AJAX

Asynchronous JavaScript And XML

by Rewadee Piputsoongnern

AJAX ?

- AJAX ย่อมาจาก Asynchronous JavaScript And XML
- AJAX ไม่ใช่ภาษาใหม่ในการเขียนโปรแกรม
- AJAX เป็นเทคนิคในการเขียนโปรแกรมที่ใช้ออบเจ็กต์ XMLHttpRequest ของ JavaScript ในการแลกเปลี่ยนข้อมูลบางส่วนกับเซิร์ฟเวอร์ที่นำมาแสดงบนหน้าเว็บ
- AJAX ทำให้ไม่ต้องโหลดข้อมูลบนหน้าเว็บใหม่ทั้งหมด แต่จะโหลดข้อมูลบางส่วน จึงทำให้หน้าจอกระพริบเฉพาะส่วนที่เปลี่ยนแปลงเท่านั้น
- AJAX ทำให้การแสดงผลเร็วขึ้น

AJAX

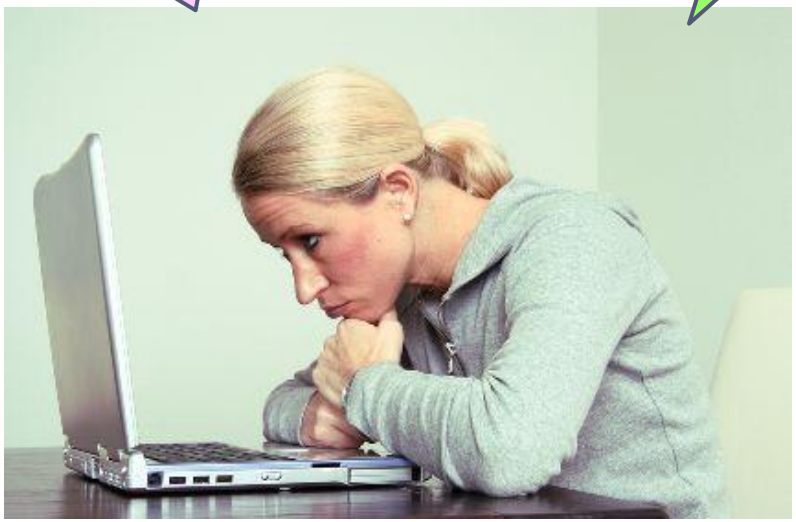
AJAX นำเอาเทคนิคต่าง ๆ รวมเข้าด้วยกัน ประกอบด้วย

- XHTML และ CSS สำหรับการแสดงผล
- DOM (Document Object Model) สำหรับการแสดงผลโต้ตอบแบบไดนามิก
- XML และ XSLT สำหรับการแลกเปลี่ยน แก้ไข และแสดงข้อมูล
- XMLHttpRequest สำหรับการติดต่อแบบ Asynchonous กับเซิร์ฟเวอร์
- JavaScript สำหรับรวมเทคโนโลยีทั้งหมดเข้าด้วยกัน

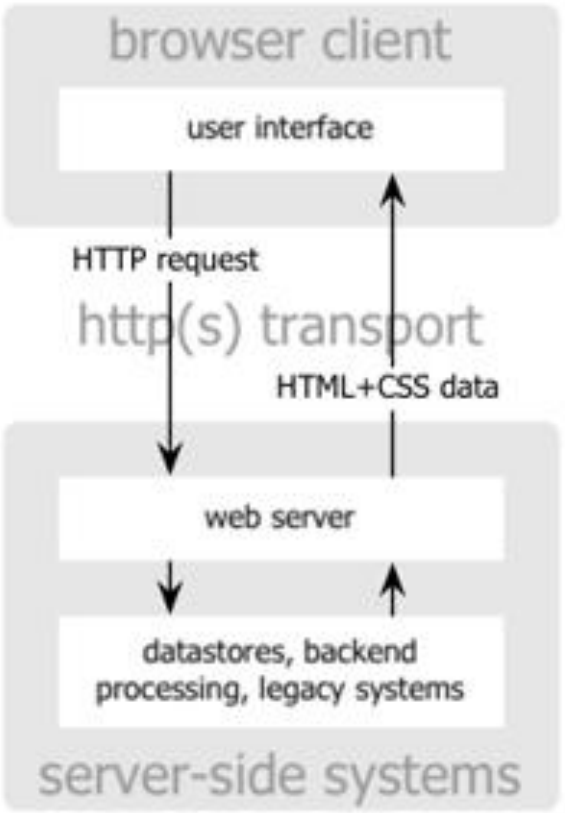
ที่มา และ ปัญหา

"Click, wait, and refresh"
user interaction paradigm

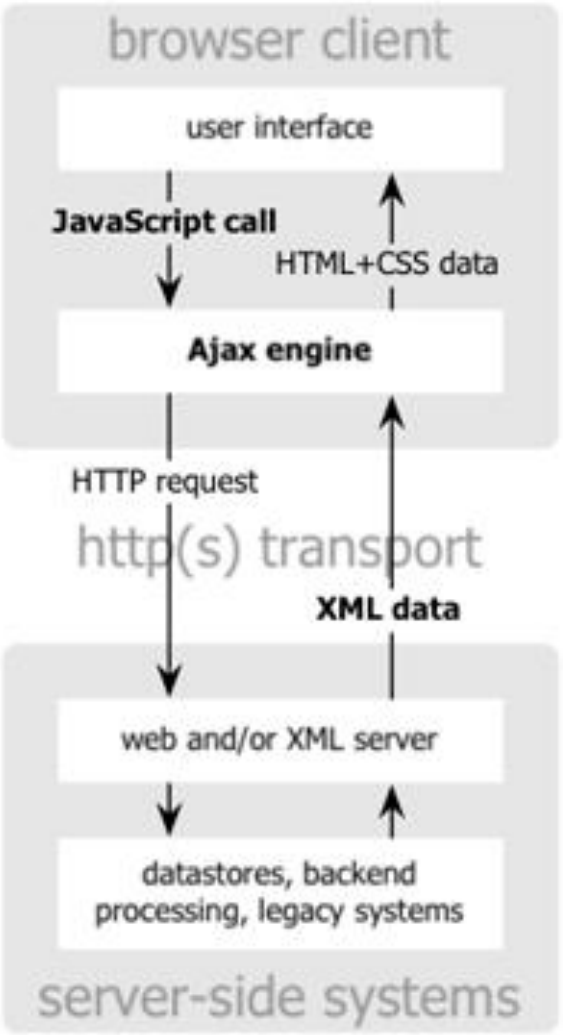
Synchronous
"request/response"
communication mode



AJAX

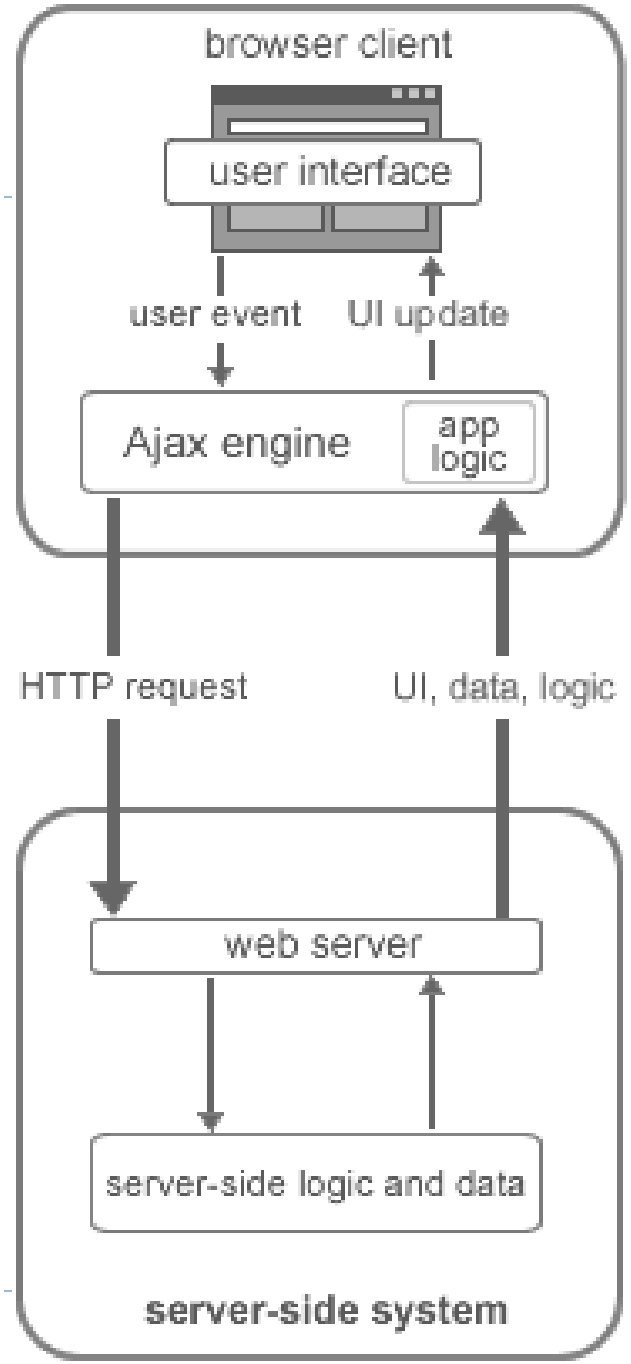


classic
web application model

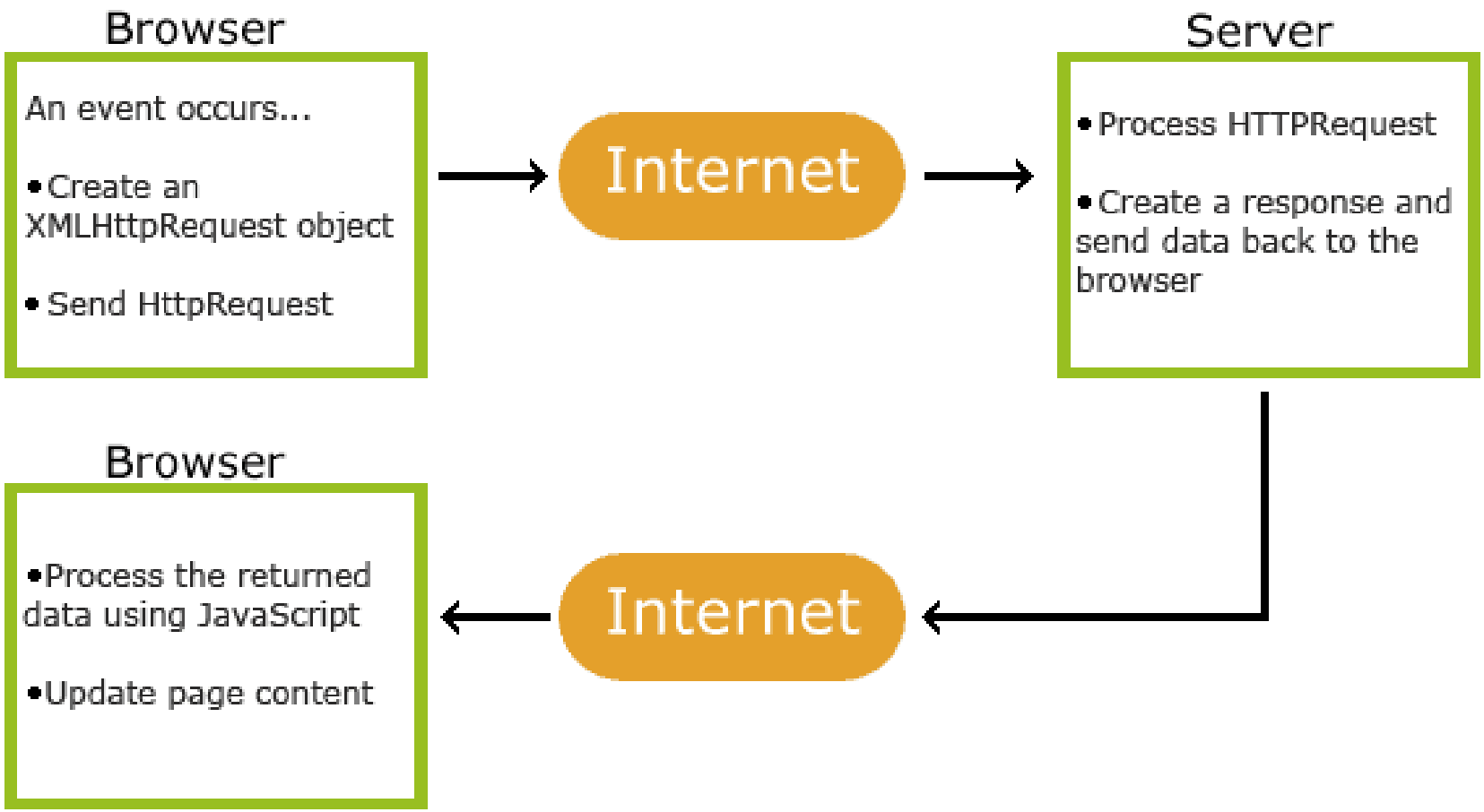


Ajax
web application model

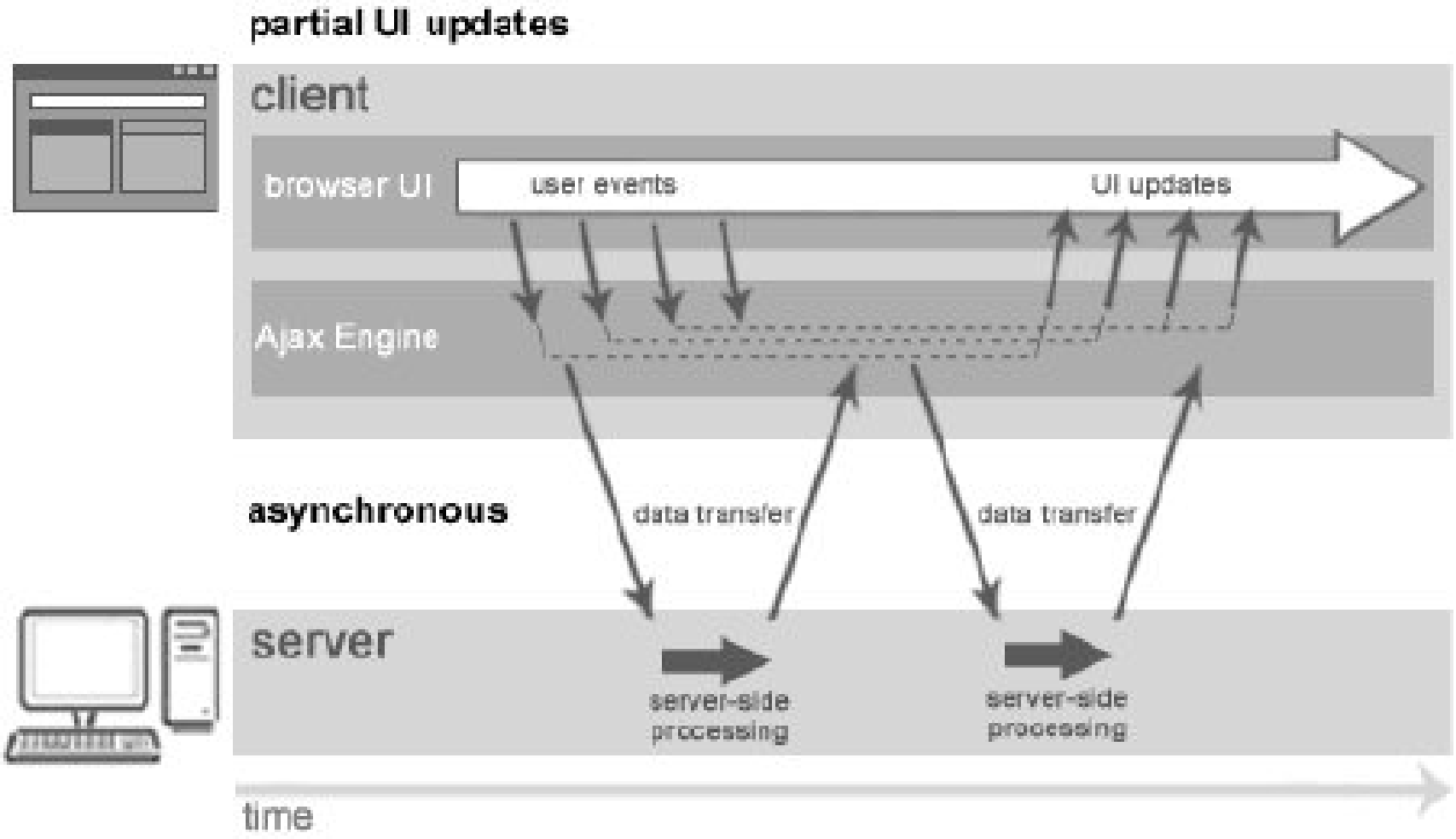
โครงสร้างของ AJAX



การทำงานของ AJAX



การทำงานของ AJAX



XMLHttpRequest ?

- คือกลุ่มของ API (Application Program Interface) ที่สามารถถูกเรียกมาใช้งานผ่านเว็บบราวเซอร์จากเครื่องผู้ใช้โดยสคริปต์ต่าง ๆ เช่น JavaScript , VBScript เป็นต้น
- โดยสร้างช่องทางในการติดต่อสื่อสารระหว่างหน้าเว็บทางฝั่งไคลเอนต์กับทางเซิร์ฟเวอร์ แล้วถ่ายโอนข้อมูลที่เป็น XML , HTML หรือข้อมูลที่เป็นเท็กซ์ ไป/มาจากเว็บเซิร์ฟเวอร์ ซึ่งเป็นการทำงานในเบื้องหลัง โดยใช้ HTTP

การสร้างออบเจ็กต์ของ XMLHttpRequest

- รูปแบบคำสั่ง

```
variable = new XMLHttpRequest();
```

- ** ถ้าเป็น IE รุ่นเก่า เช่น IE5 หรือ IE6 จะใช้ ActiveX object จะใช้คำสั่ง

```
variable=new ActiveXObject("Microsoft.XMLHTTP");
```

ตัวอย่างการสร้าง XMLHttpRequest object

```
var xmlHttp;
if (window.XMLHttpRequest)
  { // code for IE7+, Firefox, Chrome, Opera, Safari
  xmlHttp=new XMLHttpRequest();
  }
else
  { // code for IE6, IE5
  xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
  }
```

Workshop1 ตัวอย่างการใช้งาน XMLHttpRequest

```
<html>
<head>
<title>Workshop : AJAX 1</title>
<script language="javascript" type="text/javascript">
function loadAJAX()
{
        var xmlHttp = null;
        if (window.XMLHttpRequest)          {
        // code for IE7+, Firefox, Chrome, Opera, Safari
                xmlHttp=new XMLHttpRequest();
                alert("คุณใช้ IE 7+ / Firefox/Chome /Opera/Safari");
        }
        else
        { // code for IE6, IE5
                xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
                alert("คุณใช้ IE เวอร์ชั่นต่ำกว่า 7");
        }
        else
        {
                alert("เว็บบราวเซอร์ของคุณไม่สนับสนุนการใช้งาน AJAX");
        }
}
</script>
</head>
<body onload="loadAJAX();">
</body>
</html>
```

เมธอดของ XMLHttpRequest

- มีเมธอดที่สามารถเรียกใช้ได้ มี 6 เมธอด

เมธอด	ความหมาย
abort()	ยกเลิกการร้องขอ
getAllResponseHeader()	รับค่าส่วนหัวที่ร้องขอทั้งหมดเป็นคู่ คือ คีย์และค่า (key/value pair)
getResponseHeader()	รับค่าส่วนหัวที่ร้องขอตามที่ระบุ
open(method , url , [isAsynchronous]	เปิดการติดต่อเว็บเซิร์ฟเวอร์ ซึ่ง -เมธอดที่ใช้อาจจะเป็น POST , GET หรือ PUT , -URL คือ page ปลายทาง -การเชื่อมต่อเป็น Asynchronous หรือไม่ ค่าปริยายเป็น true
send(body)	ส่งการร้องขอไปยังเซิร์ฟเวอร์
setRequestHeader(header , value)	การกำหนดประเภทของข้อมูลที่จะมีการส่ง/รับ

การส่ง request ไปยังเซิร์ฟเวอร์

- ในการส่ง request ไปยังเซิร์เวอร์ใช้เมธอด open() และ send() ของ XMLHttpRequest

```
xmlhttp.open("GET",“data.txt",true);
xmlhttp.send();
```

Method	Description
open(method , url , async)	เปิดการติดต่อกับ Server method เป็น GET หรือ POST url เป็นที่อยู่ของ page ที่จะเรียกเพื่อทำงานต่อ async กำหนดการทำงานเป็นแบบ Asynchronous หรือไม่
send(string)	ทำการส่งข้อมูล พารามิเตอร์ string ใช้สำหรับ post request

GET หรือ POST ?

- GET ส่งค่าได้เร็วกว่า POST

- จะใช้ POST เมื่อ ?
  - Update file หรือ database บนเซิร์ฟเวอร์
  - ส่งข้อมูลจำนวนมากไปยังเซิร์ฟเวอร์ (POST ไม่จำกัดขนาด)
  - ส่งข้อมูลที่อาจจะไม่ใช่ text
  - POST มีความปลอดภัยในการส่งข้อมูลมากกว่า GET

Property ของ XMLHttpRequest

คุณสมบัติ onreadystatechange

- ใช้กำหนดให้ function ที่กำหนด ทำหน้าที่ตรวจสอบสถานะการทำงานของการ request และสิ่งที่จะเกิดขึ้นหลังจากการ request (ทั้งกรณีที่ request สำเร็จ และไม่สำเร็จ)
- รูปแบบ onreadystatechange = function;
- มีค่า return type เป็น void

Property ของ XMLHttpRequest

คุณสมบัติ onreadystatechange

- ตัวอย่าง

```
xmlHttp.onreadyStatechange = functio() {
        //เขียนคำสั่งต่าง ๆ
}
```

Property ของ XMLHttpRequest

คุณสมบัติ readyState

- ใช้คืนค่าสถานะการทำงานปัจจุบันของการ request
- มีค่า return type เป็น ตัวเลข มีค่าคืนกลับ ดังนี้
  - ถ้าคืนค่า 0 คือยังไม่เริ่มทำงาน (ยังไม่ได้เริ่มเรียกใช้ open())
  - ถ้าคืนค่า 1 คือกำลังโอนถ่ายข้อมล (ประมวลผลใน open() อยู่)
  - ถ้าคืนค่า 2 คือโอนถ่ายข้อมูลเสร็จแล้ว (ประมวลผลใน send() อยู่)
  - ถ้าคืนค่า 3 คือกำลังทำงานอยู่ (server กำลัง response กลับมา)
  - ถ้าคืนค่า 4 คือการทำงานเสร็จสมบูรณ์

Property ของ XMLHttpRequest

คุณสมบัติ readyState

- ตัวอย่าง

```
xmlHttp.onreadyStatechange = functio() {
        if(xmlHttp.readyState == 4) {
                //รับค่าส่งกลับจากเซิร์ฟเวอร์
        }
}
```

Property ของ XMLHttpRequest

คุณสมบัติ status

- ใช้คืนค่าสถานะกรทำงานว่าการ request ถูกต้องหรือไม่
- มีค่า return type เป็น ตัวเลข
- มีค่าคืนกลับ ดังนี้
  - ถ้าคืนค่า 200 คือการ request ถูกต้องสมบูรณ์
  - ถ้าคืนค่า 404 คือการ request ไม่ถูกต้องไม่สมบูรณ์

Property ของ XMLHttpRequest

คุณสมบัติ status

- ตัวอย่าง

```
xmlHttp.onreadyStatechange = functio() {
        if(xmlHttp.readyState == 4) {
                if(xmlHttp.status == 200) {
                        //คำสั่งทำงาน
                }
        }
}
```

Property ของ XMLHttpRequest

คุณสมบัติ responseText

- ใช้คืนค่าข้อมูลที่ response มาจาก server แบบข้อความ ซึ่งเป็นได้ทั้ง ข้อความแบบธรรมดา, แบบ html หรือแบบ xml โดย page ปลายทางจะต้อง echo ค่าออกมาเพื่อเป็นการ response ค่ากลับ
- มีค่า return type เป็น ข้อความ

Property ของ XMLHttpRequest

คุณสมบัติ responseText

- ตัวอย่าง

```
xmlHttp.onreadyStatechange = functio() {
        if(xmlHttp.readyState == 4) {
                if(xmlHttp.status == 200) {
                        alert(xmlHttp.responseText) ;
                }
        }
}
```

Property ของ XMLHttpRequest

คุณสมบัติ responseXML

- ใช้คืนค่าข้อมูลที่ response มาจาก server แบบ xml โดย page ปลายทางจะต้อง echo ค่าออกมาเพื่อเป็นการ response ค่ากลับ
- เหมาะกับการส่งข้อมูลกลับแบบหลายค่า
- มีค่า return type เป็น ข้อความ

Property ของ XMLHttpRequest

คุณสมบัติ responseXML

- ตัวอย่าง

```
xmlHttp.onreadyStatechange = functio() {
        if(xmlHttp.readyState == 4) {
                if(xmlHttp.status == 200) {
                        alert(xmlHttp.responseXML) ;
                }
        }
}
```

ตัวอย่าง GET request

- ตัวอย่าง การใช้ GET แบบง่าย

```
xmlhttp.open("GET",“get_data.php",true);
xmlhttp.send();
```

- ส่งค่าไปพร้อมกับ url

```
xmlhttp.open("GET",“get_data.php?fname=rewadee&major=IT",true);
xmlhttp.send();
```

ตัวอย่าง POST request

- ตัวอย่าง การใช้ POST แบบง่าย

```
xmlhttp.open("POST",“post_data.php",true);
xmlhttp.send();
```

- ส่งข้อมูลคล้ายการส่งผ่านฟอร์มและเพิ่ม header ด้วยเมธอดsetRequestHeader และกำหนดข้อมูลในการส่งด้วยเมธอด send()

```
xmlhttp.open("POST","ajax_test.asp",true);
xmlhttp.setRequestHeader("Content-type","application/x-www-form-urlencoded");
xmlhttp.send("fname=Henry&lname=Ford");
```

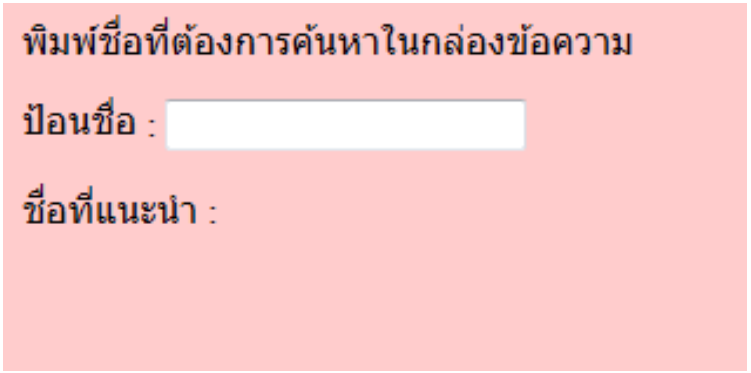
ตัวอย่าง POST request

- http Header ที่ต้องกำหนดเมื่อมีการใช้ POST request มีดังนี้

```
http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
http.setRequestHeader("Content-length", params.length);
http.setRequestHeader("Connection", "close");
```

Workshop2 ทดสอบการส่งค่าด้วย AJAX อย่างง่าย

1. สร้างไฟล์ html ชื่อ simple_hint.html
2. สร้างส่วน UI ดังนี้




```
<body>
พิมพ์ชื่อที่ต้องการค้นหาในกล่องข้อความ   <br />
<p>
<form action="">
ป้อนชื่อ   : <input type="text" name="txt_name" />
</form>
</p>
<p> ชื่อที่แนะนำ   : <div id="txtHint"> </div></p>
</body>
```

Workshop2 ทดสอบการส่งค่าด้วย AJAX อย่างง่าย

```
<Script type="text/javascript">
function show_hint(str)
{
		var xmlHttp ;
		if(str.length == 0)
		{
			document.getElementById("txtHint").innerHTML = "";
			return ;
		}

		if(window.XMLHttpRequest)
		{// code for IE7+, Firefox, Chrome, Opera, Safari
			xmlHttp=new XMLHttpRequest();
		}
		else
		{// code for IE6, IE5
			xmlHttp=new ActiveXObject("Microsoft.XMLHTTP");
		}

		xmlHttp.onreadystatechange = function()
		{
			if(xmlHttp.readyState == 4 && xmlHttp.status == 200)
			{
				document.getElementById("txtHint").innerHTML = xmlHttp.responseText;

			}
		}
		xmlHttp.open("GET" , "get_hint.php?q="+str , true);
		xmlHttp.send();
}
</script>
```

Workshop2 ทดสอบการส่งค่าด้วย AJAX อย่างง่าย

4. สร้างไฟล์ get_hint.php เขียนโค้ดดังนี้

```
<?php
// Fill up array with names
$a[]="Anna";
$a[]="Brittany";
$a[]="Cinderella";
$a[]="Diana";
$a[]="Eva";
$a[]="Fiona";
$a[]="Gunda";
$a[]="Hege";
$a[]="Inga";
$a[]="Johanna";
$a[]="Kitty";
$a[]="Linda";
$a[]="Nina";
$a[]="Ophelia";
$a[]="Petunia";
$a[]="Amanda";

//อ่านค่าพารามิเตอร์จาก  URL
$q = $_GET["q"];
```

Workshop2 ทดสอบการส่งค่าด้วย AJAX อย่างง่าย

4. สร้างไฟล์ get_hint.php เขียนโค้ดดังนี้

```
//ถ้าความยาวของชื่อที่ป้อนเข้ามามากกว่า  0  ให้ค้นหาชื่อจากอาร์เรย์
if(strlen($q) > 0)
{
        $hint = "";
        for($i = 0; $i < count($a) ; $i++)
        {
                if(strtolower($q) == strtolower(substr($a[$i], 0 , strlen($q))))
                {
                        if($hint == "" )
                        {       $hint = $a[$i];         }
                        else
                        {       $hint = $hint." , ".$a[$i];          }
                }
        }
}

//กำหนดค่าผลลัพธ์ที่ค้นหาเจอ  หรือค้นหาไม่เจอ
if($hint == "")
{       $response = "ไม่พบชื่อที่สอดคล้อง";  }
else
{       $response = $hint ;   }

echo $response;
//echo "Hello";
?>
```

การจัดการข้อมูลจากฟอร์มด้วย AJAX

ฟอร์มและอิลิเมนต์ของฟอร์ม

- ฟอร์มเป็นองค์ประกอบที่สำคัญอย่างหนึ่งของ HTML เพื่อใช้ในการรับส่งข้อมูลจากผู้ใช้เพื่อนำข้อมูลไปประมวลผลที่เซิร์ฟเวอร์
- ปกติการส่งข้อมูลจากฟอร์มออกไปโดยตรง อาจไม่จำเป็นต้องสนใจการเข้าถึงหรืออ่านข้อมูลจากฟอร์มก็ได้
- แต่ถ้าส่งข้อมูลจากฟอร์มผ่าน AJAX จำเป็นต้องทราบวิธีการอ้างถึงองค์ประกอบ หรืออิลิเมนต์ภายในฟอร์ม จึงจะสามารถอ่านข้อมูลไปใช้ได้

ฟอร์มและอิลิเมนต์ของฟอร์ม

ลักษณะของแท็ก form ทั่วไป

```
<form  แอททริบิวต์>
        อิลิเมนต์ของฟอร์ม
        ...
</form>
```

ลักษณะของแท็ก form ทั่วไป

```
<form  id=“frm” name = “frm”>
        อิลิเมนต์ของฟอร์ม
        ...
</form>
```

อิลิเมนต์ของฟอร์ม

อิลิเมนต์	การใช้งาน	ลักษณะแท็ก
text	รับข้อมูลตัวอักษรแบบบรรทัดเดียว	<input type="text">
password	รับข้อมูลที่เป็นรหัสผ่าน	<input type="password">
textarea	รับข้อมูลแบบตัวอักษรที่ยาว ๆ แบบหลายบรรทัด	<textarea></textarea>
radio	รายการตัวเลือกที่สามารถเลือกได้เพียงรายการเดียวในกลุ่ม	<input type="radio">
checkbox	รายการตัวเลือกที่สามารถเลือกได้หลายรายการ	<input type="checkbox">
select	สร้างตัวเลือกแบบดร็อปดาวน์ โดยมีให้เลือกทั้งรายการเดียว หรือหลายรายการ	<select></select>

อิลิเมนต์ของฟอร์ม

อิลิเมนต์	การใช้งาน	ลักษณะแท็ก
option	ใช้สร้างตัวเลือกสำหรับ select	<select> <option>item1</option> <option>item1</option> </select>
file	สำหรับรับไฟล์เพื่ออัปโหลดขึ้นไปบนเซิร์ฟเวอร์	<input type="file">
button	เป็นปุ่มกดเพื่อสั่งงาน	<button></button>
submit	ปุ่มกดเพื่อส่งข้อมูลจากฟอร์มไปยังเซิร์ฟเวอร์	<input type="submit">
reset	ปุ่มกดเพื่อ Clear ข้อมูลในฟอร์ม	<input type="reset">

อิลิเมนต์ของฟอร์ม

แอททริบิวต์	ใช้กับอิลิเมนต์	คำอธิบาย
name	ทุกอิลิเมนต์	กำหนดชื่อให้กับอิลิเมนต์นั้นๆ
id	ทุกอิลิเมนต์	ใช้ในการอ้างถึงอิลิเมนต์นั้น ๆ ตามรูปแบบ W3C DOM
value	ทุกอิลิเมนต์ ยกเว้น textarea และ select	สำหรับกำหนดค่าของอิลิเมนต์นั้น ซึ่งเป็นค่าที่เราจะนำไปใช้ประมวลผลต่อไป
size	text , password , file	ขนาดความยาวของอิลิเมนต์
maxlength	text , password , file	จำนวนอักขระสูงสุดที่สามารถใส่เข้าไปได้

อิลิเมนต์ของฟอร์ม

แอททริบิวต์	ใช้กับอิลิเมนต์	คำอธิบาย
cols	textarea	จำนวนแถว
rows	textarea	จำนวนคอลัมน์
multiple	select	เมื่อต้องการให้เลือกรายการใน select ได้พร้อมกัน มากกว่า 1 รายการ
selected	option	กำหนดให้ตัวเลือกนั้นถูกเลือกไว้ล่วงหน้า
checked	checkbox , radio	กำหนดให้อิลิเมนต์นั้นถูกเลือกไว้ล่วงหน้า
disabled	ทุกอิลิเมนต์	ทำให้ไม่สามารถฝช้งานอิลิเมนต์นั้นได้

การเข้าถึงอิลิเมนต์ของฟร์ม

- ทำได้ ดังนี้
  - DOM Level 0
  - W3C DOM

การเข้าถึงอิลิเมนต์แบบ DOM Level 0

- การอ้างถึงฟอร์ม

```
document.forms[เลขลำดับ]
document.forms[“ชื่อฟอร์ม”]
document.ชื่อฟอร์ม
```

- การอ้างถึงอิลิเมนต์

```
form_ref.ชื่ออิลิเมนต์
form_ref.elements[เลขลำดับ]
form_ref.elements[“ชื่ออิลิเมนต์”]
form_ref.elements.ชื่ออิลิเมนต์
```

การเข้าถึงอิลิเมนต์แบบ DOM Level 0

```
<form name="frm">
        <input type="text" name="txt">
</form>
```

ตัวอย่างการอ้างถึงอิลิเมนต์แบบต่าง ๆ

```
document.forms[0].txt ;
document.forms["frm"].txt ;
document.frm.txt;
document.forms["frm"].elements[0] ;
document.forms["frm"].elements["txt"] ;
document.forms["frm"].elements.txt ;
document.frm.elements["txt"] ;
document.frm.elements.txt ;
```

การเข้าถึงอิลิเมนต์แบบ W3C DOM

ใช้เมธอด getElementById() หรือ getElementByTagName()

```
<form name="frm">
        <input type="text" id="txt">
        <textarea  id="addr" cols="10" rows="4"></textarea>
</form>
```

ตัวอย่างการอ้างถึงอิลิเมนต์

```
document.getElementById("txt") ;
document.getElementById("addr") ;
document.getElementById("frm") ;
doucment.getElementById("frm").elements[0];
doucment.getElementById("frm").elements["txt"];
```

การอ่านข้อมูลจากอิลิเมนต์ประเภทข้อความ

```
<form name=“frm”>
        <input type=“text” id=“login”>
        <input type=“password” id=“pwd”>
        <textarea  id=“addr” cols=“10” rows=“4”></textarea>
</form>
```

การอ่านข้อมูลจากอิลิเมนต์ประเภทข้อความ

```
var     username        = document.getElementById(“login”).value ;
var      pwd            = document.getElementById(“pwd”).value;
var     addr            =document.getElementById(“addr”).value;
```

การกำหนดค่าข้อมูลให้อิลิเมนต์ประเภทข้อความ

```
document.getElementById(“addr”).value = “จ.อุดรธานี” ;
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด checkbox

```
<input type=“checkbox” name=“chk” id=“chk” value=“Ajax”>
```

การอ่านข้อมูลจากอิลิเมนต์จาก checkbox

```
var     chk     = document.getElementById(“chk”).value ;
```

** checkbox ต้องถูกเลือก (คุณสมบัติ checked) จึงจะอ่านข้อมูลได้

```
var     chk = document.getElementById(“chk”) ;
if(chk.checked) {
        value = chk.value ;
}
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด radio

```
<input type=“radio” name=“rdo”  value=“ASP”>
<input type=“radio” name=“rdo”  value=“PHP”>
<input type=“radio” name=“rdo”  value=“JSP”>
```

การอ่านข้อมูลจากอิลิเมนต์คล้ายกับ checkbox

```
var         value = “ ”;
var         num_rdo = myform.elements[‘rdo’].length ;
for(i=0;i<num_rdo ; i++)
{
            var rdo = myform.elements[‘rdo’][i];
            if(rdo.checked) {
                        value = rdo.value ;
                        break;
            }
}
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด select

length : ใช้ในการอ่านหรือนับจำนวนตัวเลือกทั้งหมดใน select

```
var     len     = document.getElementById("myselect").length ;
```

selectedIndex : ใช้ในการอ่านลำดับตัวเลือกที่ถูกเลือกในขณะนั้น
หรือ ใช้ในการกำหนดให้ตัวเลือกในลำดับใดถูกเลือก

```
var  idx = document.getElementById("myselect").selectedIndex ;

//กำหนดให้ตัวเลือกลำดับที่ 3 ถูกเลือก
document.getElementById("myselect").selectedIndex = 2;
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด select

Option[ลำดับ].value : ใช้ในการอ่านหรือกำหนดค่าให้แก่ตัวเลือกที่มีลำดับตรงกับที่ระบุ เช่น

```
<select id="myselect">
        <option value="1">PHP</option>
        <option value="2">ASP</option>
         <option value="3">JSP</option>
</select>
```

```
var     my_sl = document.getElementById("myselect") ;
var     idx = my_sl .selectedIndex ;
var     value = my_sl.options[idx].value ;
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด select

Option[ลำดับ].text : ใช้ในการอ่านข้อความที่ปรากฏในตัวเลือกนั้น

```
<select id="myselect">
        <option value="1">PHP</option>
        <option value="2">ASP</option>
         <option value="3">JSP</option>
</select>
```

```
var     my_sl = document.getElementById("myselect") ;
var     idx = my_sl .selectedIndex ;
var     value = my_sl.options[idx].text ;
```

การอ่านข้อมูลจากอิลิเมนต์ชนิด select

Option[ลำดับ].selected : ใช้ในการตรวจสอบว่าตัวเลือกในลำดับนั้นถูกเลือกหรือไม่ หากถูกเลือกจะคืนค่า true หรือกำหนดให้ตัวเลือกลำดับนั้นถูกเลือก(true) หรือไม่เลือก (false)

```
if(document.getElementById("my_sl").options[0].selected == false) {
        document.getElementById("my_sl").options[0].selected = true;
}
```

Serialize Data และ Encoding

- เนื่องจากการส่งข้อมูลด้วย AJAX นั้นไม่ได้ส่งผ่านฟอร์มโดยตรง แต่ต้องเข้าไปอ่านข้อมูลจากฟอร์มและส่งผ่าน AJAX เอง
- ดังนั้นการส่งด้วย AJAX ต้องนำข้อมูลทั้งหมดที่จะส่งมาจัดเรียงต่อกัน ในรูปแบบที่เรียกว่า Serialize Data
- รูปแบบ

```
parameter_1 = data&parameter_2=data&....&parameter_n=data
```

- เช่น

```
Fname=Rewadee&lname=Piputsoongnern
```

การเข้ารหัสข้อมูล

- การส่งข้อมูลผ่าน AJAX เราต้องเข้ารหัสเอง โดยใช้ฟังก์ชัน encodeURI()
- หลังจากเข้ารหัสก็นำไปต่อท้าย URL เพื่อส่งข้อมูลไปยังปลายทางได้ตามต้องการ
- หรือถ้าส่งด้วยเมธอด POST ก็นำไปกำหนดให้เมธอด send() เป็นต้น

Serialize Data และ Encoding

- 1. ข้อมูลแต่ละอย่างควรถูกเข้ารหัสแยกทีละตัวก่อนนำมารวมกัน

```
var title = encodeURI(myform.elements[‘title’].value) ;
var author = encodeURI(myform.elements[‘author’].value) ;
var press = encodeURI(myform.elements[‘press’].value);
```

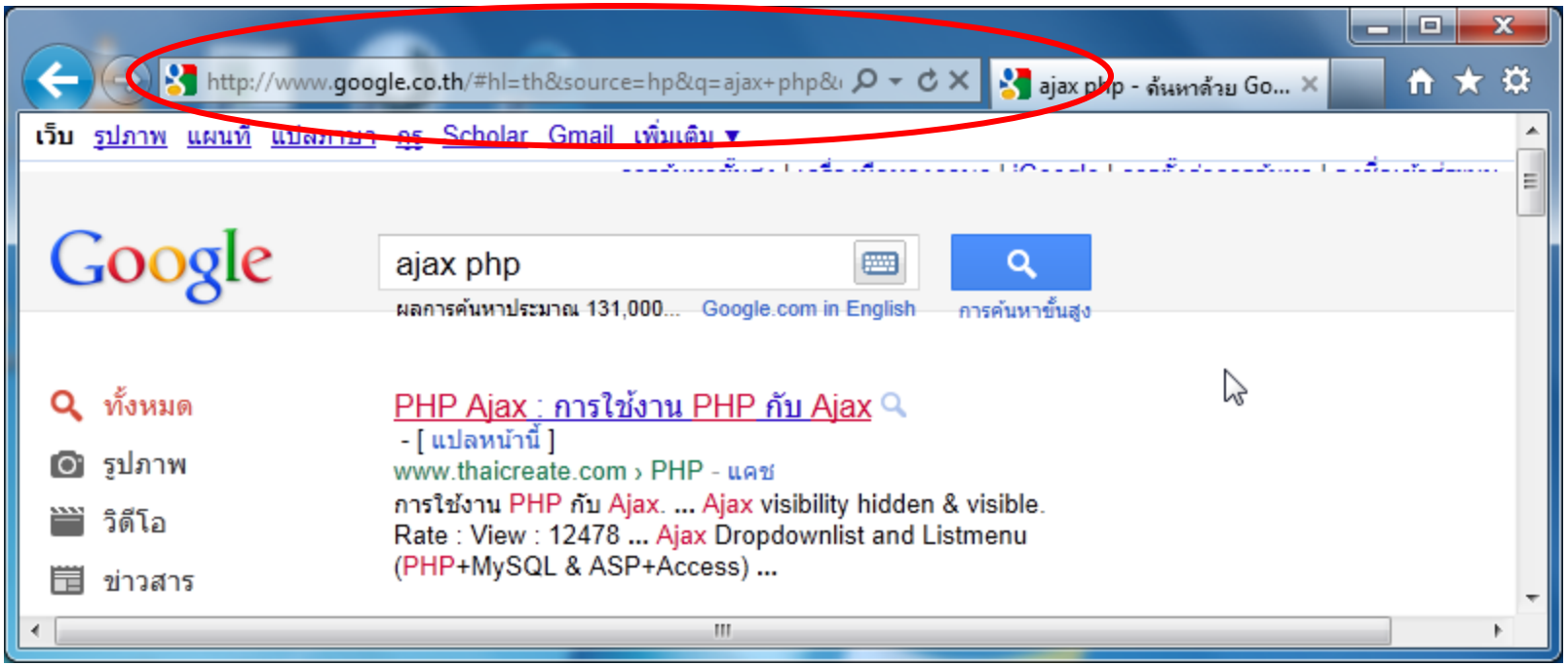
- 2. กำหนดข้อมูลแบบ Serialize ที่ต้องการ โดยใช้รูปแบบ
- ตัวแปร = ข้อมูล ถ้ามีหลายตัวแปร คั่นด้วยเครื่องหมาย &

```
var data = “title=”+ title + “&author=” + author + “&press=” + press ;
```

- 3. ส่งออกไปด้วย AJAX หากเลือก GET นำไปต่อท้าย URLตามด้วย “?” หากใช้เมธอด POST นำไปกำหนดในเมธอด send()

การส่งข้อมูลด้วยเมธอด GET

- โดยนำข้อมูลในรูปแบบ Serialize Data มาต่อท้าย URL



แนวทางการส่งข้อมูลด้วยเมธอด GET

- 1. นำข้อมูลที่จะส่งมาสร้างเป็น queryString(Serialize data) (ควรจะเข้ารหัสด้วยฟังก์ชัน encodeURI())

```
var fname = encodeURI(myform.elements['fname'].value) ;
var lname = encodeURI(myform.elements['lname'].value) ;
var data = "firstName=" + fname + "lastName=" + lname ;
```

2. นำ queryString มาต่อท้าย URL

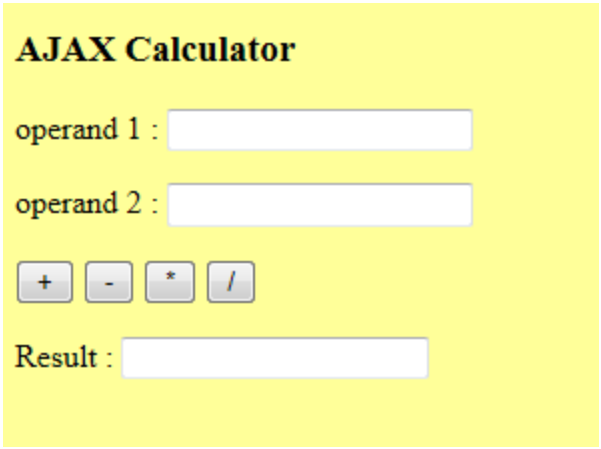
```
var url = "http://.../test.php?" + data ;
```

3. เปิดการเชื่อมต่อด้วยเมธอด open() กำหนดเมธอดเป็น "GET" และ url นำค่าจากข้อ 2 มากำหนด

```
open("GET" , url);
```

ตัวอย่างการส่งข้อมูลแบบ GET

- 1. สร้างไฟล์ HTML ชื่อ calculator.html
- 2. ฟอร์มดังนี้




```
<body>
<p><H3>AJAX Calculator</H3></p>
<p>operand 1 : <input name="op1" id="op1" type="text" /></p>
<p>operand 2 : <input name="op2" id="op2" type="text" /></p>
<p> <input name="" type="button" value="+" onclick = "calculate('plus')" />
    <input name="" type="button" value="-" onclick = "calculate('minus')" />
    <input name="" type="button" value="*" onclick = "calculate('multiply')" />
    <input name="" type="button" value="/" onclick = "calculate('divide')" />
</p>
<p>
Result : <input name="result" id="result" type="text" readonly = "true" />
</p>
</body>
```

ตัวอย่างการส่งข้อมูลแบบ GET

- 3. วางสคริปซ์ javascript ระหว่างแท็ก <head></head>
- 4. สร้างฟังก์ชัน loadAjax() ดังนี้

```
<script type="text/javascript">
function loadAjax(method , url , data , displayID)
{
    var ajax = null;
    if(window.XMLHttpRequest)
    {   ajax = new XMLHttpRequest() ; }
    else
    {   ajax = new ActiveXObject("Microsoft.XMLHTTP"); }

    ajax.open(method , url);
    ajax.onreadystatechange = function() {
        if(ajax.readyState == 4 && ajax.status == 200)
        {
            var el = document.getElementById(displayID) ;
            el.value = ajax.responseText ;
        }
    }
    ajax.send(data);
}
```

ตัวอย่างการส่งข้อมูลแบบ GET

- 5. สร้างฟังก์ชัน calculate() ดังนี้

```
function calculate(sign)
{
    var url = "calculator.php" ;
    url += "?op1=" + document.getElementById("op1").value ;
    url += "&op2=" + document.getElementById("op2").value ;
    url += "&sign=" + sign ;

    loadAjax('get' , url , null , 'result');

}
</script>
```

ตัวอย่างการส่งข้อมูลแบบ GET

- 6. สร้างไฟล์ calculator.php และเขียนโค้ดดังนี้

```php
<?php
$op1 = $_GET['op1'];
$op2 = $_GET['op2'];
$sign = $_GET['sign'];

switch($sign)
{
   case "plus" :      echo $op1 + $op2;
                      break ;
   case "minus" :     echo $op1 - $op2;
                      break;
   case "multiply" :  echo $op1 * $op2;
                      break;
   case "divide" :    echo $op1 / $op2 ;
                      break;
}
?>
```

การส่งข้อมูลด้วยเมธอด POST

- POST เป็นการส่งข้อมูลที่ไม่จำกัดขนาดของข้อมูล
- มีรูปแบบการส่งที่เพิ่มเติมจากเมธอด GET ดังนี้
- 1. เปิดการเชื่อมต่อด้วยเมธอด open(“post” , url);
- 2. ต้องเข้ารหัส HTTP Header ของข้อมูลที่จะส่งไปด้วยเมธอด setRequestHeader() ดังนี้

```
Ajax.open(“post” , url);
Ajax.setRequestHeader(“Content-Type” , “application/x-www-form-urlencodeed”);
```

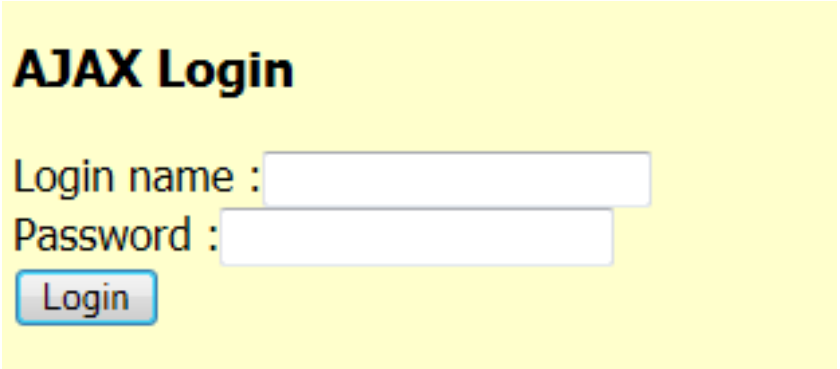
การส่งข้อมูลด้วยเมธอด POST

- 3. ข้อมูลที่จะส่งต้องอยู่ในรูปแบบ Serialize Data และส่งไปในเมธอด send() เช่น

```
Ajax.send(“fname=Rewadee&lname=Piputsoongnern”);
```

ตัวอย่างการส่งข้อมูลแบบ POST

- 1. สร้างไฟล์ HTML ชื่อ login.html
- 2. สร้างหน้าเว็บดังนี้




```
<body>
<p><h3>AJAX Login</h3></p>
<p>
   Login name :<input type="text" name="login" id="login" /> <br />
   Password    :<input type="password" name="pwd" id="pwd" /><br />
  <button onclick="login()">Login</button>
</p>
<p><div id="msg"></div></p>
</body>
```

ตัวอย่างการส่งข้อมูลแบบ POST

- 3. วางสคริปซ์ javascript ระหว่างแท็ก <head></head>
- 4. สร้างฟังก์ชัน loadAjax() ดังนี้

```
<script type="text/javascript">
function loadAjax(method , url , data , displayID)
{
   var ajax = null;
   if(window.XMLHttpRequest)
   {  ajax = new XMLHttpRequest() ; }
   else
   {  ajax = new ActiveXObject("Microsoft.XMLHTTP"); }

   ajax.open(method , url);
   ajax.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
   ajax.onreadystatechange = function() {
      if(ajax.readyState == 4 && ajax.status == 200)
      {
         var el = document.getElementById(displayID) ;
         el.innerHTML = ajax.responseText ;
      }
   }
   ajax.send(data);
}
```

ตัวอย่างการส่งข้อมูลแบบ POST

- 5. สร้างฟังก์ชัน login() ดังนี้

```
function login()
{
   var url = "chk_login.php" ;

   var  login = document.getElementById("login").value ;
   var  pwd = document.getElementById("pwd").value ;
   var  data = "login=" + login + "&pwd=" + pwd ;

   loadAjax('post' , url , data , 'msg');
}
```