Bab II

Landasan Teori

II.1. Sistem Informasi

Sebuah sistem informasi mengoleksi, memproses, menyimpan, menganalisis, dan mengantarkan informasi untuk suatu tujuan tertentu. Seperti sistem pada umumnya, sebuah sistem informasi terdiri atas *input* (data, instruksi) dan *output* (*report*, kalkulasi). Sistem memproses *input* dan menghasilkan *output* yang kemudian akan dikirimkan kepada *user* atau kepada sistem yang lainnya. Sebuah mekanisme umpan balik juga ditambahkan dalam sebuah sistem informasi yang digunakan untuk mengatur jalannya suatu operasi. Sistem informasi hanya dapat berjalan atau digunakan dalam lingkungannya.

Dalam mempelajari suatu sistem informasi, perlu diperhatikan perbedaan antara data, informasi, dan *knowledge*. Data merupakan sekumpulan fakta atau elemen yang mendeskripsikan tentang sesuatu, kejadian, aktivitas, dan transaksi yang ter-*capture*, tercatat, tersimpan, dan terklasifikasi, tetapi tidak terorganisir dan tidak untuk menjelaskan tentang sesuatu arti secara spesifik. Contohnya data *grade point average*, data *bank balance*, atau jumlah jam kerja pegawai dalam 1 periode. (Efraim Turban, R. Kelly Rainer Jr., Richard E. Potter, 2003)

Informasi merupakan sekumpulan data atau fakta yang telah di atur dalam suatu cara sehingga fakta dan data tersebut memiliki arti bagi penerima data. Contohnya, bila data nama siswa digabungkan dengan

data *grade point averages*, data nama *costumer* dengan *bank balance*, gaji dengan jumlah jam kerja, maka informasi tersebut akan lebih memiliki arti dibanding sebelumnya. Dengan kata lain, informasi berasal dari data yang telah diproses. (Efraim Turban, R. Kelly Rainer Jr., Richard E. Potter, 2003)

Knowledge terdiri atas informasi yang telah diatur dan diproses untuk memberikan penjelasan mengenai suatu pengertian, *experiences*, *accumulated learning*, atau *expertise* dimana informasi akan digunakan dalam *business problem* atau suatu proses. Informasi yang telah diproses dapat digunakan untuk menjabarkan implikasi-implikasi yang bersifat kritis, memberikan gambaran tentang peristiwa yang pernah terjadi. (Efraim Turban, R. Kelly Rainer Jr., Richard E. Potter, 2003)

Informasi akan sangat berguna bagi seorang *manager* dalam suatu instansi, oleh karena itu informasi seharusnya dapat menunjukan berbagai macam karakteristik. Informasi harus akurat, lengkap, fleksibel, handal, relevan, tepat waktu, *verifiable*, *accessible*, dan *secure*. Informasi yang tidak memenuhi klasifikasi diatas dapat memperbesar kemungkinan kesalahan dalam pengambilan keputusan, dan dapat menjadikan organisasi tersebut sebagai sebuah *"cost center"*. (Efraim Turban, R. Kelly Rainer Jr., Richard E. Potter, 2003)

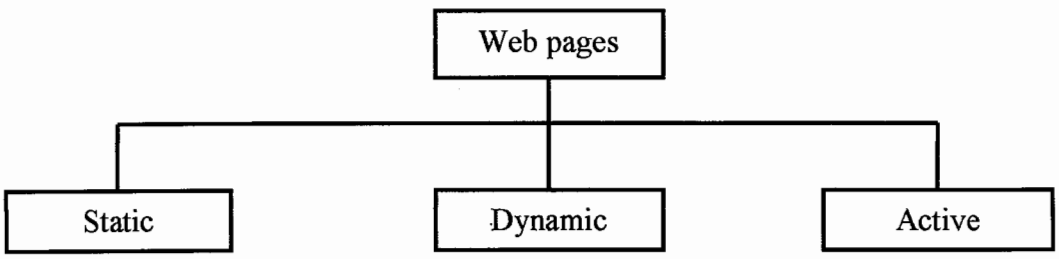
II.2. Sistem Informasi Berbasis Web

Saat ini komputer dan piranti pendukungnya telah masuk ke dalam aspek kehidupan dan pekerjaan. Komputer yang ada sekarang ini memiliki kemampuan yang lebih dari sekedar perhitungan matematika biasa.

Interconnected network yang biasanya sering disebut dengan Internet adalah sebuah sistem komunikasi yang global yang menghubungkan komputer dan jaringan-jaringan komputer diseluruh dunia. Komputer dan jaringan dengan berbagai platform yang memiliki perbedaan dan ciri khas masing-masing bertukar informasi dengan sebuah standar yang dikenal dengan nama TCP/IP (*Transmission Control Protocol/Internet Protocol*)

Web adalah fasilitas *hypertext* untuk menampilkan data berupa teks, gambar, bunyi, animasi, dan data multimedia lainnya, yang mana data tersebut saling berhubungan satu sama lainnya.

Berbagai macam teknologi *Web* telah muncul beberapa tahun yang lalu oleh karena itu teknologi *Web* kemudian dikategorikan menjadi 3 kategori utama yaitu: *static, dynamic* dan *active*. Berikut gambar peng-kategori-an teknologi *Web*:



Gambar 2.1 : Pengkategorian Web
(Sumber: Achyut S. Godbole, Atul Kahate, 2003)

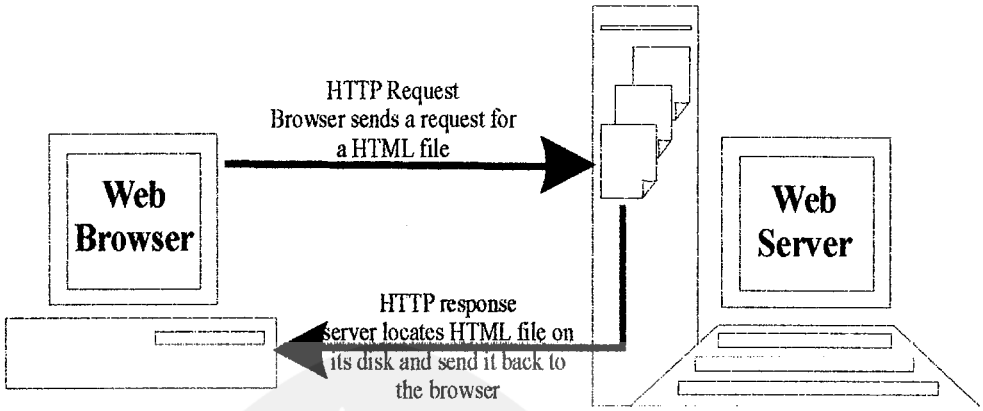
III.2. 1 Web Statis

Pada mulanya *Web* statis hanya terdiri atas teks saja sehingga pengguna *Web* statis hanya terdiri atas kalangan akademisi maupun peneliti dalam suatu institusi. Namun perkembangan Internet yang sangat pesat hingga menyamai kebutuhan akan pesawat telepon, menyebabkan layanan Internet dapat diakses oleh orang banyak. Namun dari waktu ke waktu orang-orang yang mengakses *Web* mulai merasa bosan karena halaman *Web* yang diakses hanya terdiri atas teks dan *link* saja (Achyut S. Godbole, Atul Kahate, 2003)

Agar halaman *Web* tersebut tidak terasa membosankan dan supaya terlihat lebih menarik maka halaman *Web* tersebut disisipi beberapa gambar sehingga *Web* tersebut memiliki unsur multimedia di dalamnya. Dengan penambahan gambar sebagai unsur multimedia di dalam suatu halaman *Web* maka *user* dapat mengakses halaman *Web* yang terdiri atas teks dan gambar. (Achyut S. Godbole, Atul Kahate, 2003)

Pada kenyataannya halaman *Web* diatas sudah cukup menarik. Namun halaman *Web* seperti yang dipaparkan diatas masih bersifat statis. *Web* yang bersifat statis adalah *Web* yang tidak dapat berubah dari waktu ke waktu. Hal ini mirip dengan halaman buku bacaan dimana dalam sekali cetak maka isinya akan tetap sama, hingga penerbitnya menerbitkan buku edisi berikutnya maka kita dapat melihat bahwa terdapat perubahan dalam buku tersebut. Begitu pula dengan halaman *Web* yang

bersifat statis, berikut ini gambar dari *Web* statis. (Achyut S. Godbole, Atul Kahate, 2003)



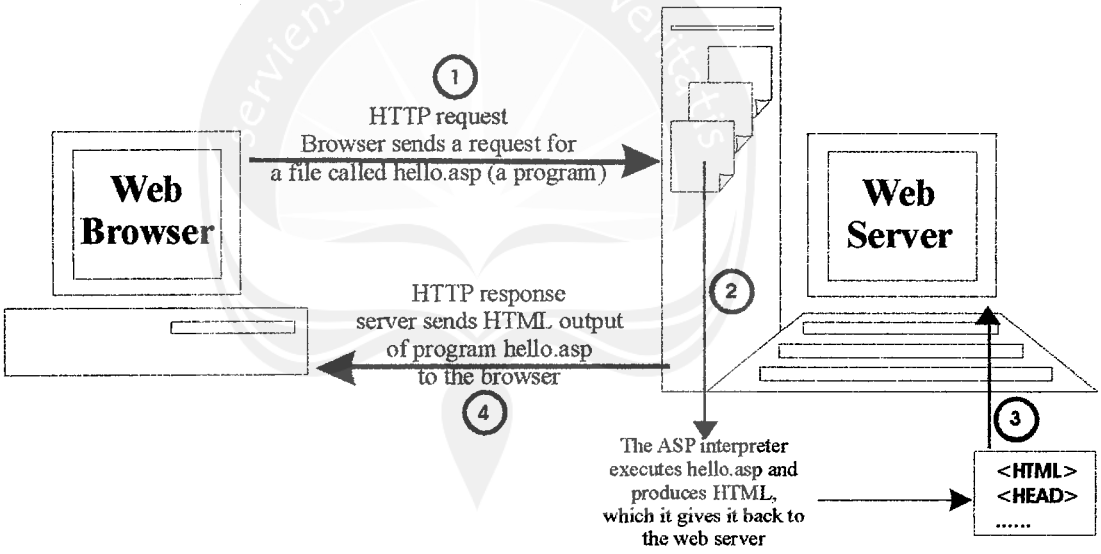
Gambar 2.2 : Static Web
(Sumber: Achyut S. Godbole, Atul Kahate, 2003)

III.2. 2 Web Dinamis

Pada *Web* statis ketika *browser* meminta *server* untuk mengirimkan sebuah halaman *Web* dengan format HTML, *server* tidak melakukan pengeksekusian terhadap program apapun. Dalam suatu halaman *Web* yang dinamis akan terjadi pengeksekusian terhadap suatu program yang ditunjukkan oleh URL ketika *browser* melakukan *request* ke *server* dan server mengirimkan *output*nya dalam format HTML ke *Web browser*. (Achyut S. Godbole, Atul Kahate, 2003)

Web dinamis akan memberikan alamat dari sebuah halaman *Web* pada *server*, yang berisi program-program yang dapat dieksekusi. Contoh program-program tersebut antara lain ASP, JSP, atau CGI atau disebut juga dengan bahasa pemrograman *server*-side, artinya program-program tersebut akan dieksekusi pada sisi *server* kemudian *output*nya akan diterjemahkan dalam bentuk HTML dan kemudian

dikirimkan kembali ke *Web browser* untuk ditampilkan. Ketika *browser* mengirimkan permintaan terhadap halaman *Web* dinamis menggunakan URL yang mengacu pada bahasa pemrograman *server*-side yang digunakan dalam *Web* dinamis, maka *Web server* akan menggunakan URL tersebut untuk mencari lokasi program, dan mengeksekusi program tersebut dengan bantuan *interpreter* yang bersesuaian dan menghasilkan *output* yang akan diterjemahkan dalam bentuk HTML yang kemudian akan dikirimkan kembali ke *browser*, berikut ini gambar dari *Web* dinamis. (Achyut S. Godbole, Atul Kahate, 2003)

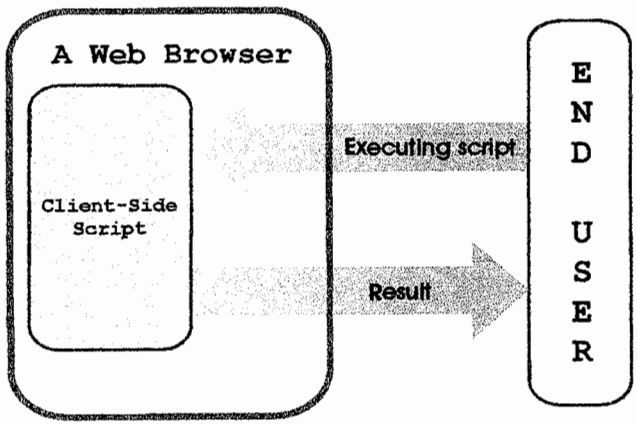


Gambar 2.3 : Dynamic Website
(Sumber: Achyut S. Godbole, Atul Kahate, 2003)

Client-side biasanya merupakan sekumpulan operasi yang berjalan disisi client. Biasanya client merupakan aplikasi komputer seperti web browser, yang berjalan disisi local komputer atau *workstation*, dan terhubung dengan server bila diperlukan.

Operasi yang dijalankan user karena client memerlukan informasi atau fungsionalitas yang akan diakses disisi client tetapi tidak berjalan disisi server misalnya pengecekan inputan yang dilakukan user, validasi email, validasi tanggal, dsb. Operasi dilakukan disisi client karena server tidak cukup powerful dalam melayani permintaan-permintaan yang membutuhkan ketepatan waktu. Sehingga jika operasi-operasi dapat dijalankan oleh client, tanpa melakukan pengiriman data ke server, maka waktu yang dibutuhkan untuk menjalankan operasi tersebut menjadi lebih kecil, dan hemat *bandwidth*

Untuk menjalankan operasi yang dapat berjalan disisi client dalam hal ini web browser, maka diperlukan client-side scripting yang berjalan di web browser. Client-side scripting merupakan bagian dari Dynamic HTML (DHTML). Client side script biasanya dibubuhkan di dalam dokumen HTML, tetapi bisa juga dijadikan file eksternal, sehingga terpisah dari dokumen HTML



Gambar 2.4. Client-Side Script

III.2. 3 Web Aktif

Kemunculan halaman *Web* aktif sebenarnya untuk menjawab beberapa permasalahan yang terungkap dalam pertanyaan-pertanyaan berikut:

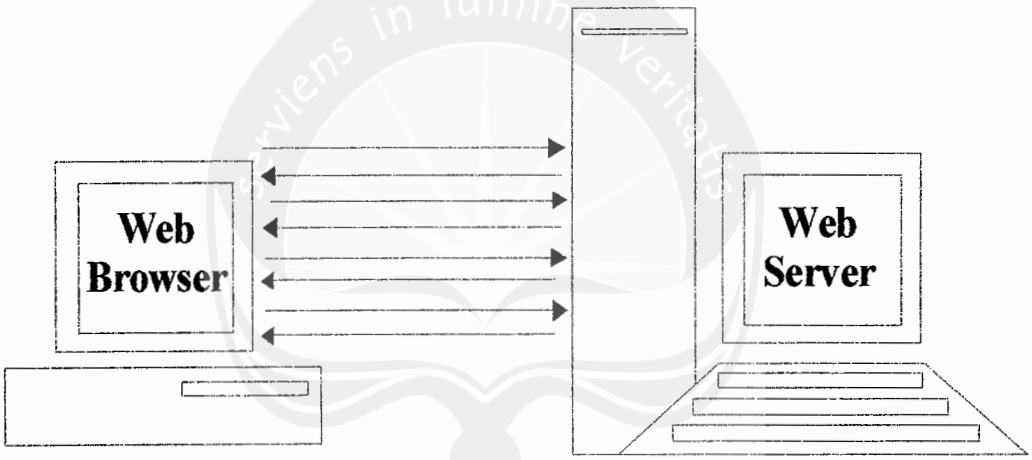
1. mengapa *client* harus lama menunggu untuk dapat memperoleh *image* di Internet?
2. mengapa halaman *Web* di Internet tidak dapat menampilkan gambar video ataupun suara yang sebenarnya seperti di televisi?

Pertanyaan diatas mengungkapkan permasalahan yang dihadapi Internet yaitu Internet hanya dapat menjalankan dokumen yang bersifat statis dan tidak dapat menampilkan gambar video. Kemudian munculah solusi untuk mengatasi permasalahan diatas yang dikenal dengan nama *client-pull*. (Achyut S. Godbole, Atul Kahate, 2003)

Cara kerja teknik *client pull* yaitu *browser* yang berjalan disisi *client* akan melakukan *request* secara berulang-ulang ke *server* untuk halaman *Web* yang berisi sekumpulan *image* dan ketika halaman *Web* tersebut diterima, maka *image* tersebut akan ditampilkan satu per satu, dimana *image* yang ditampilkan memiliki kapasistas 30 per detik, oleh karena itu proses *client-pull* terasa lamban. (Achyut S. Godbole, Atul Kahate, 2003)

Permasalahan teknik *client-pull* baru dapat dirasakan ketika teknik ini diimplementasikan pada *Web-Web* yang terhubung dengan jaringan Internet. Masalah yang dijumpai pada teknik *client-pull* ini adalah masalah kecepatan transmisi data. Meskipun suatu *browser* dijalankan pada komputer yang

memiliki teknologi paling canggih dengan menggunakan koneksi Internet berkecepatan tinggi namun tetap saja tidak dapat melakukan transmisi data dengan cepat. Contohnya gambar animasi yang seharusnya memiliki gerakan yang lembut akan tampak putus-putus atau terjadi *delay* antara gerakan yang satu dengan gerakan yang lainnya. Dalam hal ini *browser* membutuhkan waktu untuk men-*transfer* data agar dapat memperoleh data tersebut secara utuh. (Achyut S. Godbole, Atul Kahate, 2003)



Client Pull

Gambar 2.5 : Client Pull

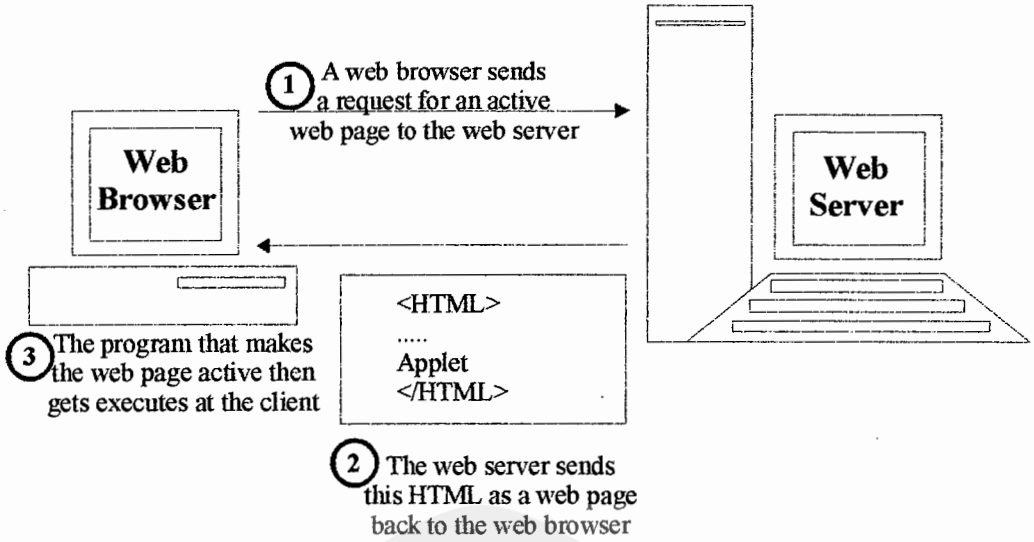
(Sumber: Achyut S. Godbole, Atul Kahate, 2003)

Masalah yang timbul ketika menggunakan teknik *client-pull* dapat diatasi dengan menggunakan teknologi *Web* aktif. Teknologi *Web* aktif dapat menghasilkan gerakan animasi tanpa harus menyisipkan *delay* antara gerakan yang satu dengan yang lainnya. Hal inilah yang kemudian menjadi tujuan dibuatnya halaman *Web* yang bersifat aktif. Halaman *Web* ini berisi program-program untuk membentuk dan menjalankan sebuah *image* atau

animasi yang akan ditampilkan dilayar monitor. Program ini akan dieksekusi pada sisi *client* tepatnya pada *Web browser*. Berikut ringakasan cara kerja *Web* aktif :

1. Sama halnya dengan halaman *Web* statis dan dinamis, halaman *Web* aktif juga disimpan di *Web server*. Dimana setiap halaman *Web* aktif akan berasosiasi dengan URL sebagai penunjuk lokasi dimana halaman *Web* aktif tersebut disimpan
2. Untuk memperoleh halam *Web* aktif, *browser* akan menunjuk URL dari sebuah halaman *Web* aktif, kemudian *Web server* akan mencari lokasi *file* yang berisi halaman *Web* aktif dan mengirimkannya kembali ke *client*.
3. Pada halaman *Web* aktif, *browser* sesungguhnya menjalankan program komputer yang ditempelkan pada halaman *Web* aktif. Program ini digunakan untuk membentuk dan menjalankan sebuah *image* atau animasi. Oleh karena itu halaman *Web* aktif dieksekusi di sisi *client* bukan di sisi *server*, sehingga halaman *Web* dapat di*download* dengan cepat.

(Achyut S. Godbole, Atul Kahate, 2003)



Gambar 2.6 : Active Website
(Sumber: Achyut S. Godbole, Atul Kahate, 2003)

II.3. Asynchronous Javascript and XML

AJAX merupakan teknik pengembangan *Web* yang muncul pada tahun 2005 tepatnya pada pertengahan februari. AJAX menjadi bahan pembicaraan yang hangat di kalangan *Web* developer sejak Jesse James Garret dari adaptive path menggunakan istilah AJAX dalam artikelnya tentang pendekatan baru dalam aplikasi *Web*. (www.telerik.com, 2005)

AJAX muncul sebagai suatu pendekatan untuk meminimalkan "garis-garis kesenjangan" antara aplikasi yang berbasis *Web* dan aplikasi yang berbasis *dekstop*. Berikut ini kesenjangan yang ada antara aplikasi berbasis *Web* dan aplikasi berbasis *dekstop*:

a. *Poor Interactivity*

Aplikasi *Web* klasik mengharuskan seorang *user* untuk duduk dan menunggu untuk sebuah

halaman *Web* yang lengkap setiap kali melakukan interaksi dengan *user*

b. *Unresponsiveness*

Aplikasi *Web* klasik melakukan *transfer* data ke *server* kemudian *server* akan memprosesnya dan mengirimkannnya kembali ke *browser* dalam bentuk HTML. Hal ini akan terus terjadi selama ada permintaan akan sebuah halaman *Web* ke *server* dan *server* akan memprosesnya dan mengirimkannya kembali ke *browser*, jika terus berlangsung maka kejadian ini tidak efisien karena membutuhkan *bandwith* yang besar dan dapat mengganggu *performannce* jaringan secara signifikan

c. *Simplistic Interfaces*

Kebutuhan akan sebuah halaman *Web* yang lengkap kapanpun akan mengubah tampilan *Web user* interface namun seringkali hal tersebut terbentur pada keterbasan tingkat kemajuan *Web user* interface. Contoh halaman *Web* yang menarik diimplementasikan dengan menggunakan teknologi flash. Namun hal ini sangat tidak praktis karena cara ini sangat kompleks dan membutuhkan ketrampilan khusus yang harus dimiliki oleh seorang *Web developer*. Pedekatan ini juga tidak praktis karena memaksakan seorang *end-user* untuk memasang sebuah *plug-in* agar teknologi ini dapat bejalan dengan semestinya.

d. *Low Usability*

Bila *user* melakukan suatu aksi terhadap *Web* aplikasi, kemudian *Web* me-*reload* seluruh halaman seluruh halaman *Web*, hal ini akan membuat *user* dari halaman *Web* tersebut menjadi terganggu.

(www.telerik.com, 2005)

Ide utama dari munculnya AJAX adalah membuat komunikasi dengan *server* berjalan secara *asynchronous*, sehingga data dapat di*transfer* dan diproses "di belakang layar". Hal ini agar *user* dapat terus bekerja pada sebuah halaman *Web* tanpa terganggu suatu interupsi. Dalam *AJAX* hanya elemen halaman tertentu dan perlu saja yang akan di*update*. (www.telerik.com, 2005)

Berbeda dengan komunikasi *client-server* secara *synchronous*, dimana *Web* aplikasi perlu untuk melakukan *reload* data secara keseluruhan setiap kali data di*transfer* dari/ke *server*. Hal ini menimbulkan beberapa dampak buruk antara lain:

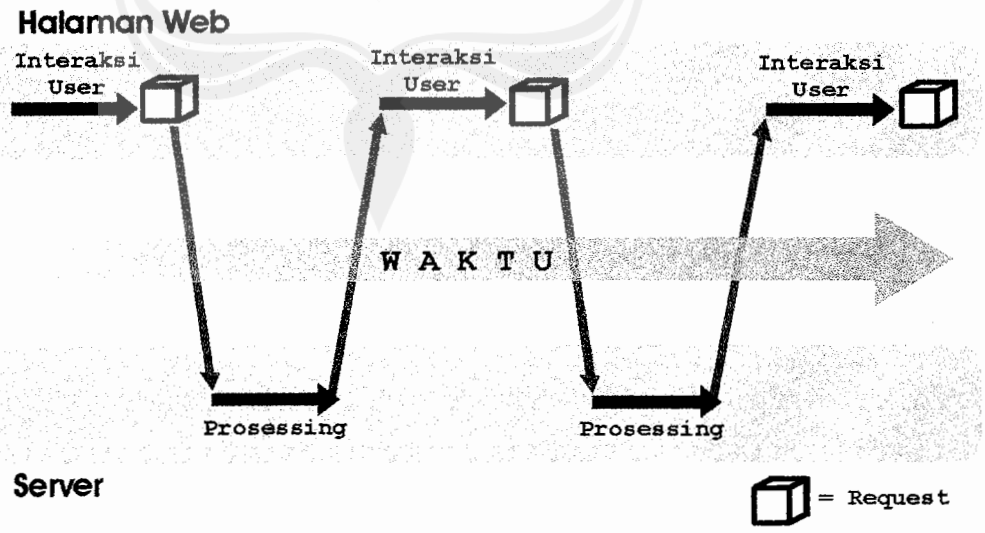
a. interaksi antara *user* dan aplikasi *Web* selau terinterupsi setiap kali melakukan komunikasi dengan *server*
b. *user* harus menunggu dan selalu dihadapkan pada "*blank screen*" selama proses *postback*
c. sebuah halaman *Web* yang lengkap akan diterjemahkan lebih dahulu ke dalam format HTML dan kemudian di*transfer* setiap kali akan terjadi proses *postback*, hal ini memakan

waktu dan menurunkan kualitas lalu lintas jaringan

(www.telerik.com, 2005)

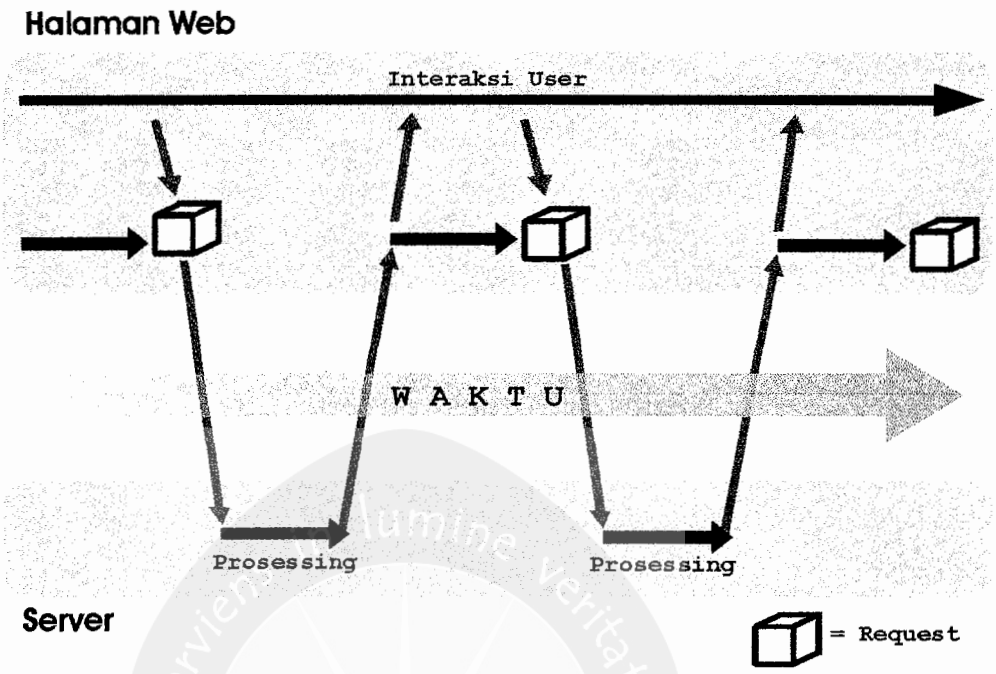
Metode yang digunakan *AJAX* untuk komunikasi antara *client* dan *server* menggunakan metode *asynchronous*. Metode ini dimplimentasikan dengan menggunakan AJAX *engine* yang akan dibangkitkan pada sisi *client* ketika initial page dipanggil. Kemudian *ajax engine* akan berperan sebagai mediator yang mengirimkan data yang relevan ke *server* dalam bentuk XML dan kemudian respon yang diberikan *server* akan diproses untuk meng*update* elemen dari halaman *Web* yang perlu di*update*. (www.telerik.com, 2005)

Berikut diagram *Web* berbasis yang tidak berbasis *AJAX*:



Gambar 2.7 : Cara Kerja Tanpa AJAX
(www.ajaxinfo.com)

Berikut diagram *Web* berbasis *AJAX*:



Gambar 2.8 : Cara Kerja Sistem Berbasis AJAX
(www.ajaxinfo.com)

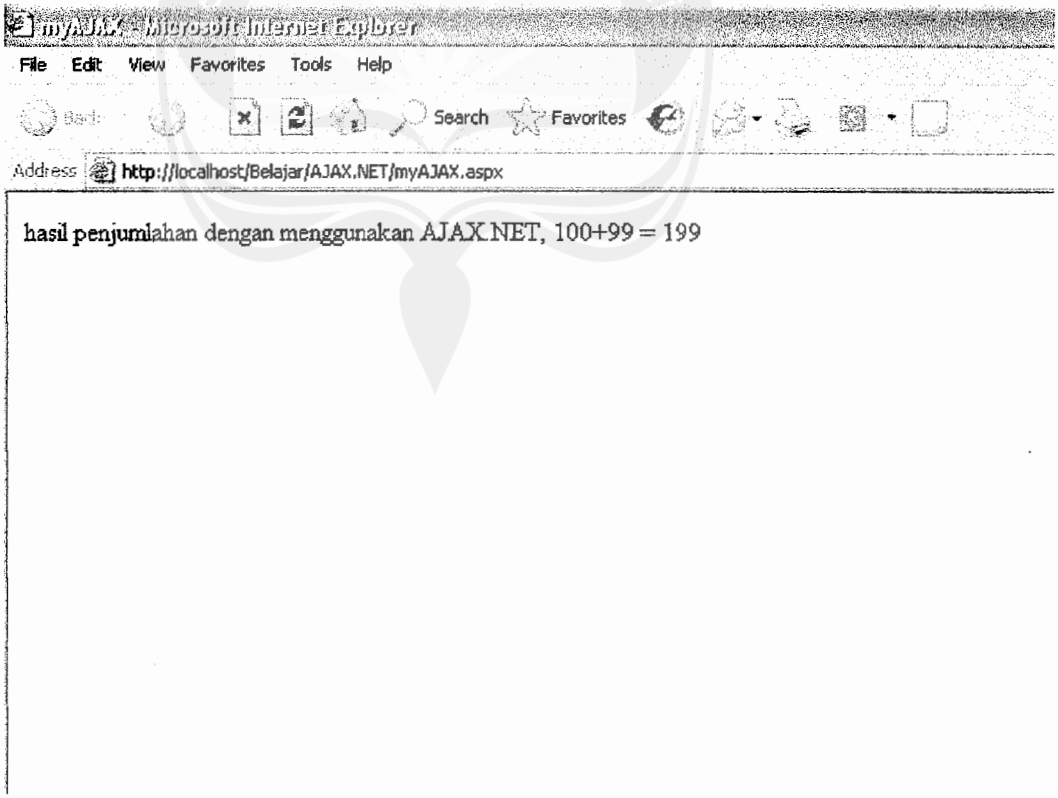
a. Inisial *request* yang dilakukan oleh *browser*, dimana *client* me-*request* suatu URL ke *server*.
b. Halaman *Web* yang lengkap dikirimkan oleh *server* ke *Web browser* yang terdapat disisi *client*
c. Semua *request* yang akan dikirim ke *server* kemudian di inisialisasikan sebagai suatu *function call* ke AJAX *engine*
d. AJAX *engine* kemudian akan membuat sebuah XmlHttpRequest yang kemudian dikirim ke *server*
e. *Server* memproses *request* tersebut dan mengirimkan responnya ke *client* dalam

format XML. Dokumen XML yang dikirimkan oleh *server* hanya terdiri atas data-data yang diperlukan untuk mengubah elemen-elemen tertentu dari suatu halaman *Web*.

f. AJAX *engine* memproses *response* dari *server*, kemudian meng*update* bagian tertentu dari suatu halaman *Web* yang perlu di*update* atau melakukan operasi terhadap data yang diterima dari *server*.

(www.telerik.com, 2005)

Berikut ini contoh penerapan aplikasi web yang menerapkan aplikasi AJAX dengan menggunakan AJAX.NET dan ASP.NET:



berikut ini langkah-langkahnya :

1. Membuat referensi AJAX dengan file ajax.dll

2. Melakukan setting HttpHandler di web.config

```
<configuration>
   <system.web>
      <httpHandlers>
        <add verb="POST,GET" path="ajax/*.ashx"
           type="Ajax.PageHandlerFactory, Ajax"
/>
        </httpHandlers>
    ...
   <system.web>
</configuration>
```

3. Menambahkan fungsi berikut pada page_load event

```
Public Class myAJAX
  Inherits System.Web.UI.Page

  Private Sub Page_Load(sender As Object, e As
EventArgs) Handles MyBase.Load

Ajax.Utility.RegisterTypeForAjax(GetType(Index))
          '...
  end sub
  '...
End Class
```

4. Menambahkan javascript pada halaman web untuk memanggil fungsi RegisterTypeForAjax, berikut code javascriptnya :

```
<head>
  <script language="javascript"
src="ajax/common.ashx"></script>
  <script language="javascript"

src="ajax/Belajar.myAJAX,Belajar.ashx"></script>
 </head>
```

5. Membuat fungsi yang akan dipanggil oleh client, dimana fungsi ini diletakkan di server

```
<Ajax.AjaxMethod()>
```

```
Public Function ServerSideAdd (byval
firstNumber As Integer, byval secondNumber _

As Integer) As Integer
 Return firstNumber + secondNumber
End Function
```

6. Melakukan pemanggilan fungsi di sisi client

```
<%@ Page Inherits="Belajar.myAJAX"
Codebehind="myAJAX.aspx.vb" ... %>
<html>
 <head>
  <script language="javascript"
src="ajax/common.ashx"></script>
  <script language="javascript"
          src="
ajax/Belajar.myAJAX,Belajar.ashx "></script>
 </head>
  <body>
    <form id="Form1" method="post"
runat="server">
      <script language="javascript">
        var
response=myAJAX.ServerSideAdd(100,99);
        document.write("hasil penjumlahan
dengan menggunakan AJAX.NET, 100+99 = " +
response.value);
      </script>
    </form>
  </body>
</html>
```