunakan **GETELEMENTBYTAGNAME**
7. Melakukan looping sebanyak array *tag <photo>* yang didapat
8. Membaca detail informasi pd masing2 tag <photo> dengan **GETELEMENTBYTAGNAME**
9. Menampilkan data masing-masing photo dengan mengunakan **INNERHTML**

Dari langkah2 diatas proses *createObject*, *OPEN*, *SEND* dan *INNERHTML* telah dijelaskan sebelumnya dibagian awal artikel, proses baru akan kita lakukan disini adalah menerima data dengan *responseXML*, dan juga membaca data XML dengan *getElementByTagName* dan juga *firstChild.data*.

RESONSEXML

Saat kita menerima data dalam format XML kita harus mengunakan responseXML agar javascript mengetahui data tersebut adalah XML format, dan kita dapat membaca data tersebut dengan mengunakan getElementByTagName atau getElementByID.
Sintaks pengunaanya tidak berbeda dengan responseText yang kita gunakan sebelumnya, secara lengkap penulisannya adalah sebagai berikut:

```
Variable = (XMLHttpRequest Object).responseXML
```

Setelah mengunakan sintaks diatas Variable yang kita gunakan akan berisi dengan format data XML yang diterima dari server, dan kemudian kita bisa memprosesnya lebih lanjut.

GETELEMENTBYTAGNAME
Function ini kita gunakan untuk membaca TAG-TAG XML yang telah kita dapatkan sebelumnya dengan mengunakan responseXML, dengan *getElementByTagName* kita dapat membaca tag-tag yang kita perlukan dan mengubahnya menjadi sebuah array. Setelah array terbentuk maka kita dapat melakukan pembacaan masing-masing array tersebut untuk dapat ditampilkan. Sintaks dan pengunaanya ada sebagai berikut:

```
Variable = xmlData.getElementByTagName(“nama tag”);
```

Setelah proses sintaks diatas Variable yang kita gunakan akan berisi dengan Array yang telah dikembalikan oleh getElementByTagName, langkah selanjutnya adalah membaca semua array tersebut secara berulang untuk ditampilkan datanya.

Tambahan lanjutan berikutnya adalah kita akan mengunakan *firstChild.data* untuk mengambil data dari masing-masing tag yang ada. Pengunaan *getElementByTagName* dan *firstChild.data* ini adalah bagian dari proses memanupulasi Document Object Model (DOM) data XML yang kita terima.

Function lengkap untuk melakukan proses request kepada server dan pembacaan data XML yang dikembalikan oleh server adalah sebagai berikut:

```
function requestContent(){
  oRequest = createObject();
  var url = "contohData.xml";

  // Buka komunikasi dengan server
  oRequest.open("GET", url, true);

  // menunggu respon dari server
  oRequest.onreadystatechange = function () {
              document.getElementById("divContent").innerHTML=
                    "<div align='center'>Menunggu Respon server</div>";

              if (oRequest.readyState == 4) {
                  document.getElementById("divContent").innerHTML = "";
                  //baca data respon dari server

                 var xmlData = oRequest.responseXML;
                 var photoList = xmlData.getElementsByTagName("photo");

                for (var i=0; i < photoList.length; i++){
```

```
                    var currentPhoto = photoList[i];
                    var id_ = currentPhoto.getElementsByTagName("id")[0].firstChild.data;
                    var nama_=
                         currentPhoto.getElementsByTagName("nama")[0].firstChild.data;
                    var file_ =
                        currentPhoto.getElementsByTagName("file")[0].firstChild.data;
                    var keterangan_ =
                        currentPhoto.getElementsByTagName("keterangan")[0].firstChild.data;
                    var data_=
                        "<div>" + id_ + ":" + nama_ + ":" + file_ + " : " + keterangan_ + "</div>";

                    document.getElementById("divContent").innerHTML =
                         document.getElementById("divContent").innerHTML + data_;

                    }
                }
            }
  // Send the request
  oRequest.send(null);
}
```

Untuk mencoba contoh berikut ini buatlah 2 file, yang pertama berisi data XML photoBook kita dengan diberi nama **contohData.xml** dan yang yang kedua buatlah sebuah file html biasa dengan nama yang **contoh1.html**.

Untuk file **contohData.xml** isikan datanya dengan XML format diatas, kemudian untuk file contoh1.html tuliskan script berikut didalamnya :

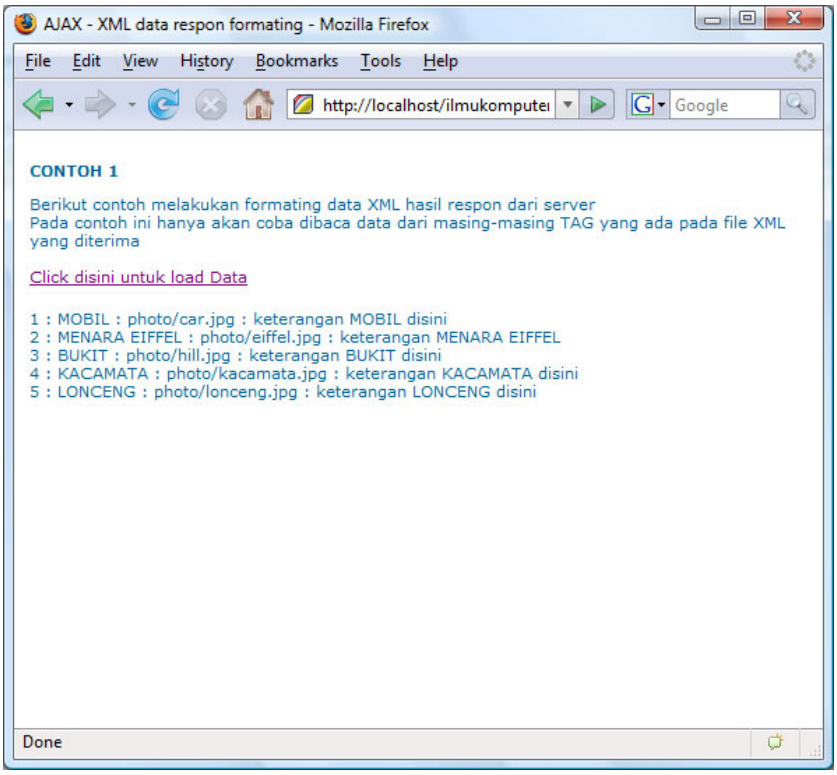
(pada file contoh source yang disertakan dalam ilmukomputer.com, script berikut dapat ditemukan pada file contoh3-responXML.html)

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>AJAX - XML data respon formating</title>
<script type="text/javascript" src="createObject.js"></script>
<script language="javascript">
function requestContent(){
  oRequest = createObject();
  var url = "contohData.xml";

  // Buka komunikasi dengan server
```

```
 oRequest.open("GET", url, true);

 // menunggu respon dari server
 // hasil respon akan di tampilkan pada DIV dengan ID='divContent'
 oRequest.onreadystatechange = function () {

            document.getElementById("divContent").innerHTML=
                    "<div align='center'>Menunggu Respon server</div>";

            if (oRequest.readyState == 4) {
                document.getElementById("divContent").innerHTML = "";
                //baca data respon dari server
                //alert(oRequest.responseXML);

                var xmlData = oRequest.responseXML;
                var photoList = xmlData.getElementsByTagName("photo");
                //alert("count: " +  photoList.length);

                for (var i=0; i < photoList.length; i++){
                        var currentPhoto = photoList[i];
                        var id_ = currentPhoto.getElementsByTagName("id")[0].firstChild.data;
                        var nama_ =
                              currentPhoto.getElementsByTagName("nama")[0].firstChild.data;
                        var file_ =
                             currentPhoto.getElementsByTagName("file")[0].firstChild.data;
                        var keterangan_ =
                              currentPhoto.getElementsByTagName("keterangan")[0].firstChild.data;
                        var data_ =
                             "<div>" + id_ + ":" + nama_ + ":" + file_ + ":" + keterangan_ + "</div>";

                        document.getElementById("divContent").innerHTML =
                              document.getElementById("divContent").innerHTML + data_;

                        }

                }
            }

 // Send the request
 oRequest.send(null);
}

</script>
<style type="text/css">
```

```
<!--
body{
        font-family: Verdana, Arial, Helvetica, sans-serif;
        font-size: 11px;
        color: #006699;
}
-->
</style>
<link href="ajax-example.css" rel="stylesheet" type="text/css" />
</head>

<body>
<table width="100%" border="0" cellspacing="0" cellpadding="3">
  <tr>
    <td>
     Berikut contoh melakukan formating data XML hasil respon dari server <br />
    Pada contoh ini hanya akan coba dibaca data dari masing-masing TAG yang ada pada file
XML yang diterima<br />
      <br />
    <a href="#" onclick="javascript:requestContent();">Click disini untuk load Data</a> </td>
  </tr>
  <tr>
    <td><div id="divContent"></div> </td>
  </tr>
</table>
</body>
</html>
```

Setelah membuat kedua file tersebut letakan keduanya pada sebuah FOLDER LATIHAN yang telah dibuat sebelumnya. Kemudian cobalah buka file contoh **contoh1.html** pada web browser dan click link terdapat didalamnya, saat link yang ada di click akan muncul sebuah data baru yang telah berhasil direquest dari server seperti pada gambar 5 dibawah ini.



Gambar 7
(Hasil contoh pertama)

Pada contoh pertama ini kita hanya mencoba membaca data XML yang ada kemudian menampilkanya kedalam document yang telah ada, semua data diperlakukan sebagai text. Jika dilihat pada tag file seharusnya kita dapat menampilkan data images photo dari link file yang ada, pada contoh kedua kita akan mencoba melakukan sedikit formating pada data tersebut dan menampilkan data images yang ada tersebut:

CONTOH2: MEMBACA DATA XML + MANIPULASI DOM
Berikutnya kita akan mencoba membaca data dan juga memanipulasi DOM dengan menambahkan struktur baru didalamnya saat data XML telah diterima.

Untuk menambahkan struktur data baru pada DOM yang telah ada kita akan mengunakan 3 function berikut :

- createElement : menambahkan sebuah element/tag baru kedalam DOM yang telah ada
  sintaks penulisannya sbb:
  ```
  document.createElement("namaTagBaru");
  ```

- appendChild : menambahkan data kedalam struktur element/tag
  sintaks penulisannya sbb:

```
<object DOM aktif>.appendChild("data yang akan ditambahkan");
```

- CreateTextNode : membuat text node baru kedalam tag/element DOM
  sintaks penulisannya sbb:

```
document.createTextNode("data text yang akan ditambahkan");
```

Pada contoh kedua ini kita akan melakukan skenario proses seperti berikut:

1. Membuat object XMLHttpRequest dengan menanfaatkan function **createObject**
2. Membuka komunikasi dengan server mengunakan **OPEN** method
3. Mengirimkan request dengan **SEND** method
4. Menunggu respon dengan menafaatkan **ONREADYSTATECHANGE** event
5. Menerima data yg dikirimkan dengan mengunakan r**esponseXML**
6. Mengubah data *tag <photo>* pada file XML yang diterima menjadi array mengunakan **GETELEMENTBYTAGNAME**
7. Melakukan looping sebanyak array *tag <photo>* yang didapat
8. Membaca detail informasi pd masing2 tag <photo> dengan **GETELEMENTBYTAGNAME**
9. Menambahkan beberapa tag/element baru kedalam DOM untuk setiap loop photoBook yang dilakukan

Dari skenario proses yang akan kita lakukan tidak berbeda dengan proses sebelumnya pada contoh 1, langkah 1 sampai 8 semuanya sama. Perbedaan proses adalah pada langkah ke 9, dimana setiap kali kita melakukan loop proses dari array photoBook yang ada akan ditambahkan sebuah struktur DOM baru kedalam document yang telah ada.

Perubahan yang kita lakukan akan dibuat pada function requestContent() yang ada, berikut detail lengkap proses requestContect() yang telah dimodifikasi untuk pembuatan object DOM baru didalamnya.

```
function requestContent(){
  oRequest = createObject();
  var url = "contohData.xml";

  // Buka komunikasi dengan server
  oRequest.open("GET", url, true);

  // menunggu respon dari server
  // hasil respon akan di tampilkan pada DIV dengan ID='divContent'
  oRequest.onreadystatechange = function () {


         document.getElementById("divContent").innerHTML=
            "<div align='center'>Menunggu Respon server</div>";

        if (oRequest.readyState == 4) {
```

```
            document.getElementById("divContent").innerHTML = "";
            //baca data respon dari server
            //alert(oRequest.responseXML);

            var xmlData = oRequest.responseXML;

            var photoList = xmlData.getElementsByTagName("photo");
            //alert("count: " +  photoList.length);

            for (var i=0; i < photoList.length; i++){
               var currentPhoto = photoList[i];

               var newThumb = document.createElement("div");
               var newNama = document.createElement("div");
               var nama_ =
document.createTextNode(currentPhoto.getElementsByTagName("nama")[0].firstChild.data);
               newNama.appendChild(nama_);

               var newFile = document.createElement("div");
               var file_ =
               "<img src='"+currentPhoto.getElementsByTagName("file")[0].firstChild.data+"' />";

              newFile.innerHTML= file_;

              var newKeterangan = document.createElement("div");
              var keterangan_
document.createTextNode(currentPhoto.getElementsByTagName("keterangan")[0].firstChild.
data);

              newKeterangan.appendChild(keterangan_);
              newThumb.appendChild(newNama);
              newThumb.appendChild(newFile);
              newThumb.appendChild(newKeterangan);

               document.getElementById("divContent").appendChild(newThumb);

                }
              }
            }

 // Send the request
 oRequest.send(null);
}
```

Perbedan proses pembacaan data XML ini dengan contoh sebelumnya terdapat pada bagian

dalam loop proses, dimana ditambahkan pembuatan object tag baru. Masing-masing proses pembuatan tag baru tersebut adalah sebagai berikut:

Pertama dibuat sebuah tag DIV baru dalam sintaks berikut:

```
var newThumb = document.createElement("div");
```

Kemudian ditambahkan sebuah tag DIV baru untuk nama photoBook.

```
var newNama = document.createElement("div");
```

Berikutnya dilakukan pembuatan textNode baru yang berisi data dari tag <nama> yang berada pada data XML yang dibaca.

```
var nama_ =
document.createTextNode(currentPhoto.getElementsByTagName("nama")[0].firstChild.data);
```

Kemudian textNode yang telah dibuat ditambahkan kedalam tag DIV nama yang baru dibuat dengan mengunakan appendChild.

```
newNama.appendChild(nama_);
```

Proses berikutnya adalah pembacaan data tag file photo, disini kita langsung menambahkan data photo tersebut sebagi source image untuk tag <img>.

```
var newFile = document.createElement("div");
var file_ = "<img src='"+currentPhoto.getElementsByTagName("file")[0].firstChild.data+"' />";
newFile.innerHTML= file_;
```

Untuk pembacaan data tag keterangan juga dilakukan hal yang sama dengan proses penambahan data nama sebelumnya.

```
var newKeterangan = document.createElement("div");
var keterangan_ =
    document.createTextNode(currentPhoto.getElementsByTagName("keterangan")
    [0].firstChild.data);
newKeterangan.appendChild(keterangan_);
```

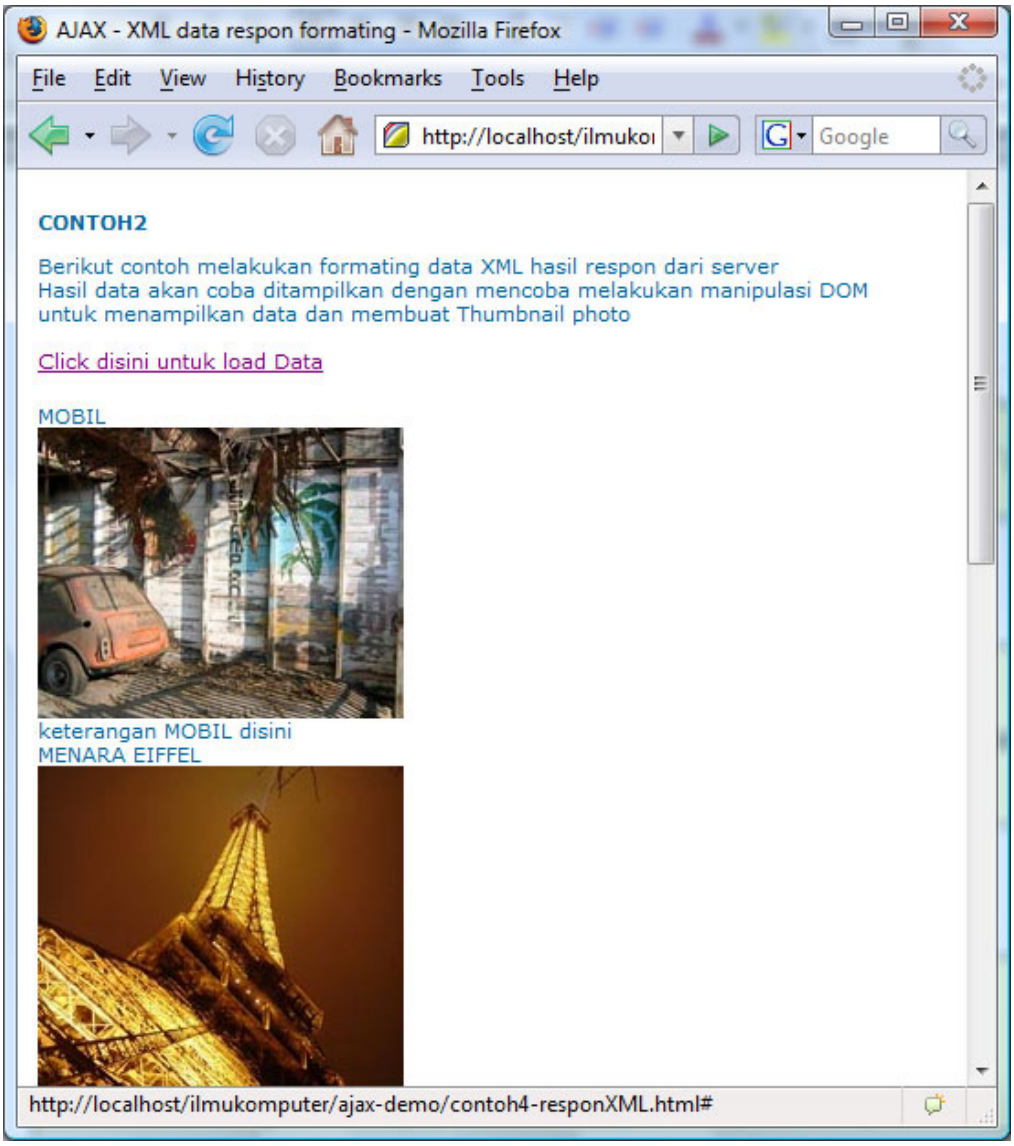
Setelah semua data tersebut selesai dibuat maka selanjutnya menambahkan data-data tag/element baru tersebut kedalam divContent yang telah dipersiapkan sebelumnya.

```
newThumb.appendChild(newNama);
```

```
            newThumb.appendChild(newFile);

            newThumb.appendChild(newKeterangan);

            document.getElementById("divContent").appendChild(newThumb);
```

Source lengkap untuk contoh ini dapat ditemukan pada file contoh4-responXML.html.

Berikut hasil proses untuk contoh kedua ini saat dijalankan pada web browser.



Gambar 7
(Hasil contoh kedua)

CONTOH3: MEMBACA DATA XML + MANIPULASI DOM + FORMATING DENGAN CSS

Contoh ketiga ini kita akan mencoba menambahkan aksesoris dengan CSS kedalam hasil data XML yang telah dibaca kedalam struktur DOM baru. Untuk dapat menambahkan aksesoris CSS pada setiap tag DIV yang dibuat saat loop proses pembacaan data photoBook, kita perlu nambahkan class pada masing DIV, kemudian membuat formating CSS untuk masing-masing class tersebut.

Untuk menambahkan ATTRIBUTE CLASS pada masing-masing DIV TAG yang kita buat saat loop pembacaan data photoBook akan digunakan function *setAttribute* yang ada pada DOM.

Sintaks penulisan function setAttribute adalah sebagai berikut:

```
<nama tags/element>.setAttribute("<nama_Attribute>", "<nilai_attribute>");
```

Pada setiap DIV yang dibuat akan kita tambahkan 2 attribute baru yaitu CLASS dan CLASSNAME, mengapa harus 2 yang kita tambahkan karena terdapat sedikit perbedaan penambahan attribute class ini pada web browser mozilla, firefox dan IE (Internet Explorer).

Proses skenario yang akan dilakukan masih sama dengan proses pada contoh 2, hanya saat pembuatan tag akan ditambahkan proses pembuataan attribute untuk masing-masing tag tersbut, berikut function createRequest() lengkap setelah ditambahkan proses penambahan attribute pada masing-masing tag.

```
function requestContent(){
  oRequest = createObject();
  var url = "contohData.xml";

  // Buka komunikasi dengan server
  oRequest.open("GET", url, true);

  // menunggu respon dari server
  // hasil respon akan di tampilkan pada DIV dengan ID='divContent'
  oRequest.onreadystatechange = function () {


              document.getElementById("divContent").innerHTML=
                    "<div align='center'>Menunggu Respon server</div>";

              if (oRequest.readyState == 4) {
                    document.getElementById("divContent").innerHTML = "";
                    //baca data respon dari server
                    //alert(oRequest.responseXML);
```

```
var xmlData = oRequest.responseXML;

var photoList = xmlData.getElementsByTagName("photo");
 //alert("count: " +  photoList.length);

for (var i=0; i < photoList.length; i++){
    var currentPhoto = photoList[i];

    var newThumb = document.createElement("div");
            newThumb.setAttribute("className", "thumbnail");
    newThumb.setAttribute("class", "thumbnail");
            var newNama = document.createElement("div");
    newNama.setAttribute("className", "nama");
    newNama.setAttribute("class", "nama");

    var nama_ =
     document.createTextNode(currentPhoto.getElementsByTagName("nama")
     [0].firstChild.data);
    newNama.appendChild(nama_);

    var newFile = document.createElement("div");
    newFile.setAttribute("className", "photo");
    newFile.setAttribute("class", "photo");
    var file_ =
      "<img src='"+currentPhoto.getElementsByTagName("file")[0].
      firstChild.data+"' />";
    newFile.innerHTML= file_;

    var newKeterangan = document.createElement("div");
    newKeterangan.setAttribute("className", "keterangan");
    newKeterangan.setAttribute("class", "keterangan");
    var keterangan_ =
        document.createTextNode(currentPhoto.getElementsByTagName
        ("keterangan")[0].firstChild.data);

    newKeterangan.appendChild(keterangan_);
    newThumb.appendChild(newNama);
    newThumb.appendChild(newFile);
    newThumb.appendChild(newKeterangan);


    document.getElementById("divContent").appendChild(newThumb);

    }
```

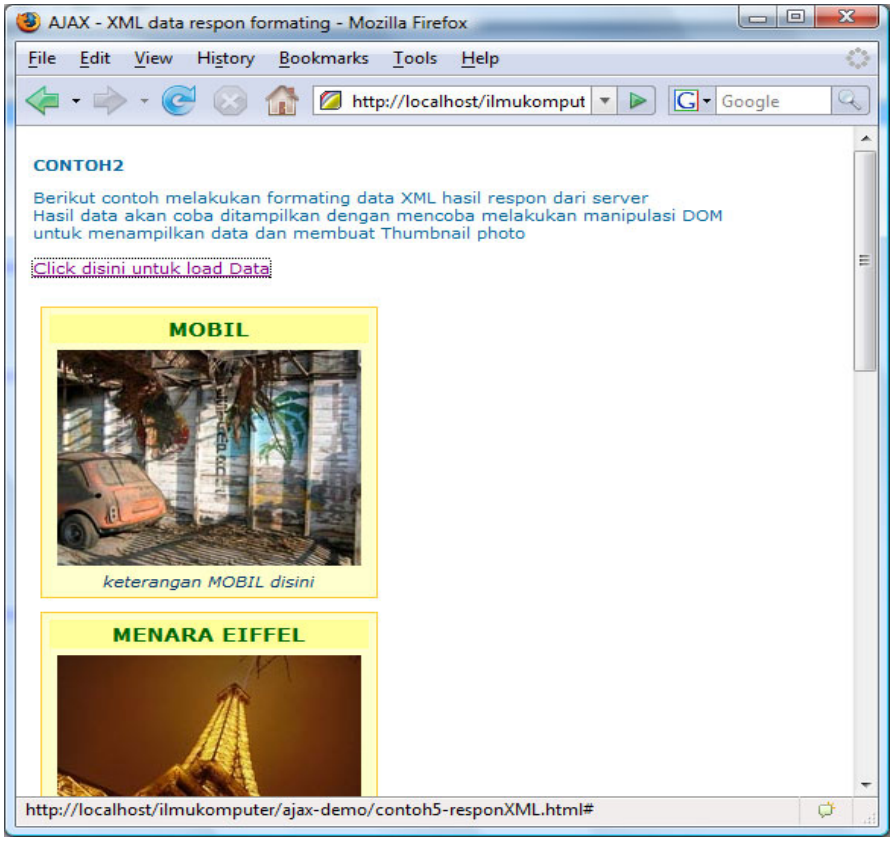
```
            }
        }

 // Send the request
 oRequest.send(null);
}
```

Setelah menambahkan attribute CLASS kedalam masing-masing tag, selanjutnya kita bisa membuat pengaturan CSS untuk untuk masing-masing class tersebut. Class yang ditambahkan pada contoh diatas adalah sebagai berikut:

- thumbnail : sebagai class div masing2 thumbnail photoBook
- nama : sebagai class untuk header nama masing-masing photoBook
- photo: sebagai class untuk photo images
- keterangan: sebagai class untuk keterangan yang tampil dibawah gambar photo.

Selanjutnya terserah pada keinginan anda untuk design format layout CSS class tersebut.
Source lengkap untuk contoh ketiga ini dapat ditemukan pada file contoh5-responXML.html

Hasil contoh ketiga ini pada browser adalah sebagai berikut:

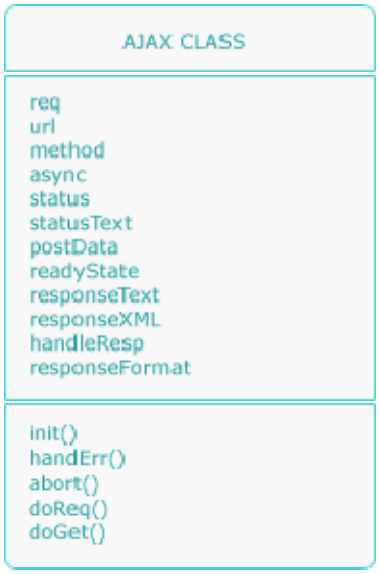


Gambar 8
(Hasil contoh ketiga)

MEMBUAT AJAX CLASS

Pada bagian akhir artikel BERKENALAN DENGAN AJAX ini kita akan membuat sebuah AJAX class yang nantinya bisa digunakan pada setiap aplikasi kita, setelah membuat class ini kita akan mencoba membuat sebuah aplikasi Photo Gallery sederhana.

Berikut design atau gambaran AJAX class yang akan kita buat:



Gambar 9
(Design AJAX class yang akan dibuat)

AJAX class akan tersebut memiliki 12 buah property dan juga 5 buah event, berikut detail penjelasan rancangan untuk masing-masing Property dan juga Method yang akan dibuat.

(Pembuatan class AJAX ini berdasarkan buku karya Mathew Errnisse : Build your own AJAX web application)

Property

Req	object objectXMLHttpRequest
url	lokasi URL file/data
method	Metode request yang ingin digunakan (POST/GET)
async	tipe request asynchronous (TRUE atau FALSE)
status	status error respon server
statusText	text error status respon
postData	Data yang akan dikirimkan ke server
readyState	Status respon dari server
reponseText	Berisi hasil response jika Text Data.
responseXML	Berisi hasil response jika XML Data.
handleResp	Function handle respon
responseFormat	Format respon

Method

init	membuat XMLHttpRequest object
handErr	function error handle
abort	membatalkan request
doRequest	melakukan request
requestProses	membuka proses request

Class AJAX ini akan dibuat kedalam sebuah file javascript yang diberi nama ajax-class.js, dan dari semua Method yang ada pada pengunaanya kita hanya perlu berinteraksi dengan method RequestProses saja dengan mengirimkan halaman yang direquest dan juga function yang ingin dilakukan saat respon dari server telah diterima, Berikut script lengkap pembuatan class tersebut:

ajax-class.js

```
function ajax() {
	this.req = null;
	this.url = null;
	this.method = 'GET';
	this.async = true;
	this.status = null;
	this.statusText = '';
	this.postData = null;
	this.readyState = null;
	this.responseText = null;
	this.responseXML = null;
	this.handleResp = null;
	this.responseFormat = 'text', // 'text', 'xml', or 'object'

	this.init = function() {
	  if (!this.req) {
	       try {
	         // Firefox, Safari, IE7, etc.
	         this.req = new XMLHttpRequest();
	         } catch (e) {
	          try {
	           // IE Versi lama.
	           this.req = new ActiveXObject('MSXML2.XMLHTTP');
	          } catch (e) {
	          try {
	               // IE Versi terbaru.
	               this.req = new
ActiveXObject('Microsoft.XMLHTTP');
	           } catch (e) {
	               // Gagal membuat object XMLHttpRequest.
	               return false;
```

```
                }
            }
        }
    }
    return this.req;
    };

    this.doGet = function(url, hand, format) {
            var self = this;
            self.url = url;
            self.handleResp = hand;
            self.responseFormat = format || 'text';
            self.doReq();
    };


    this.doReq = function() {
            if (!this.init()) {
              alert('Gagal membuat XMLHttpRequest object.');
              return;
            }

            this.req.open(this.method, this.url, this.async);

            var self = this;

            this.req.onreadystatechange = function() {
                    var resp = null;
                    if (self.req.readyState == 4) {

                     switch (self.responseFormat) {
                           case 'text':
                                   resp = self.req.responseText;
                                   break;
                           case 'xml' :
                                   resp = this.req.responseXML;
                                   break;
                           case 'object':
                                   resp = req;
                                   break;
                    }
                    self.handleResp(resp);
                    }
            };

            this.req.send(this.postData);
    };

    this.handleErr = function() {
              alert('ERROR, browser mengaktifkan pop-up blocker \n'
              + 'Status Code: ' + this.req.status + '\n'
              + 'Status Respon: ' + this.responseText + '\n'
              + 'Status Description: ' + this.req.statusText);
```

```
        };

        this.abort = function() {
                if (this.req) {
                  this.req.onreadystatechange = function() { };
                  this.req.abort();
                  this.req = null;
                }
        };
}
```

Sekarang kita coba mengunakan class tersebut dalam sebuah aplikasi sederhana, contoh aplikasi sederhana ini akan melakukan request proses hello.html, file yang direquest ini hanya akan menampilkan proses say Hello dengan AJAX Class

File contoh pengunaan class tersebut adalah sebagai berikut:

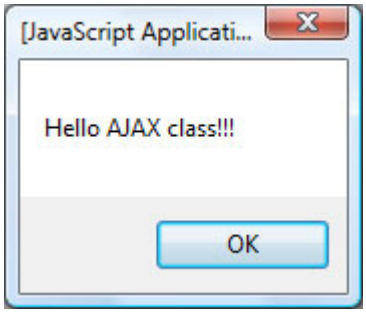
contoh6-testclass.html

```
<html>
<head>
<title>Contoh AJAX</title>
<script language="javascript" src="ajax-class.js"></script>
</head>
        <script language="javascript">
                var handler = function(str) {
                        alert(str);
                }
        </script>
<body>
        <script language="javascript">
                var oAJAX = new ajax();
                oAJAX.doGet("hello.html",handler);
        </script>
</body>
</html>
```

Hello.html

```
Hello AJAX class!!!
```

Setelah selesai membuat ketiga file tersebut letakan semua file tersebut pada folder latihan, kemudian buka file contoh6-class.html pada browser, maka file tersebut akan mencoba membuat object AJAX dan memangil file hello.html.



Gambar 10
(Hasil testing class AJAX)

Setelah berhasil mencoba class AJAX yang baru saja kita buat maka file **ajax-class.js** bisa kita include kedalam setiap project yang ingin mengunakan AJAX proses. Untuk menginclude sebuah file javascript digunakan script berikut pada bagian awal document.

```
<script language="javascript" src="ajax-class.js"></script>
```

Pada bagian *src* kita tuliskan alamat file yang akan diinclude, jika berada pada folder yang sama dengan aplikasi yang mengunakannya maka tidak perlu diikutsertakan path directornya.

Dari class AJAX yang ktia buat tersebut sekarang ktia perlu melakukan beberapa hal berikut untuk mengunakannya:

- Include file *ajax-class.js* tersebut kedalam file project kita
- Buatlah sebuah function yang akan digunakan untuk memproses data yang telah berhasil dikembalikan oleh server
- Create object AJAX baru dari class ajax yang sudah kita buat, untuk membuat object ajax baru digunakan script seperti berikut:

```
var oAJAX = new ajax();
```

- Lakukan request proses dari server dengan mengunakan method *doGet* yang ada pada AJAX CLASS, berikut sintaks penulisannya:

```
oAJAX.doGet(URL,HANDLER);
```

URL: alamat halaman yang direquest dari server
HANDLER: function yang dijalankan saat data respon dari server telah diterima

MEMBUAT AJAX SHOUTBOX

Setelah mencoba membuat class AJAX, sekarang kita mencoba membuat sebuah program Shoutbox yang biasa digunakan pada sebuah weblog untuk memberikan fasilitas pada penunjung meninggalkan pesan mereka.

Source shoutbox ini tersedia pada file contoh-source.zip yang disertakan, dan disini saya hanya membahas secara global saja cara installasi dan proses masing-masing file yang ada pada shoutbox tersebut, kita tidak bahas terlalu detail masing-masing proses yang ada didalamnya.

Program shoutbox ini mengunakan PHP dan juga mySQL sebagai database penyimpanan data, jadi pastikan kedua program tersebut sudah terinstall sebelum mengunakan program shoutbox yang disertakan. Jika belum memiliki program-program tersebut sebaiknya gunakanlah beberapa paket installasi yang banyak tersedia antara lain (XAMPP, appServ, phpTriad dsb).

SPAM :)
Untuk proses installasi XAMPP bisa coba membaca panduan yang juga pernah saya tulis pada ilmukomputer.com, untuk mendownload artikel tersebut bisa dari link berukut:
http://ilmukomputer.com/2007/11/12/xampppaket-apache-php-dan-mysql-instant/

Jika sudah memiliki program-program tersebut sekarang kita bisa coba mengunstall program shoutbox tersebut, berikut langkah yang perlu dilakukan:

1. Copy semua folder shoutbox yang ada pada folder latihan kedalam root folder web server.
2. Buatlah sebuah database MySQL baru dan beri nama sesuai keinginan anda, setelah database terbentuk berikutnya gunakan script yang terdapat pada file **SQL.TXT** untuk membuat struktur table shoutbox didalam database yang telah dibuat, proses ini akan membuat sebuah table shoutbox dan 1 buah record untuk testing hasil data.
3. Langkah berikutnya adalah melakukan setting pada file config.php yang ada pada folder shoutbox, berikut data-data yang perlu disetting didalamnya:
   $core["DB_HOST"] --> diisi dengan nama/ip dimana database berada
   $core["DB_NAME"] --> disi dengan nama database yang dibuat pada tahap 2 diatas
   $core["DB_USER"] --> Nama user untuk masuk kedalam MySQL
   $core["DB_PASS"] --> Password user MySQL
   $core["MAX"] --> Jumlah pesan shoutbox yg akan ditampilkan dilayar
   $core["PATH_IMAGES"] --> Lokasi file2 gambar, sebaiknya tidak perlu dirubah
4. Setelah langkah diatas program shoutbox sudah siap digunakan, cobalah buka file index.php dari webbrowser. Dan cobalah tambahkan beberapa data pesan, saat tombol KIRIM di click data akan automatis muncul dilayar setelah sebelumnya disave pada database.

Gambar 11
(AJAX Shoutbox)

Pada aplikasi shoutbox ini terdapat beberapa file berikut:

- ajax-class.js : File berisi class ajax yang digunakan
- config.php : File config pengaturan koneksi database
- core-java.js : File berisi kumpulan fungsi javascript yang digunakan
- index.php : Berisi file utama yang menampilkan shoutbox
- loadData.php : File yang melakukan proses membaca & formating data shoutbox
- save.php : File yang melakukan proses penyimpanan data saat dikirim
- shoutbox.css : File style CSS yang digunakan untuk pengaturan warna

Aplikasi shoutbox ini bebas untuk digunakan namun mohon tidak membuang banner yang ada dibagian bawah shoutbox :)

PENUTUP

Artikel ini hanya sebagai awal berkenalan sedikit dengan AJAX, disini telah coba kita bahas tentang pembuatan object XMLHttpRequest dan kemudian proses mengunakannya dalam melakukan request secara Asynchronous kepada server. Kemudian juga proses menerima respon dari server baik berupa data TEXT ataupun XML.
Hal-hal tersebut hanya awal pemanfaatan AJAX dalam membuat aplikasi web yang lebih kaya atau banyak orang menyebutnya sebagai pengembangan Rich Internet Application(RIA), dan dari sini juga muncul istilah WEB 2.0 atau pengembangan web generasi kedua.

Dalam WB 2.0 pengembagan aplikasi web bisa menyerupai pengembangan aplikasi dekstop, dimana kita dapat berinterakasi dan berkomunikasi secara asynchronous langsung dengan mesin server penyedia data dan proses.

Saat ini pengembangan aplikasi dengan menafaatkan AJAX dan object XMLHttpRequest ini telah jauh berkembang dari saat pertama kali diperkenalkan, dan telah banyak aplikasi atau framework open source menawarkan pemanfaatan tehnik proses ini. Dari aplikasi-aplikasi open source yang ditawarkan tersebut kita bisa banyak belajar tentang bagaimana mereka melakukan management. proses dan juga mengembangkan object-object yang sangat Fancy dan menarik untuk digunakan.

Semua telah tersedia diinternet, selanjutnya tinggal bagaimana kita mengatur waktu untuk mempelajarinya dan juga mengatur ruang yang terbatas dalam otak ini untuk menampung informasi-informasi pengetahun tersebut.

Have nice coding and learning

RINGKASAN

Dari pembahasan diatas berikut sedikit ringkasannya:

- AJAX bukan sebuah bahasa pemrograman baru
- AJAX merupakan sebuah tehnik pemanfaatan object XMLHttpReguest untuk melakukan komunikasi Asynchronous dengan server.
- Istilah AJAX merupakan singkatan dari Asynchronous JavaScript And XML

Berikut langkah dasar pemanfaatan AJAX dalam pengembagan aplikasi WEB

1. Buat object XMLHttpRequest
2. Membuka komunikasi dengan server dengan menafaatkan method OPEN dari XMLHttpRequest yang telah dibuat
3. Menyiapkan function untuk menerima respon dari server dengan menafaatkan event ONREADYSTATECHANGE
4. Mengisi parameter-parameter pada object XMLHttpRequest yang telah dibuat, kemudian mengirimkan SEND request kepada server

Berikut sedikit catatan tentang object XMLHttpRequest (PROPERTY, METHOD dan EVENT) yang dapat digunakan untuk pengembangan pembuatah AJAX Class

METHOD	
open()	Membuka komunikasi dengan server
send()	Mengirimkan request kepada server
abort()	Membatalkan request
setRequestHeader()	Mengirimkan nilai spesifik untuk header yg dikirimkan
setResponseHeader()	Menerima Respon header dari server
getAllResponseHeader()	Membaca semua respon header dari server
PROPERTY	
status	Status dari respon komunikasi dengan server
statusText	Status yang disertakan pada pengembalian respon text
readyState	Status state yg berisi FLAG indikator komunikasi dengan server
responseText	Hasil respon TEXT data yang dikembalikan oleh server
responseXML	Hasil respon berupa XML data yang dikembalika oleh server
Nilai pada XMLHttpRequest readyState property	
0	Object telah dibuat, namun komunikasi dengan server melalui methode OPEN belum dilakukan

1	Object telah dibuat, komunikasi dengan server melalui OPEN method telah dilakukan namun status request SEND belum dilakukan
2	Status reqeust dengan SEND method telah dilakukan namun belum ada respons yang diberikan oleh server
3	Status respon dari server telah diterima berupa responseBody dan responseText, namun belum semua data yang dikirimkan diterima oleh client browser
4	Status disaat semua data respon dari server telah diterima

Pemanfaatan DOM

- getElementsByTagName : digunakan untuk membaca tag berdasarkan NAMA
- getElementById : digunakan untuk membaca tag berdasarkan ID yang diberikan
- getAttribute : digunakan untuk membaca tag berdasarkan ATTRIBUTE
- firstChild : membaca Node array pertama dari data tag yang kita baca
- lastChild : membaca Node array terakhir dari data tag yang kita baca.
- data : membaca data text dari tag atau node yang kita inginkan
- innerHtml : membaca/menuliskan data kedalam bagian tag, innerHtml berbeda dengan property data sebelumnya karena innerHTML dapat berupa sebuah struktur data yang ada didalam TAG yang ambil, dialamnya mungkin akan dapat berisi tag-tag baru lagi, sedangkan property data hanya dapat digunakan jika data yang diambil berupa Text.
- createElement : menambahkan sebuah element/tag baru kedalam DOM yang telah ada
- appendChild : menambahkan data kedalam struktur element/tag
- CreateTextNode : membuat text node baru kedalam tag/element DOM
- setAttribute: membuat attribute untuk tag/element

Bahan bacaan

- AJAX: A New Approch To Web Applications
  http://www.adaptivepath.com/publications/essays/archives/000385.php
- Mathew Ernisse: Build your own AJAX web application
  http://www.sitepoint.com/launch/53fc13

Referensi

- www.w3c.org
- Marini, Joe (2005): Developing AJAX aplication
- Morgan, Kaufman (2007): Unleashing WEB 2.0, from concept to creativity
- Babin, Leen (2007): Beginning Ajax with PHP: From Novice to Professional

OPEN SOURCE AJAX FRAMEWORK

- Dojo : www.dojotoolkit.com
- Yahoo User Interface : http://developer.yahoo.com/yui/
- Adobe spry : http://labs.adobe.com/technologies/spry/
- Mochikit : http://www.mochikit.com
- Qooxdoo.org : http://qooxdoo.org/

Dan masih banyak lagi, kalau ada yang mengetahui open source lain seputar AJAX tolong di share juga infonya. thanks

Biografi Penulis

Jerry Peter Saerang. Lahir di jakarta 15 Januari 1979. Menyelesaikan studi S1 di Universitas Gunadarma, Jakarta. Dan saat ini dalam proses menyelesaikan Thesis pada program Studi S2 di Universitas Gunadarma.
Saat ini bekerja sebagai System Analyst & Software Developer di sebuah perusahaan Farmasi Nasional.

Informasi lebih lanjut tentang penulis bisa di dapat melalui:
Email: jerry.peter@gmail.com
Blog: www.ruangkecil.or.id