บทที่ 12
การใช้เทคนิค AJAX ร่วมกับ PHP

ปัจจุบันนี้ ลักษณะการทำงานแบบ ไคลเอนต์ - เซิร์ฟเวอร์ เริ่มถูกนำมาใช้งานอย่างแพร่หลายในลักษณะการติดต่อสื่อสารผ่านทางเว็บบราวเซอร์ การทำงานแบบนี้ จะมีการทำงานโดย ไคลเอนต์ จะร้องขอและต้องการข้อมูลบางอย่างจาก เซิร์ฟเวอร์ ดังนั้นการโหลดและการรีเฟรชหน้าจอ เป็นสิ่งที่หลีกเลี่ยงไม่ได้ จึงเป็นผลให้การทำงานของฝั่ง ไคลเอนต์ นี้ทำให้ผู้ใช้ต้องหยุดรอการโหลดและการรีเฟรชหน้าจอถือว่าเป็นการทำงานที่ไม่มีประสิทธิภาพ

12.1 AJAX

AJAX (Asynchronous JavaScript and XML) ไม่ใช่ชื่อของการเขียนโปรแกรมหรือเป็นชื่อของภาษาที่ใช้ในการโปรแกรม แต่เป็นชุดของเทคโนโลยีต่างๆ AJAX ย่อมาจาก Asynchronous JavaScript And XML หมายถึง การทำงานร่วมกันของ JavaScript และ XML แบบ Asynchronous มีหลักการทำงาน 2 ประเด็น คือ การอัพเดท (update) หน้าจอแบบบางส่วน และการติดต่อสื่อสารกับ เซิร์ฟเวอร์ โดยใช้หลักการ Asynchronous ทำให้ผู้ใช้ไม่ต้องหยุดการทำงาน เพื่อรอการประมวลผลจาก เซิร์ฟเวอร์ รวมถึงการโหลดและการรีเฟรชหน้าจอ ของบราวเซอร์ในฝั่ง ไคลเอนต์ มีการใช้ AJAX โดยการเพิ่มเลเยอร์ระหว่างเว็บบราวเซอร์ของผู้ใช้กับ เซิร์ฟเวอร์ ทำให้ผู้ใช้สามารถทำงานได้โดยไม่ต้องรอให้ ไคลเอนต์ติดต่อไปยังเซิร์ฟเวอร์ รวมถึงการโหลดและการรีเฟรชหน้าจอทั้งหมดด้วย ดังนั้นผู้ใช้สามารถใช้งาน แอพพลิเคชันได้อย่างมีประสิทธิภาพมากขึ้น

AJAX จึงไม่ใช่เทคโนโลยีในตัวของมันเอง แต่ว่าเป็นการนำเทคโนโลยีหลายๆ ตัวมารวมกันเช่น JavaScript, DHTML, XML, CSS, DOM และ XMLHTTPRequest

AJAX Engine ทำหน้าที่เป็นตัวกลางระหว่าง ไคลเอนต์และเซิร์ฟเวอร์ ฉะนั้นเมื่อ ไคลเอนต์ มีการร้องขอแทนที่จะส่ง HTTP request ไปยังเซิร์ฟเวอร์โดยตรง ไคลเอนต์จะส่ง JavaScript call ไปยัง AJAX Engine เพื่อโหลดข้อมูลที่ผู้ใช้ต้องการ และหาก AJAX Engine ต้องการข้อมูลเพิ่มเติมในการตอบสนองต่อ AJAX Engine ของผู้ใช้จะส่งการร้องขอไปยังเซิร์ฟเวอร์ โดยใช้ XML เทคโนโลยีต่างๆ ที่เป็นส่วนประกอบของ AJAX ดังนี้

12.1.1 HTML/XHTML เป็นภาษาในการจัดแสดงข้อมูล

12.1.2 CSS เป็นรูปแบบการจัดแต่ง XHTML

12.1.3 Document Object Model (DOM) สำหรับ dynamic display and interaction

12.1.4 XML เป็นรูปแบบการแลกเปลี่ยนข้อมูล

12.1.5 XSLT สำหรับ แปลง XML เป็น XHTML

12.1.6 XMLHTTPRequest สำหรับ asynchronous data retrieval

12.1.7 JavaScript เป็นภาษาในการใช้งาน AJAX Engine

12.2 ประวัติความเป็นมา

ในช่วงแรกๆ ของการพัฒนา คือ ประมาณปี ค.ศ. 1997 นั้น องค์ประกอบแรกๆ ที่เกิดขึ้นทางฝั่งไคลเอนต์ ถูกเขียนขึ้นโดยทีมพัฒนา Outlook Web Access ต่อมาถูกนำมาใช้เป็นส่วนหนึ่งของ Internet Explorer 5.0 นั่นก็คือจุดเริ่มต้นที่เริ่มรู้จักการทำงานแบบ AJAX และในปี 2005 Google ได้ใช้การติดต่อสื่อสารแบบ Asynchronous เพื่อเป็นรากฐานที่ทำให้รู้จักกับ AJAX กันอย่างแพร่หลาย การทำงานแบบ ไคลเอนต์ - เซิร์ฟเวอร์ ถูกนำมาใช้งานเป็นจำนวนมาก เช่น การติดต่อกับฐานข้อมูลที่เซิร์ฟเวอร์ หรือการให้บริการทางอินเทอร์เน็ต โดย Google เป็นผู้ลงทุนลงแรงอย่างหนัก ในพัฒนาและการทดสอบ AJAX จึงสังเกตได้ว่า ผลผลิตใหญ่ของ Google คริส อัลแมน และลูซินดา ดิเคส (Chris Ullman and Lucinda Dykes, 2007, p.3) ได้กล่าวไว้ว่า ในช่วงต้นปี 2005 ได้มีการนำการนำ AJAX มาประยุกต์ใช้งานกับเว็บไซต์ต่างๆ ทั่วโลก เช่น Flickr (www.flickr.com), Basecamp (basecamp.com), Amazon CloudSearch (www.A9.com), Google Suggest and Google Maps และอีกหลายเว็บไซต์ทั่วโลก

12.3 ที่มาของปัญหา

เนื่องจากแอพลิเคชันที่ใช้งานในปัจจุบันนี้ มีหลักการที่ทำงานแล้วเกิดการสูญเสียเวลาและทรัพยากรของผู้ใช้ในการรอคอยการทำงานต่างๆ ทำให้ผู้ใช้ต้องหยุดคอย ดังนั้นการทำงานของผู้ใช้จึงเป็นไปอย่างไม่ต่อเนื่อง หลักการดังกล่าวคือ

12.3.1 Click, wait, and refresh" user interaction paradigm

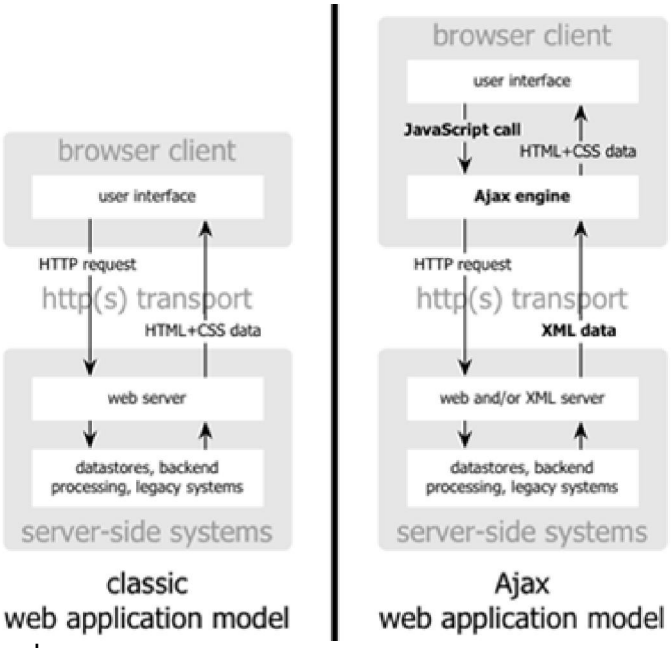
การที่บราวเซอร์ตอบสนองต่อการทำงานของผู้ใช้ โดยจะทิ้งหน้าเว็บที่แสดงอยู่ในขณะนั้น แล้วไปทำการส่ง HTTP request กลับไปยังเซิร์ฟเวอร์แทน ทำให้ผู้ใช้ไม่สามารถทำอะไรได้เลยในขณะนั้น นอกจากการรอคอย เมื่อเซิร์ฟเวอร์ทำการประมวลเสร็จก็จะส่งหน้า HTML กลับมายังบราวเซอร์ ต่อจากนั้นบราวเซอร์ก็จะรีเฟรชและแสดงหน้า HTML หน้าใหม่ และนี้เองที่ทำให้ผู้ใช้สามารถใช้งานต่อไปได้

จะเห็นว่าผู้ใช้มีช่วงเวลาของการหยุดรอคอยเป็นเวลานานสำหรับการประมวลผลของเซิร์ฟเวอร์ และการรีเฟรชหน้า HTML ใหม่ทั้งหน้า ซึ่งเป็นสิ่งที่ไม่มีประสิทธิภาพในเชิงไดนามิกของการทำงานบนเว็บแอพลิเคชัน

12.3.2 Synchronous "request/response" communication mode

การที่บราวเซอร์เริ่มทำการร้องขอข้อมูลและเซิร์ฟเวอร์ ก็ตอบสนองเฉพาะการร้องขอที่บราวเซอร์ร้องขอมาเซิร์ฟเวอร์ จะไม่สามารถส่งข้อมูลได้ถ้าบราวเซอร์ไม่ได้ร้องขอข้อมูลในขณะนั้น ถือได้ว่าเป็นการติดต่อสื่อสารเป็นแบบทิศทางเดียว

วงจรการ Request/Response แบบ Synchronous คือ การทำงานแบบประสานจังหวะระหว่างบราวเซอร์กับเซิร์ฟเวอร์ ทำให้เกิดความล่าช้าในการทำงานทำให้ผู้ใช้ทำอะไรไม่ได้อีกนอกจากการคอยการตอบสนองกลับมาจากเซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ประมวลผลเสร็จ

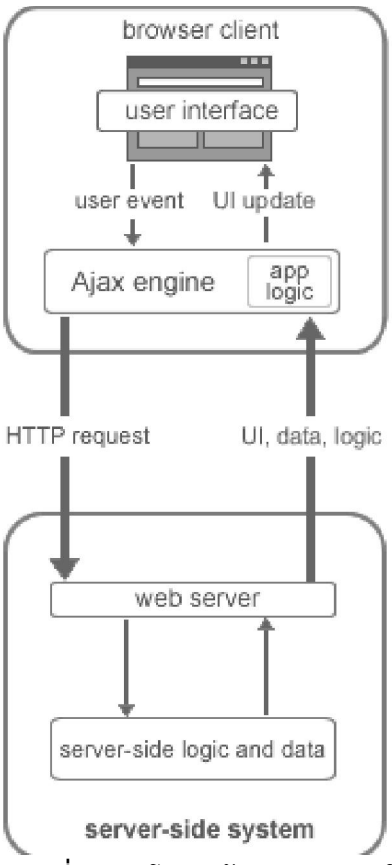


ภาพที่ 12.1 เปรียบเทียบการทำงานแบบเดิม กับ AJAX

ที่มา: เนคเทค. (2556).

12.4 โครงสร้างของ AJAX

มุมมองของโครงสร้างของ AJAX ต่างจากเว็บแอพลิเคชันในทุกวันนี้ เนื่องจากมีการเพิ่ม Engine ทางฝั่งไคลเอนต์



ภาพที่ 12.2 โครงสร้างของ AJAX ในปัจจุบัน

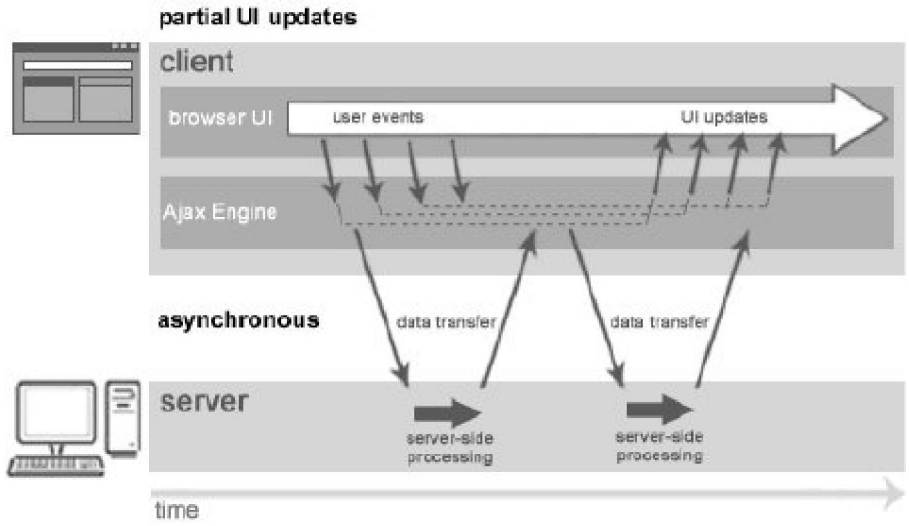
ที่มา: เนคเทค. (2556).

จากภาพ AJAX Engine นี้ อยู่ระหว่างผู้ใช้งานกับเซิร์ฟเวอร์ จะมองว่าเป็นการทำงานที่ ไคลเอนต์การทำงานต่างๆ ของผู้ใช้ โปรแกรมจะไปเรียก AJAX Engine ตัวนี้ขึ้นมา แทนที่การร้องขอหน้าเว็บจากเซิร์ฟเวอร์โดยตรง และจะใช้โครงสร้างข้อมูลแบบ XML ในการขนย้ายข้อมูลระหว่างเซิร์ฟเวอร์ กับ AJAX Engine เมื่อบราวเซอร์ทำการร้องขอข้อมูลจากเซิร์ฟเวอร์

นอกจากนี้ AJAX Engine ไม่ต้องทำการติดตั้งส่วนเสริมการทำงานต่างๆ และไม่สามารถดาวน์โหลดได้ เพราะ AJAX เป็นแนวคิดในการแก้ปัญหาการหยุดชะงักการทำงานของผู้ใช้

12.5 การทำงานของ AJAX

AJAX จะช่วยลดการติดต่อระหว่างไคลเอนต์กับเซิร์ฟเวอร์ โดยในการโหลดหน้าเว็บเพจนั้น บราวเซอร์จะโหลดข้อมูลจาก AJAX Engine แทนการร้องขอข้อมูลจากเซิร์ฟเวอร์ โดยตรง ดังนั้น AJAX จะทำหน้าที่ทั้งการแสดงผลรวมทั้งติดต่อกับผู้ใช้ และติดต่อไปยัง เซิร์ฟเวอร์ แล้ว AJAX Engine อนุญาตให้การกระทำต่างๆ ในเว็บแอพพลิเคชันเป็นแบบ Asynchronous คือ ความเป็นอิสระในการติดต่อไปยังเซิร์ฟเวอร์นั่นเอง ดังนั้นผู้ใช้จะไม่พบกับบราวเซอร์หน้าขาวๆ อีกต่อไป และไม่ต้องรอการโหลดข้อมูลต่างๆ จากเซิร์ฟเวอร์



ภาพที่ 12.3 หลักการทำงานของ AJAX

ที่มา: เนคเทค. (2556).

12.5.1 Asynchronous

ลักษณะของ Asynchronous ในที่นี้คือ ไม่ต้องหยุดรอข้อมูล หมายความว่า หลังจากส่งข้อมูลออกไปแล้วก็ยังสามารถใช้งานส่วนอื่นๆ ในเว็บเพจนั้นได้ตามปกติ เพราะกระบวนทั้งหมดเป็นแบบ behind the scene

12.5.2 JavaScript

เนื่องจาก AJAX คือ การเอาเทคโนโลยีอื่นๆ มาใช้ร่วมกัน การที่จะใช้เทคโนโลยีต่างๆ เหล่านั้นได้ จำเป็นต้องใช้ภาษาอย่าง JavaScript มาเป็นตัวจัดการให้ และสำหรับ AJAX ต้องใช้ JavaScript เป็นตัวควบคุมกระบวนการทั้งหมด นับตั้งแต่เริ่มต้นจนสิ้นสุดการติดต่อสื่อสาร ดังนั้นผู้จะศึกษาเทคนิค AJAX นี้ได้จึงจำเป็นต้องมีพื้นฐานด้านภาษา JavaScript และ DHTML ที่ดีพอสมควร ยิ่งรู้เกี่ยวกับ DHTML/ JavaScript มากเท่าใด ก็ยิ่งมีแนวทางในการประยุกต์ใช้ AJAX มากเท่านั้น

12.5.3 XML

XML (Extensible Markup Language) เป็นภาษาที่ใช้สำหรับการเขียนเอกสาร markup (markup document) โดยที่เอกสาร markup นั้นมีการใช้ metadata หรือ tags เพื่อบอกหน้าที่และประเภทของข้อมูลของส่วนต่างๆ ของเอกสารนั้นได้โดยชัดเจน การเพิ่ม metadata หรือ tags เข้าไปในเอกสารสามารถทำให้โครงสร้างของเอกสารชัดเจนขึ้น และทำให้การประมวลผลเอกสารเป็นไปโดยง่ายและไม่จำเป็นที่จะต้องอาศัยมนุษย์เพื่อตีความเอกสาร

การใช้เทคโนโลยี XML ในการพัฒนามาตรฐานเพื่อการกระจายข่าวเนื่องจาก XML เป็นภาษาที่เหมาะกับการแลกเปลี่ยนข้อมูลผ่านเครือข่ายคอมพิวเตอร์ เนื่องจาก XML ไม่ได้ขึ้นอยู่กับโปรแกรมประยุกต์หรือระบบปฏิบัติการใด นอกจากนี้ XML ยังเป็นภาษาที่มีความยืดหยุ่น เนื่องจากผู้ใช้สามารถที่จะกำหนดและตั้งค่า metadata หรือ tags ให้เหมาะกับเอกสารเฉพาะที่ต้องการได้อย่างอิสระ

และยังสามารถเพิ่มเติม metadata หรือ tags ได้ในภายหลังโดยไม่มีผลกระทบต่อโปรแกรมที่มีอยู่ก่อนหน้า

XML สำหรับการพัฒนาร่วมกับ AJAX Application แล้วนั้น อาจไม่จำเป็นต้องรู้ XML เลยก็ได้ เพราะ AJAX มีเมธอดที่ใช้แปลงข้อมูลให้อยู่ในรูปแบบสตริงธรรมดาแบบง่ายๆ ได้

12.5.4 Server Side Script กับ AJAX

ตามหลักการของ AJAX จำเป็นต้องส่งข้อมูลบางอย่างไปยังเซิร์ฟเวอร์ ดังนั้นในฝั่งเซิร์ฟเวอร์ก็จำเป็นต้องเขียนสคริปต์ไว้สำหรับรับข้อมูลเข้ามาประมวลผล และส่งผลลัพธ์กลับไปให้ สามารถเลือกใช้ Engine ตัวใดก็ได้ที่ถนัด เช่น PHP, ASP, NET, JSP เป็นต้น ทั้งนี้การเขียนสคริปต์ทางฝั่งเซิร์ฟเวอร์นั้น ก็เหมือนกับการเขียนสำหรับเว็บเพจทั่วไปตามปกติ ไม่มีวิธีการหรือองค์ประกอบใดๆ เป็นพิเศษสำหรับ AJAX

12.6 การใช้ AJAX Framework

ตามขั้นตอนที่แท้จริงของเทคนิค AJAX นั้น ต้องเริ่มจากการใช้ JavaScript สร้างแบบเจ็กต์ XMLHTTP (สำหรับ IE 5-6) หรือไม่ก็ XMLHttpRequest (สำหรับ IE 7 ขึ้นไป และบราวเซอร์อื่นๆ) ขึ้นมาก่อน แล้วขั้นตอนต่าง ๆ ของ AJAX ก็ทำผ่านเมธอด และพร็อปเพอร์ตี้ของออบเจ็กต์นี้ แต่เนื่องจากลำดับขั้นตอนมีค่อนข้างมาก เพื่อลดความยุ่งยากดังกล่าวนี้ ในที่นี้จะนำ AJAX Framework มาใช้แทน ลักษณะของเฟรมเวิร์กนี้ก็คือจะแยกส่วนที่เป็น JavaScript ที่ต้องทำซ้ำๆ เหมือนกันทุกครั้งในรับส่งข้อมูลไปไว้ในไฟล์ต่างหาก แล้วสามารถนำเข้ามาใช้งานได้ในทุกเว็บเพจที่ต้องการ (คล้ายกับการ include file ใน PHP นั่นเอง) ช่วยลดขั้นตอนการเขียน JavaScript ลงไปได้มากทีเดียว ข้อกำหนดเบื้องต้นที่ต้องรู้ที่เอาไว้ในเบื้องต้นของการใช้ AJAX Framework มีดังนี้

12.6.1 เมธอดที่ใช้ในการส่งข้อมูล

ใน AJAX ต้องกำหนดเมธอดในการส่งข้อมูลเช่นเดียวกับฟอร์ม โดยเมธอดที่กำหนดได้ก็คือ POST หรือ GET หากเป็นข้อมูลสั้นๆ ไม่เน้นความปลอดภัยนักก็ใช้เมธอด GET หากเป็นข้อมูลยาวและต้องการความปลอดภัยก็ใช้แบบ POST

12.6.2 เว็บเพจปลายทางด้านเซิร์ฟเวอร์

เว็บเพจปลายทาง คือ เว็บเพจที่จะเป็นผู้รับข้อมูลมาประมวลผล เหมือนกับการกำหนดแอตตริบิวต์ "action" ของฟอร์ม ทั้งนี้เว็บเพจปลายทางควรแยกไปสร้างไว้อีกเว็บเพจหนึ่งต่างหากมากกว่าที่จะใช้วิธี Postback (การทำงานทั้งหมดอยู่ในหน้าเดียวกัน)

12.6.3 ข้อมูลที่จะส่งออกไป

ถ้าหากเขียนโปรแกรมตามขั้นตอนปกติ จะต้องนำข้อมูลจากฟอร์มมาเรียงต่อกันในลักษณะของ Query String เช่น

```
Keyword=AJAX&page=2&rowperpage=20
```

แต่เพื่อลดความยุ่งยากตรงจุดนี้ ผู้เขียนจึงได้สร้างฟังก์ชันไว้ใน AJAX Framework สำหรับการอ่านข้อมูลจากฟอร์มเพื่อมาเรียงต่อกันแบบ Query String ให้เรียกใช้งานแบบง่ายๆ แต่หากข้อมูลนั้นไม่ได้อ่านจากฟอร์มต้องกำหนดวิธีการจัดเรียงเอง แต่ก็ไม่มีอะไรยุ่งยาก สามารถนำความรู้เรื่องการสร้าง Query String ใน PHP มีลักษณะเช่นเดียวกันมาใช้ได้ทันที

12.6.4 การแสดงผลลัพธ์

หลังส่งข้อมูลขึ้นไปประมวลผลที่เซิร์ฟเวอร์แล้ว เมื่อผลลัพธ์ถูกส่งกลับมา หากเป็นข้อมูลสตริงธรรมดา (หรืออาจมีแท็ก HTML รวมอยู่ด้วย) ต้องกำหนดว่าจะนำไปแสดงที่อิลิเมนต์ (Element หรือส่วนนำเข้าข้อมูลจากฟอร์ม คือ ตัวเดียวกัน) ใดภายในเว็บเพจนั้น นั่นแสดงว่าต้องสร้างอิลิเมนต์ที่ใช้แสดงผลเอาไว้ล่วงหน้าแล้ว ก่อนที่จะเริ่มการทำงานของ AJAX โดยทั่วไปมักใช้ div หรือไม่ก็ span พร้อมกำหนดค่า id ให้กับมันเพื่อนำมาใช้อ้างอิงได้

12.6.5 การสร้างออบเจ็กต์ XMLHTTP/XMLHttpRequest

บัญชา ปะสีละเตสัง (2551, หน้า 174) ได้กล่าวไว้ว่า คลาส XMLHTTP และ XMLHttpRequest คือ คลาสที่ใช้จัดการข้อมูลของ AJAX ทั้งหมด ตั้งแต่การส่งข้อมูลไปจนถึงการรับข้อมูลผลลัพธ์กลับมา แต่อย่างไรก็ตาม เนื่องบราวเซอร์แต่ละตัวนั้นรองรับการใช้คลาสนี้แตกต่างกันออกไป ถ้าเป็น IE จะใช้คลาสที่มีชื่อว่า XMLHTTP แต่ถ้าเป็น Netscape/Firefox/Mozilla จะใช้คลาสที่ชื่อว่า XMLHttpRequest แต่สมาชิกภายในคลาส คือ เมธอด และพร็อปเพอร์ตี้สำคัญที่จะใช้งานหลักๆ จะคล้ายกัน แต่เนื่องจากทั้ง XMLHTTP/XMLHttpRequest อยู่ในรูปแบบของคลาสไม่ใช่ออบเจ็กต์ จึงยังไม่สามารถนำมาใช้งานโดยตรงได้ ต้องสร้างเป็นออบเจ็กต์เสียก่อน โดยแยกพิจารณาตามชนิดของบราวเซอร์ดังนี้

ตัวอย่างที่ 12.1 การออบเจ็กต์ XMLHTTP สำหรับ IE

```
<script>
        var ajax = null;
        if (window.ActiveXObject) {
                ajax = new ActiveXObject ("Microsoft.XMLHTTP");
        }
        .       // เขียนสคริปต์คำสั่งตามความต้องการของผู้ใช้
        .
</script>
```

จากตัวอย่างที่ 12.1 แสดงการออบเจ็กต์ XMLHTTP สำหรับ IE อธิบายได้ดังนี้

1) สร้างตัวแปรอินสแตนซ์สำหรับออบเจ็กต์ โดยกำหนดตัวแปรชื่อ ajax

2) ตรวจสอบว่าบราวเซอร์รองรับการใช้ ActiveX หรือไม่ ถ้ารองรับแสดงว่าเป็น IE เพราะเทคโนโลยี ActiveX นั้นจะมีเฉพาะในไมโครซอฟต์ แต่ถ้าไม่รองรับแสดงว่าเป็นบราวเซอร์ชนิดอื่นๆ

3) สร้างออบเจ็กต์สำหรับ AJAX เพื่อนำไปใช้งานต่อไป

ตัวอย่างที่ 12.2 การออบเจ็กต์ XMLHttpRequest สำหรับบราวเซอร์ที่ไม่ใช่ IE

```
<script>
        var ajax = null;
        if (window.XMLHttpRequest) {
                ajax = new XMLHttpRequest;
        }
        .       // เขียนสคริปต์คำสั่งตามความต้องการของผู้ใช้
        .
</script>
```

จากตัวอย่างที่ 12.2 การออบเจ็กต์ XMLHttpRequest สำหรับบราวเซอร์ที่ไม่ใช่ IE อธิบายได้ดังนี้

1) สร้างตัวแปรอินสแตนซ์สำหรับออบเจ็กต์ โดยใช้ตัวแปรชื่อ ajax แบบ XMLHttpRequest

2) ตรวจสอบว่าบราวเซอร์รองรับการใช้ XMLHttpRequest หรือไม่ ถ้ารองรับจึงจะสร้างออบเจ็กต์นี้ได้

3) สร้างออบเจ็กต์สำหรับ AJAX เพื่อนำไปใช้งานต่อไป

แม้ว่าผู้ใช้โดยส่สวนใหญ่จะใช้ IE เป็นหลัก และมีเพียงส่วนน้อยที่ใช้บราวเซอร์อื่นๆ แต่การสร้างเว็บไซต์ที่ดีควรสามารถใช้ได้ในหลายๆ บราวเซอร์ จะได้ไม่เกิดปัญหา จากการสร้างออบเจ็กต์สำหรับ AJAX ที่ผ่านมาจะเห็นว่าสามารถตรวจสอบบราวเซอร์ของผู้ใช้งานได้ ดังนั้นการประยุกต์ใช้ในขั้นตอนนี้ คือ การตรวจสอบบราวเซอร์ก่อนการใช้งานดังนี้

ตัวอย่างที่ 12.3 การออบเจ็กต์ของ AJAX ให้รองรับในทุกบราวเซอร์

```
<script>
        var ajax = null;
        if (window.ActiveXObject) {
                ajax = new ActiveXObject ("Microsoft.XMLHTTP");
        } else if (window.XMLHttpRequest) {
                ajax = new XMLHttpRequest;
```

```
        } else {
                alert ("บราวเซอร์ของคุณไม่รองรับการใช้งาน AJAX");
        }
        .       // เขียนสคริปต์คำสั่งตามความต้องการของผู้ใช้
        .
</script>
```

12.7 ฟังก์ชันที่ใช้จัดการข้อมูล

ภายใน AJAX Framework ผู้เขียนได้สร้างฟังก์ชันเพิ่มเติมขึ้นมา 2 ฟังก์ชัน เพื่อใช้ในการรับส่งข้อมูล และอ่านข้อมูลจากฟอร์มดังนี้

12.7.1 ฟังก์ชัน AJAXLoad ( )

ใช้ในการส่งข้อมูลจากบราวเซอร์ไปยังเซิร์ฟเวอร์ และรับผลลัพธ์กลับมาแสดงผลที่อิลิเมนต์ที่ กำหนด มีรูปแบบดังนี้

รูปแบบ

```
AJAXLoad (เมธอดเว็บเพจปลายทาง, ข้อมูลที่จะส่งออกไป, อิลิเมนต์แสดงผล)
```

ตัวอย่างที่ 12.4 ฟังก์ชัน AJAXLoad ( )

```
AJAXLoad ("post,test.php","kw=AJAX&page=10","mydiv");
```

12.7.2 ฟังก์ชัน getFormData( )

ใช้ในการอ่านข้อมูลจากฟอร์มมาต่อกันแบบ Query String โดยต้องระบุ id หรือชื่อแอตตริบิวต์ของฟอร์มที่ต้องการอ่านข้อมูล

รูปแบบ

```
getFormData (form_id หรือ form_name)
```

ตัวอย่างที่ 12.5 ฟังก์ชัน getFormData( )

```
var data = getFromData ("from1");
```

12.8 แนวทางการพัฒนา AJAX Application

กระบวนการของ AJAX สามารถแยกกระบวนการทำงานเป็น 2 ส่วน คือ 1) กระบวนการฝั่งบราวเซอร์ และ 2) กระบวนฝั่งเซิร์ฟเวอร์ มีรายละเอียดดังนี้

12.8.1 การบวนการฝั่งบราวเซอร์

1) ถ้าจะส่งข้อมูลจากฟอร์มออกไป ก็ให้นำฟอร์มและส่วนนำเข้าข้อมูลมาวางตามปกติ ทั้งนี้ต้องกำหนดค่า id หรือ name ให้กับฟอร์มด้วย เพื่อใช้อ้างถึงในการอ่านข้อมูล

2) จัดวางอิลิเมนต์ที่จะใช้แสดงผลลงไป ณ ตำแหน่งที่ต้องการ อาจใช้ Div หรือ Span และจะต้องกำหนดค่า id ให้กับอิลิเมนต์ด้วย ตัวอย่างดังนี้

```
<div id="diplayAJAX"> </div>
```

3) สร้างฟังก์ชัน JavaScript สำหรับจัดการ AJAX โดยภายในฟังก์ชันเป็นการจัดเตรียมข้อมูลสำหรับฟังก์ชัน AJAXLoad ( ) ตามที่ได้กล่าวถึงมาแล้ว ตัวอย่างดังนี้

ตัวอย่างที่ 12.6 การสร้างฟังก์ชัน JavaScript สำหรับจัดการ AJAX

```
<head>
    <script src="../AJAX/framework.js"> </script>
    <script>
    function AJAXCall ( ) {
        var data = getFromdataa ('form1');
        var URL = 'AJAX_test.php';      // นำข้อมูลไปกำหนดให้แก่ฟังก์ชัน AJAXLoad()
                                        // เพื่อส่งออกไป ฟังก์ชันนี้อยู่ใน framework.js
        AJAXLoad ('post', URL,data, 'DisplayAJAX');
    }
    </script>
</head>
```

4) จากฟังก์ชัน AJAXCall ( ) ที่สร้างขึ้นในข้อที่ 3 ต้องพิจารณาต่อไปอีกว่า จะเรียกฟังก์ชันนี้ขึ้นมาทำงานเมื่อใด เช่นอาจเพิ่มปุ่ม Button เข้าไป แล้วเพื่อเรียกใช้ฟังก์ชัน AJAXCall ดังนี้

ตัวอย่างที่ 12.7 การสร้างปุ่มเพื่อเรียกใช้ฟังก์ชัน AJAXCall

```
<input type="button" value="OK" onclick = "AJAXCall ( )"/>
```

12.8.2 กระบวนการฝั่งเซิร์ฟเวอร์

ทางด้านเซิร์ฟเวอร์นั้น ก็ใช้วิธีที่เหมือนกับการเขียนเว็บเพจปกติ โดยไม่มีอะไรเป็นพิเศษสำหรับ AJAX แต่ก็พอสรุปเป็นหลักการได้ดังนี้

1) การรับข้อมูลที่ถูกส่งเข้ามาผ่านตัวแปร $_GET หรือ $_POST ขึ้นอยู่กับเมธอดที่ถูกส่งเข้ามา

2) หลังการประมวลผล ก็ส่งข้อมูลกลับออกไปด้วยเพื่อแสดงผลด้วยฟังก์ชัน echo ( ) หรือ print ( ) ตามปกติ ถ้าข้อมูลที่ส่งนั้นยาวมากก็ควรนำมารวมเป็นสตริงเดียวกันก่อนส่ง

12.9 ปัญหาภาษาไทยใน AJAX

ในการใช้ AJAX ร่วมกับภาษา PHP จะมีปัญหาสำคัญอย่างหนึ่ง คือ การแสดงภาษาไทยจะกลายเป็นอักขระที่อ่านไม่รู้เรื่อง เช่น จากตัวอย่างที่แล้ว หากไม่ใช้ฟังก์ชัน header ( ) และถ้าใส่ข้อมูลที่ไม่ใช่ตัวเลข จะได้ผลลัพธ์เป็นคำว่า "ค่าที่ใส่ไม่ใช่ตัวเลข"

สำหรับปัญหาอักขระภาษาไทยที่เกิดขึ้นกับAJAX/PHPโดยส่วนใหญ่จะมี 2 ลักษณะ คือ

12.9.1 ถ้าผลลัพธ์ที่ส่งกลับมาแสดงผลภาษาไทยไม่ถูกต้อง แก้ปัญหาโดยใช้ฟังก์ชัน header ( ) แล้วกำหนด Content-Type เป็น tis-620 หรือ utf-8 เช่น

ตัวอย่างที่ 12.8 การใช้ฟังก์ชัน header ( ) เพื่อควบคุมการแสดงผลภาษาไทย

```
header ("Content-Type:text/plain; charset=tis-620");
```

12.9.2 ถ้ามีการส่งข้อมูลที่มีภาษาไทยจากฝั่งบราวเซอร์ขึ้นไป ก่อนส่ง JavaScript จะเปลี่ยนเป็น Unicode แบบ utf-8 ถึงแม้ว่า utf-8 จะรองรับภาษาไทยได้ แต่ก็อาจเกิดปัญหาในบางกรณี ดังนั้นเพื่อเป็นการป้องกันปัญหาเกี่ยวกับอักขระ ทางฝั่งเซิร์ฟเวอร์ที่รับข้อมูลควรเข้ารหัสใหม่ให้เป็น tis-620 ด้วย ฟังก์ชัน iconv ( ) ฟังก์ชันนี้ก็คุ้นเคยกันดีอยู่แล้วเพราะในหลายๆ บทที่ผ่านมาก็ใช้งานมาตลอด เช่น

ตัวอย่างที่ 12.9 การส่งข้อมูลที่มีภาษาไทยจากฝั่งบราวเซอร์ขึ้นไป ก่อนส่ง JavaScript

```
$input = iconv ("utf-8","tis-620",$_POST["input"]);
```

ในทางปฏิบัติ การรับข้อมูลของทางฝั่งเซิร์ฟเวอร์ที่ส่งผ่าน AJAX เข้ามา ควรเข้ารหัสก่อนเสมอ ถึงแม้ว่าข้อมูลที่ผ่านส่งไปจะไม่มีภาษาไทยอยู่ด้วยก็ไม่มีปัญหา หลังการใช้ฟังก์ชัน iconv ( ) ข้อมูลที่ได้จะตรงกับข้อมูลที่ส่งไปเสมอ

12.10 การส่งผลลัพธ์กลับมาเป็น JavaScript

การส่งผลลัพธ์กลับมาในรูปแบบสตริงธรรมดา อาจทำให้ไม่สามารถแก้ไขอะไรได้มากนัก โดยเฉพาะอย่างยิ่งในกรณีที่ต้องควบคุมการทำงานจากเซิร์ฟเวอร์ วิธีที่จะทำได้ก็คือ ส่งผลลัพธ์กลับคืนมาในรูปแบบคำสั่งของ JavaScript โดยแยกพิจารณาระหว่างฝั่งเซิร์ฟเวอร์และบราวเซอร์ดังนี้

12.10.1 การส่งผลลัพธ์ที่เป็น JavaScript ในฝั่งเซิร์ฟเวอร์

จะต้องสร้างคำสั่ง JavaScript ในรูปแบบของสตริงธรรมดา โดยจะมีจำนวนคำสั่งเท่าไหร่ก็ได้ แต่ต้องไม่มีแท็ก <script> และ </script> คร่อมระหว่างคำสั่งเหล่านั้น นอกจากนี้เพื่อให้ทางฝั่งบราวเซอร์สามารถตรวจสอบได้ว่า ผลลัพธ์ที่ได้รับมานั้นเป็น JavaScript ควรกำหนดเฮดเดอร์กำกับไว้ด้วยฟังก์ชันheader ( ) ในลักษณะดังนี้

```
header ("Content-type:text/javascript; charset=tis-620");
```

หรือบางครั้งการจะส่งผลลัพธ์กลับไปแบบใด อาจขึ้นกับเงื่อนไข ดังนั้นอาจตรวจสอบเงื่อนไปก่อนกำหนดเฮดเดอร์ เช่น

ตัวอย่างที่ 12.10 การเขียนคำสั่งเพื่อให้รองรับภาษาไทยผ่านฟังก์ชัน header ( )

```
<?php
        $x = ...;
        if (empty($x)) {
                header ("Content-Type: text/plain; charset=tis-620");
                echo"...";
        } else {
                header ("Content-Type: text/javascript;charset=tis-620");
                //สร้างคำสั่ง JavaScriptในแบบสตริง กี่คำสั่งก็ได้ แล้วส่งออกไป
                $js = "
                var el = document.getElementById('x');
                el.style.display='none
                alert ('...');
                ";
                echo $js;
        }
?>
```

12.10.2 การจัดการผลลัพธ์ที่เป็น JavaScript ฝั่งบราวเซอร์

หากเขียนโปรแกรม Javascript เอง ปกติก็ต้องมีการตรวจสอบเฮดเดอร์ของผลลัพธ์ก่อนแล้วใช้ฟังก์ชัน eval ( ) ของใน JavaScript ในการประมวลผล แต่เนื่องจากใน AJAX Framework นั้นผู้เขียนได้ทำให้รองรับผลลัพธ์ที่เป็นคำสั่ง JavaScript โดยอัตโนมัติอยู่แล้ว ดังนั้นจึงไม่ต้องทำอะไรเพียงแต่ฝั่งเซิร์ฟเวอร์ต้องกำหนดเฮดเดอร์ก่อนไม่เช่นนั้นจะไม่สามารถจำแนกได้ว่า ผลลัพธ์เป็นสตริงธรรมดาหรือเป็น JavaScript และสำหรับอิลิเมนต์ที่จะใช้แสดงผล หากจะกำหนดวิธีการผ่าน JavaScript อาจไม่มีอิลิเมนต์นี้ก็ได้ และกำหนดค่าให้กับฟังก์ชัน AJAXLoad ( ) เป็น null เช่น

```
        AJAXLoad ('get', URL, data, null);
```

เพื่อให้เข้าในแนวทางการส่งผลลัพธ์เป็น JavaScript มากยิ่งขึ้น ตัวอย่าง การส่งผลลัพธ์เป็น JavaScript โดยการจะเป็นการรับข้อมูลตัวเลข 2 ตัวและเครื่องหมายแล้วส่งด้วยเทคนิค AJAX ไปคำนวณที่เซิร์ฟเวอร์ หลังจากนั้นส่งผลลัพธ์กลับมาในแบบคำสั่ง JavaScript

ตัวอย่างที่ 12.11 ตัวอย่างการเรียกใช้ฟังก์ชัน AJAXcall ในไฟล์ index.html

```
<head>
        .
        .
```

```
        <script src="../AJAX/framwork.js"></script>
        <script>
        function AJAXcall ( ) {
                var data = getFormData ('from1');
                var URL = 'calculator.php';
                AJAXLoad ('post', URL, date, null);
                //ให้อิลิเมนต์ที่ใช้แสดงผลเป็น null เพราะส่งผลลัพธ์เป็น JavaScript
        }
        </script>
</head>
```

เพิ่มตัวจัดการอีเวนต์ (Event) ให้กับปุ่ม Button โดยการเรียกใช้ฟังก์ชัน AJAXCall ( )

```
        <input type="button" value"คำนวน" onclick = "AJAXCall ( )"/>
```

ตัวอย่างที่ 12.12 ไฟล์ calculator.php เป็นเว็บเพจฝั่งเซิร์ฟเวอร์สำหรับรับข้อมูลเพื่อประมวลผล

```
<?php
        $num1 = $_POST ['num1'];
        $num2 = $_POST ['num2'];
        $op = $_POST ['op'];
        $result = 0 ;
        switch ($op) {
                case "+" : $result = num1 + $num2; break;
                case "-" : $result = num1 - $num2; break;
                case "*" : $result = num1 * $num2; break;
                case "/" : $result = num1 / $num2; break;
        }
        // สร้างคำสั่ง JavaScript ในแบบสตริงของ PHP มาแทรกลงไปได้
        $js = "
                var msg = '$num1 $op $num2 =$result';
        ";
        header ("Content-Type:Text/javascript;Charset=tis-620");
        echo $js;
?>
```

การใช้ AJAX กับการตรวจสอบข้อมูล เทคนิคนิยมที่ใช้กันมากในปัจจุบัน สำหรับตัวอย่างนี้จะเป็นการตรวจสอบล็อกอินว่าใช้งานได้หรือไม่ โดยส่งล็อกอินผ่าน AJAX ไปยังเซิร์ฟเวอร์ แล้วส่งผลลัพธ์เป็น JavaScript

ตัวอย่างที่ 12.13 การใช้ AJAX กับการตรวจสอบข้อมูล ภายในไฟล์ index.php

```
<head>
        .
        .
        <script src="../AJAX/framework.js"></script>
        <script>
        function AJAXCall ( ) {
                var date = getFormData ('form1');
                var url = 'check_login.php';
                AJAXLoad ('post', url. date, 'displayAJAX');
        }
        // ฟังก์ชันนี้จะถูกเรียกขึ้นมาทำงานจากฝั่งเซิร์ฟเวอร์
        // เพื่อแสดงผลลัพธ์ว่าล็อกอินนั้นสามารถใช้ได้หรือไม่
        function displayResult (valid) {
                var el = document.getElementById('displayAJAX');
                If (!valid) {
                el.style.color = 'red';
                el.innerHTML ='ชื่อนี้ใช้ไม่ได้ หรือมีผู้อื่นใช้แล้ว กรุณาเปลี่ยนชื่อใหม่'
                Document.getElementById('login').select();
        } else {
                el.sytle.color = 'black';
                el.innerHTML = 'ชื่อนี้สามารถใช้งานได้';
                }
        }
</script>
</head>
```

กำหนดอีเวนต์ onblur เพื่อว่าหลังป้อนข้อมูลไปแล้ว และย้ายไปยังส่วนนำเข้าข้อมูลอื่นๆ จะใช้อีเวนต์นี้ในการส่งล็อกอินผ่าน AJAX ไปตรวจสอบ โดยในมุมมองอัพเดทให้เพิ่มอีเวนต์ onblur และ เพิ่มแท็ก ดังนี้

```
<input name="login" type="text" id="login" onblur="AJAXCall ( )"/>
<span id="displayAJAX"></span>
```

ตัวอย่างไฟล์ check_login.php เป็นเว็บเพจฝั่งเซิร์ฟเวอร์ที่รับข้อมูลล็อกอินไปตรวจสอบ แล้วส่งผลลัพธ์เป็น JavaScript กลับมาโดยในที่นี้จะเป็นพียงการเรียกใช้ฟังก์ชัน displayReuslt ที่เว็บเพจ index.php ขึ้นมาทำงาน

ตัวอย่างที่ 12.14 รับข้อมูลล็อกอินไปตรวจสอบ แล้วส่งผลลัพธ์เป็น JavaScript

```
<?php
    $login = $_POST['LOGIN'];
    $patter = "^[a-zA-Z0-9]{3,10}$";
    $users = array ("abc","123","xyz"); //สมมติชื่อผู้ใช้ที่มีอยู่แล้ว
    // ตรวจสอบ login แล้วเรียกฟังก์ชัน Javascripth displayResult ( )
    // ที่เว็บเพจ index.php ขึ้นมาทำงาน
    If (!eregi (pattern,$login) || in_array ($login,$users)) {  //ถ้าล็อกอินไม่สำเร็จ
        $js = "displayResult (false);";
    } else { //ถ้าล็อกอินสำเร็จ
        $js = "displayResult (ture);";
    }
    header ("Content-Type:text/javascript;Charset=tis-620");
    echo $js;
?>
```

12.11 การอัปเดทส่วนนำเข้าข้อมูล Select ด้วยเทคนิค AJAX

โดยปกติแล้วตัวเลือกของส่วนนำเข้าข้อมูล Select จะสร้างไว้ล่วงหน้าด้วยแท็ก <option> เช่น อัพเดตส่วนนำเข้าข้อมูล Select ด้วยเทคนิค AJAX

การเปลี่ยนแปลงตัวเลือกของส่วนนำเข้าข้อมูล Select จะต้องอาศัยคำสั่ง JavaScript เป็นหลัก ดังนั้นก่อนที่จะกล่าวถึงเทคนิคการใช้ร่วมกับ AJAX จะกล่าวถึงการใช้ JavaScript ในการแปลงรายการใน Select ก่อน

12.11.1 การสร้างตัวเลือกด้วย JavaScript

หากต้องการจะเพิ่มตัวเลือกนั้นต้องเริ่มที่การสร้างตัวเลือกใหม่ขึ้นมาก่อน การสร้างตัวเลือกใหม่นั้นวิธีที่ง่ายที่สุด คือ การสร้างด้วยออบเจ็กต์ Option โดยมีรูปแบบดังนี้

```
option ('text', 'value')
```

เมื่อ text หมายถึง ข้อความที่ปรากฏบนตัวเลือกนั้น ต้องกำหนดในรูปแบบสตริง

value หมายถึง ค่าของตัวเลือกนั้น ต้องกำหนดในรูปแบบสตริง

ตัวอย่างที่ 12.15 การสร้างตัวเลือกด้วย JavaScript

```
var opt = new option ('Thailand', 'th');
```

เทียบได้กับในรูปแบบ HTML ต่อไปนี้

```
<option value="th">Thailand</option>
```

12.11.2 การเพิ่มตัวเลือกด้วย JavaScript

การเพิ่มตัวเลือกเข้าไปในส่วนนำเข้าข้อมูล Select นั้นปกติทำได้หลายวิธี แต่บางวิธีจะได้ผลเฉพาะกับบราวเซอร์ IE เท่านั้น ดังนั้นจะกล่าวถึงเฉพาะวิธีที่ใช้ได้ทั้ง IE และ Firefox โดยใช้วิธีการกำหนดค่าให้แก่ คุณสมบัติตัวเลือก เช่น ถ้าต้องการเพิ่มตัวเลือกเข้าไปให้เป็นรายการลำดับที่สาม ก็กำหนดสคริปต์คำสั่งดังนี้

ตัวอย่างที่ 12.16 การเพิ่มตัวเลือกด้วย JavaScript

```
Document.getElementById ('id').option [2] = opt;     // เมื่อรายการแรกลำดับเป็น0
```

12.11.3 การลบตัวเลือกด้วย JavaScript

การลบนั้นเพื่อให้เป็นแนวทางเดียวกับการเพิ่ม ก็อาจใช้วิธีกำหนดคุณสมบัติตัวเลือกของรายการลำดับนั้นให้เป็น Null เช่น

ตัวอย่างที่ 12.17 การลบตัวเลือกด้วย JavaScript

```
Document.getElementById ('id').option [2] = Null;     // ลบรายการที่ 3
```

12.11.4 การใช้ AJAX กับการอัพเดทส่วนนำเข้าข้อมูล Select

หลังจากที่ทราบเทคนิคการเปลี่ยนแปลงตัวเลือกด้วย JavaScript แล้ว ต่อไป คือ การนำมาประยุกต์ใช้ร่วมกับ AJAX การเปลี่ยนแปลงตัวเลือกที่ได้กล่าวมานั้น ข้อมูลที่จะทำตัวเลือกถูกกำหนดตายตัวมาแล้ว แต่เป็นกรณีที่ข้อมูลเก็บอยู่ในฐานข้อมูลทางฝั่งเซิร์ฟเวอร์ อาจไม่สามารถใช้วิธีตามที่กล่าวมาแล้วได้ แต่จากการที่สร้างคำสั่งที่เซิร์ฟเวอร์ แล้วส่งผ่านทาง AJAX จึงแสดงผลบนบราวเซอร์ได้

เว็บเพจที่จะแสดงตัวเลือกทางด้านบราวเซอร์ ต้องประกอบด้วยสิ่งที่จะใช้กำหนดเงื่อนไขในการเปลี่ยนตัวเลือก โดยทั่วไปนิยมใช้ส่วนนำเข้าข้อมูล Select คู่กัน เช่น Select อันแรกแสดงชื่อฐานข้อมูลที่มีใน MySQL และอีกอันแสดงชื่อตารางที่มีอยู่ในฐานข้อมูลที่เลือกใน Select อันแรก เป็นต้น

เมื่อส่งข้อมูลจาก Select อันแรกผ่าน AJAX ขึ้นไป ทางฝั่งเซิร์ฟเวอร์จะนำข้อมูลนั้นไปใช้เป็นเงื่อนไขในการอ่านข้อมูลจากตาราง เพื่อนำมาสร้างตัวเลือกใน Select อันที่สอง

หลังจากได้ข้อมูลต่อไป คือ การส่งผลลัพธ์กลับไปเพื่ออัพเดทส่วนนำเข้าข้อมูล Select โดยก่อนที่จะอัพเดท ต้องลบตัวเลือกเดิมทั้งหมดออกไปก่อน แต่การลบตัวเลือกนั้นไม่ขึ้นกับข้อมูล ดังนั้นในที่นี้การลบตัวเลือกจะสร้างเป็นฟังก์ชันเอาไว้ที่เว็บเพจด้านบราวเซอร์ แล้วใช้วิธีสั่งคำสั่ง JavaScript จากด้านเซิร์ฟเวอร์มาเรียกฟังก์ชันนี้ขึ้นมา

ต่อไปคือ การนำข้อมูลที่อ่านได้จากฐานข้อมูลจากขั้นตอนก่อนนี้ มาสร้างเป็นตัวเลือกในรูปแบบคำสั่ง JavaScript เพื่อส่งกลับไปประมวลผลที่ด้านบราวเซอร์ ทั้งนี้ตัวเลือกอาจมีมากกว่า 1 รายการดังนั้นอาจต้องใช้การวนลูปเพื่อสร้างคำสั่ง JavaScript ตามจำนวนตัวเลือกที่มี

12.11.5 ขั้นตอนการออกแบบ และเขียนสคริปต์

สร้างเว็บเพจจำนวน 2 เว็บเพจ โดยเว็บเพจแรกสำหรับวางส่วนนำเข้าข้อมูล Select จำนวน 2 รายการ เอาไว้ทดสอบ ส่วนอีกเว็บเพจหนึ่งฝั่งเซิร์ฟเวอร์สำหรับอ่านชื่อตารางฐานข้อมูล แล้วส่งผลลัพธ์กลับมาในแบบ JavaScript

ตัวอย่างที่ 12.18 ตัวอย่างข้อมูลภายในไฟล์ index.php

```
<head>
        .
        .
        <script>
        function AJAXCall ( ) {
                var data = getFormData ('form1');
                var URL = 'update_tables.php';
                AJAXLoad ('get', URL, data, null);
        }
        function removeOption ( )  {
                var el = document.getElementByld ('table');
                while (el.length > 0)    {
                        el.option [0] = null;
                }
        }
        </script>
</head>
```

ตำแหน่งส่วนนำเข้าข้อมูล Select อันแรก (ที่ใช้เก็บชื่อฐานข้อมูล) สิ่งที่ต้องทำก็คือ

1) เพิ่มอีเวนต์ change เพื่อเรียกใช้ AJAX ในการอัพเดท Select

2) เนื่องจากเป็นการแสดงชื่อฐานข้อมูล ควรอ่านจาก MySQL โดยตรงมากกว่าที่จะสร้างเป็นรายการไว้ล่วงหน้าดังนั้นในส่วนรายตัวเลือกใช้ สคริปต์ PHP ในการอ่านชื่อฐานข้อมูลแล้วนำมาสร้างเป็นรายการตัวเลือก

ตัวอย่างที่ 12.19 แสดงส่วนของการติดต่อกับฐานข้อมูล

```
<select id = "database" name = "database" onchange = "AJAXCall ( )">
<?php
        @mysql_connect ("localhost", "root",  "leaf")  or die (mysql_error ( ))'
        $dbs = mysql_list_dbs();
        while (list ($db) = mysql_fetch_row ($dbs)) {
                echo "<option value = $db</option>";
        }
?>
</select>
```

เพื่อให้เกิดการอัพเดททันทีที่เปิดเว็บเพจ ให้เพิ่มการเรียกใช้ฟังก์ชัน AJAXCall ( ) ไว้ที่ส่วนท้ายของเว็บเพจดังนี้

```
        </form>
        <script> AJAXCall ( ); </script>
</body>
```

update_tables.php เว็บเพจนี้จะนำชื่อฐานข้อมูลที่รับเข้ามาไปใช้การอ่านชื่อตารางที่มีอยู่ในฐานข้อมูลนั้น แล้วนำมาสร้างเป็นคำสั่ง JavaScript เพื่อส่งกลับไปอัพเดทที่ส่วนนำเข้าข้อมูล Select ที่แสดงชื่อตาราง ตามหลักการที่ได้กล่าวมาแล้ว

ตัวอย่างที่ 12.20 ส่วนของการ Update ข้อมูลภายในไฟล์ update_tables.php

```
<?php
        @mysql_connect ("localhost", "root", "leaf">  or die (mysql_error ( ));
        $db = $_GET ['database'];
        $tbs = mysql_list_tables($db);
        // คำสั่ง JavaScript สำหรับลบ Option เดิม โดยเรียกฟังก์ชัน remeveOption ( )
        // สร้างไว้ที่เว็บเพจ index.php
        $js = "removeOption ( );";
        If (mysql_num_rows($tbs) == 0 {       // ถ้าฐานข้อมูลนั้นไม่มีตารางอยู่เลย
                $js   .=  "
                var opt = new Option ('', '');
                Document.getElementByID ('table') .options [0] = opt;
        } else  {
```

```
            $i = 0;
            while (list ($tb) = mysql_fetch_row ($tbs)) {
                //คำสั่ง javascript สำหรับสร้าง และเพิ่ม Option ใหม่จนครบทุกตัว
                $js .= "
                    var opt = new Option ('$tb', '$tb');
                    Document.getElementByID ('table') .options [$i] = opt;
                ";
                $i++;
            }
        }
        header ("Content-Type:text/Javascript; charset=tis-620);
        echo $js;
?>
```

12.12 อีเวนต์

บัญชา ปะสีละเตสัง. (2551: 103) อธิบายความหมายของอีเวนต์ (Event) คือ เหตุการณ์หรือการกระทำบางอย่างที่เกิดขึ้น เช่น การคลิกหรือเคลื่อนย้ายเมาส์ การกดปุ่มคีย์บอร์ด เป็นต้น ทั้งนี้มักใช้อีเวนต์ที่เกิดขึ้นเป็นตัวกำหนดให้เกิดการตอบสนอง หรือกระทำบางอย่างต่อไป เช่น เมื่อคลิกปุ่มก็ให้ส่งข้อมูลจากฟอร์มไปยังเซิร์ฟเวอร์ หรือ เมื่อมีการพิมพ์คีย์บอร์ด ก็อาจตรวจสอบว่าข้อมูลที่ใส่เข้ามาตรงตามเงื่อนไขที่กำหนดหรือไม่ เป็นต้น

ชนิดของการเกิดอีเวนต์ เนื่องจากอีเวนต์เกิดได้จากหลายสาเหตุ เช่น การคลิกเมาส์ การเคลื่อนย้ายเมาส์ การพิมพ์ข้อความ เป็นต้น นอกจากนี้แต่ละอิลิเมนต์ก็สามารถเกิดอีเวนต์ได้เพียงบางลักษณะเท่านั้น เช่น div สามารถเกิดอีเวนต์เกี่ยวกับเมส์ได้ แต่ไม่สามารถเกิดอีเวนต์เกี่ยวกับคีย์บอร์ด เป็นต้น ดังนั้นจึงควรทราบก่อนว่าใน JavaScript มีอีเวนต์ชนิดใดบ้าง เพื่อจะได้นำไปใช้ร่วมกันได้อย่างถูกต้อง มีรายละเอียดดังนี้

ตารางที่ 12.1 แสดงอีเวนต์ของ JavaScript สำหรับกำหนดให้กับอิลิเมนต์

Event Handler	การเกิดอิลิเมนต์
onclick	เกิดเมื่อคลิกที่อิลิเมนต์ (กดแล้วปล่อย)
ondblclick	เกิดเมื่อดับเบิลคลิก
onmousedown	เกิดเมื่อกดเมาส์ลงไป (เกิดทันทีที่กด)
onmouseup	เกิดเมื่อปล่อยมือจากการกดเมาส์

ตารางที่ 12.1 (ต่อ)

Event Handler	การเกิดอิลิเมนต์
onmouseover	เกิดเมื่อเคลื่อนย้ายเมาส์เข้าไปในอิลิเมนต์
onmouseout	เกิดเมื่อนำเมาส์ออกจากอิลิเมนต์
onmousemove	เกิดเมื่อเคลื่อนย้ายเมาส์ไปบนอิลิเมนต์
onkeypress	เป็นลักษณะการพิมพ์ทั่วไป คือ กดแล้วปล่อย
onkeydown	เกิดเมื่อกดปุ่มคีย์บอร์ด (เกิดทันทีที่กด)
onkeyup	เกิดเมื่อปล่อยมือจากปุ่มคีย์บอร์ด
onabort	เกิดเมื่อขัดจังหวะขณะที่กำลังโหลดรูปภาพ เช่น คลิกปุ่ม Stop ของบราวเซอร์ขณะที่ยังโหลดภาพไม่สมบูรณ์
onblur	เกิดเมื่ออิลิเมนต์นั้นเสียโฟกัส หรือย้ายโฟกัสไปยังอิลิเมนต์
onchange	เกิดเมื่อเปลี่ยนข้อความใช้ช่องรับข้อมูล เช่น text, textarea หรือเกิดเมื่อเปลี่ยนรายการที่เลือกจากเดิมไปเป็นรายการอื่นใน <select>
onerror	เกิดเมื่อมีข้อผิดพลาดเกิดขึ้นขณะที่กำลังโหลดภาพ
onfocus	เกิดเมื่ออิลิเมนต์นั้นได้รับโฟกัส
onload	เกิดเมื่อเริ่มทำการโหลดเอกสารหรือรูปภาพ
onreset	เกิดเมื่อคลิกปุ่ม Reset ของฟอร์ม
onresize	เกิดเมื่อเปลี่ยนขนาดของวินโดว์
onselect	เกิดเมื่อทำแถบสีเพื่อเลือกข้อความ
onsubmit	เกิดเมื่อส่งข้อมูลจากฟอร์มไปยังเซิร์ฟเวอร์
onunload	เกิดเมื่อยกเลิกการโหลดเอกสาร เช่น คลิกปุ่ม Stop

หมายเหตุ

Event Handler คือ ตัวจัดการอีเวนต์ หรือทำหน้าที่การเรียกสคริปต์ส่วนที่ใช้ตอบสนองอีเวนต์ขึ้นมาทำงานเมื่อเกิดอีเวนต์ขึ้น ถ้าตัดคำว่า "on" ข้างหน้า Event Handler ออกก็จะกลายเป็นชนิดของอีเวนต์นั้นๆ

12.13 การกำหนดอีเวนต์

ก่อนการกำหนดอีเวนต์จะต้องตรวจสอบและพิจารณาก่อนว่าต้องการกระทำเมื่อเกิดอีเวนต์ใดและเกิดกับอิลิเมนต์ใด โดยดูว่าอิลิเมนต์นั้นรองรับอีเวนต์ที่ต้องการหรือไม่ เช่น หากต้องการจะกระทำบางอย่างเมื่อผู้ใช้คลิกที่ ก็ต้องดูว่า รองรับอีเวนต์ที่เกี่ยวกับ "click" หรือไม่เป็นต้น

12.13.1 การกำหนดอีเวนต์แบบแอททริบิวต์

วิธีนี้จะเขียนสคริปต์แทรกไว้ในแท็กของอิลิเมนต์ที่เกิดอีเวนต์ โดยนำ Event Handler ของอีเวนต์นั้นมาเป็นแอททริบิวต์ตัวหนึ่ง เช่น

ตัวอย่างที่ 12.21 การแทรกสคริปต์ในแท็กของอิลิเมนต์ที่เกิดอีเวนต์

```
<body onclick = "alert ('Hello')">
```

หมายความว่าเมื่อคลิกที่เอกสารจะเกิด alert ที่แสดงข้อความ Hello วิธีนี้เหมาะสำหรับสคริปต์ที่ไม่ยาวมากนัก เพราะหากเป็นสคริปต์ยาวๆ อาจจะเขียนไม่สะดวก เช่น

ตัวอย่างที่ 12.22 การแทรกสคริปต์ในแท็กของอิลิเมนต์ที่เกิดอีเวนต์ แบบมีเงื่อนไข

```
<div id="dv" onmousedown="if (confirm ('Are you 18+')) {
location.href='index.html'; } else { window.close ( ) }" > 18+ ? </div>
```

จากสคริปต์ หมายถึง เมื่อเกิดอีเวนต์ mousedown ที่ จะปรากฏกล่องโต้ตอบ (Dialog box) ยืนยันอายุ โดยหากยืนยันว่าคุณอายุมากกว่า 18 จะโหลดหน้า index.html หากไม่จะปิดบราวเซอร์ที่กำลังเรียกใช้งานอยู่

12.13.2 การกำหนดอีเวนต์แบบพร็อปเพอร์ตี้

วิธีนี้จะไม่เขียน Event Handler ไว้ในแท็กของอิลิเมนต์ที่ต้องการดักอีเวนต์ แต่จะนำมากำหนดในรูปแบบพร็อปเพอร์ตี้ของอิลิเมนต์นั้นแทน เช่น

ตัวอย่างที่ 12.23 การกำหนดอีเวนต์แบบพร็อปเพอร์ตี้

```
<script>
        function fa ( ) {
                .
                .
        }
</script>
<div id="dv"> Click Me </div>
<script>
        document.getElementById ("dv").onclick = f1;
</script>
```

จะพบว่าชื่อฟังก์ชันที่นำมากำหนดให้กับ Event Handler นั้นจะต้องไม่มีวงเล็บต่อท้าย ต้องมีเฉพาะชื่อฟังก์ชันอย่างเดียวเท่านั้น ทั้งนี้อิลิเมนต์เดียวกันสามารถมีได้มากกว่า 1 Event Handler เช่น

ตัวอย่างที่ 12.24 การกำหนดให้อิลิเมนต์เดียวมีมากกว่า 1 Event Handler

```
var el = document.getElementById ("dv");
el.onclick = f1;
el.onmouseover = f2;
el.onmousout = f3;
```

หรือจะเขียนแบบ Anonymous Function ก็ได้ เช่น

ตัวอย่างที่ 12.25 การกำหนดอิลิเมนต์เดียว แบบ Anonymous Function

```
var el = document.getElementById ("dv");
el.onclick = function ( ) {
        .
        .
}
```

12.14 ออบเจ็กต์อีเวนต์

เมื่อเกิดอีเวนต์ขึ้นอาจจะมีข้อมูลบางอย่างเกี่ยวกับอีเวนต์นั้นที่อาจต้องการทราบ เช่น ปุ่มเมาส์หรือคีย์บอร์ดที่ถูกกด หรือตำแหน่งที่เกิดอีเวนต์ เป็นต้น เมื่อมีอีเวนต์เกิดขึ้น สำหรับ IE แล้วข้อมูลต่างๆ เกี่ยวกับอีเวนต์จะถูกเก็บไว้ในออบเจ็กต์อีเวนต์ (Object Event) ทั้งนี้จะต้องอ้างถึงข้อมูลเหล่านั้นในแบบพร็อปเพอร์ตี้ พร็อปเพอร์ตี้ที่น่าสนใจมีรายละเอียดดังนี้

ตารางที่ 12.2 แสดงอีเวนต์ของ JavaScript แบบพร็อปเพอร์ตี้

พร็อปเพอร์ตี้	ความหมาย
allKey	ตรวจสอบว่าคีย์บอร์ดปุ่ม "Alt" ถูกกดขณะเกิดอีเวนต์หรือไม่
ctrlKey	ตรวจสอบว่าคีย์บอร์ดปุ่ม "Ctrl" ถูกกดขณะเกิดอีเวนต์หรือไม่
shiftKey	ตรวจสอบว่าคีย์บอร์ดปุ่ม "Shift" ถูกกดขณะเกิดอีเวนต์หรือไม่
keyCode	รหัสของปุ่มคีย์บอร์ดที่ถูกกด
button	ตรวจสอบว่าปุ่มของเมาส์ปุ่มใดที่ถูกคลิก โดย 1=ซ้าย, 2=ขวา, 4=กลาง และ 3=ซ้าย+ขวา
fromElement	อิลิเมนต์ที่เมาส์ชี้อยู่ก่อนจะมาที่อิลิเมนต์ปัจจุบัน เช่น ตอนแรกเมาส์ชี้อยู่ที่ ต่อมาเลื่อนเมาส์มายัง ถ้าอ่านค่า fromElement ที่ div2 จะได้ค่าเป็น div1
toElement	อิลิเมน์ที่เมาส์ชี้อยู่หลังจากเลื่อนไปจากอิลิเมนต์เดิม เช่น ตอนแรกเมาส์ชี้อยู่ที่ ต่อมาเลื่อนเมาส์มายัง ถ้าอ่านค่า fromElement ที่ div1 จะได้ค่าเป็น div2
srcElement	อิลิเมนต์ที่ทำให้เกิดอีเวนต์ เช่น ถ้าคลิกที่ จะได้ค่า srcElement เป็น div1
type	ชนิดของอีเวนต์ที่เกิดขึ้น คือ การนำชื่อ Event Handler มาตัดคำว่า "on" ออกไป เช่น เป็นอีเวนต์ "onclick" จะได้ type เป็น "click" เป็นต้น

ตารางที่ 12.2 (ต่อ)

พร็อปเพอร์ตี้	ความหมาย
clientX, clientY	เป็นการวัดระยะโดยเทียบกับพื้นที่ในการแสดงผลของบราวเซอร์
offsetX, offsetY	เป็นการวัดค่าจะวัดจากขอบด้านซ้าย และด้านบนของอิลิเมนต์ที่ทำให้เกิดอีเวนต์
screenX, screenY	เป็นการวัดระยะโดยเทียบกับหน้าจอด้านซ้าย และด้านบนมายังจุดที่เกิดอีเวนต์
cancelButton	ใช้ในการหยุดการเกิด Event Bubbling จาก child ไปยังพาเรนต์
returnValue	กำหนดค่าที่ส่งคืนให้แก่ระบบ มักใช้เพื่อยกเลิกพฤติกรรมปกติของอีเวนต์ เช่น ไม่ต้องการแสดงเมนูเมื่อคลิกขวาก็กำหนดค่า returnValue เป็น false เป็นต้น

เนื่องจากในแต่ละพร็อปเพอร์ตี้ยังมีรายละเอียดเพิ่มเติมอีกมาก จะนำไปกล่าวถึงในหัวข้อต่อๆ ไป แยกตามความเกี่ยวข้องของอีเวนต์ สำหรับแนวทางการใช้พร็อปเพอร์ตี้ของออปเจ็กต์อีเวนต์ มีตัวอย่างดังนี้

ตัวอย่างที่ 12.26 การใช้พร็อปเพอร์ตี้ของออปเจ็กต์อีเวนต์

```
<script>
        function eventType ( ) {
                var ev_type = event.type;
                alert ("อีเวนต์ที่เกิดขึ้น คือ " + ev_type);
        }
</script>
<body onclick = "eventType ( )" onmousedown = "eventType ( ) ">
```

12.15 อีเวนต์เกี่ยวกับเมาส์

จากเรื่องออบเจ็กต์อีเวนต์ จะเห็นว่า เมื่อเกิดอีเวนต์เกี่ยวกับเมาส์ขึ้น มีข้อมูลหลายอย่างที่สามารถตรวจสอบ หรือนำมาใช้งานต่อไปได้ เช่น ปุ่มหรือตำแหน่งที่ถูกคลิก เป็นต้น

12.15.1 อีเวนต์เกี่ยวกับการเคลื่อนไหวของเมาส์

อีเวนต์ที่เกี่ยวกับการเคลื่อนไหวของเมาส์ได้แก่ mousemove, mouseover และ mouseout โดยมีรายละเอียดดังนี้

1) เมื่อเคลื่อนเมาส์เข้ามาแตะของของอิลิเมนต์จะเกิดอีเวนต์ mouseover

2) ถ้าเคลื่อนเมาส์เข้าไปอยู่เหนืออิลิเมนต์จะเกิดอีเวนต์ mousemove แม้จะหยุดการเคลื่อนไหวก็ยังเป็นอีเวนต์ mousemove

3) หากนำเมาส์ออกไปจากอิลิเมนต์จะเกิดอีเวนต์ mouseout

4) หากไม่กำหนดอีเวนต์ mousemove แต่กำหนดเฉพาะ mouseover เมื่อเคลื่อนที่เมาส์มาวางเหนืออิลิเมนต์ก็จะเกิดอีเวนต์ mousemove

12.15.2 อีเวนต์เกี่ยวกับการกดเมาส์

อีเวนต์ที่เกี่ยวกับการกดเมาส์ได้แก่ click, dblclick, mousedown และ mouseup มีตัวอย่างดังนี้

ตัวอย่างที่ 12.27 อีเวนต์ที่เกี่ยวกับการกดเมาส์

```
<body onmousedown = "window.status=event.type"
        onmouseup = "alert (event.type)"
>
```

12.15.3 การตรวจสอบปุ่มเมาส์ที่ถูกกด

ปุ่มเมาส์ที่ถูกกดสามารถตรวจสอบได้จากพร็อปเพอร์ตี้ button ของออบเจ็กต์อีเวนต์ ค่าที่ได้สำหรับบราวเซอร์ IE มีตัวอย่างดังนี้

ตัวอย่างที่ 12.28 การตรวจสอบปุ่มเมาส์ที่ถูกกด

```
<script>
        function mouseButton ( ) {
                var bt = event.button;
                switch (bt) {
                        case 1 : window.status = "ซ้าย"; break;
                        case 2 : window.status = "ขวา"; break;
                        case 4 : window.status = "กลาง"; break;
                        case default : window.status = "ไม่สามารถระบุได้"; break;
                }
        }
</script>
```

12.16 อีเวนต์เกี่ยวกับคีย์บอร์ด

อีเวนต์ที่เกี่ยวกับคีย์บอร์ด เช่น การตรวจสอบว่าปุ่มถูกกดหรือมีการกดปุ่ม Alt/Ctrl/Shift ด้วยหรือไม่ เป็นต้น ทั้งนี้การตรวจสอบปุ่มนั้น สามารถนำไปใช้ในการป้องกันไม่ให้ผู้ใช้ใส่ข้อมูลที่ไม่ตรงตามเงื่อนไข เช่น ต้องการระบุให้ป้อนข้อมูลชนิดตัวเลข 0-9 เท่านั้น หากผู้ใช้กดปุ่มคีย์ที่ไม่ใช่ 0-9 ก็จะไม่ยอมรับข้อมูลนั้น เป็นต้น

อีเวนต์ที่เกี่ยวกับคีย์บอร์ดมี 3 อีเวนต์ คือ keypress, keydown และ keyup มีตัวอย่างดังนี้

ตัวอย่างที่ 12.29 อีเวนต์ที่เกี่ยวกับคีย์บอร์ด

```
<body   onkeydown = "window.status = event.type" onkeypress = "alert (event.type)"
        onkeyup = "alert (event.type)"
>
```

12.16.1 การตรวจสอบปุ่มคีย์บอร์ดที่ถูกกด

เมื่อคีย์บอร์ดปุ่มใดๆ ถูกกดสามารถตรวจสอบได้ว่าปุ่มนั้น คือ ปุ่มอะไร โดยใช้พร็อปเพอร์ตี้ keyCode แต่ค่า keyCode ที่ได้จะเป็นตัวเลขรหัสที่ใช้แทนปุ่มนั้น มีรายละเอียดของรหัสดังตารางต่อไปนี้

ตารางที่ 12.3 แสดงตัวอย่างรหัสที่ใช้แทนปุ่มคีย์บอร์ด

ปุ่มคีย์บอร์ด	keyCode	ปุ่มคีย์บอร์ด	keyCode
backspace	8	right window key	92
tab	9	select key	93
enter	13	numpad 0-9	96-105
shift	16	multiply	106
ctrl	17	add	107
alt	18	subtract	109
pause/break	19	decimal point	110
caps lock	20	divide	111
escape	27	F1-F12	112-123
page up	33	num lock	144
page down	34	scroll lock	145
end	35	semi-colon	186
home	36	equal sign	187
left arrow	37	comma	188
up arrow	38	dash	189
right arrow	39	period	190
down arrow	40	forward slash	191
insert	45	grave accent	192
delete	46	open bracket	219
เลข 0-9	48-57	back slash	220
อักขระ a-z	65-90	close braket	221

ตารางที่ 12.3 (ต่อ)

ปุ่มคีย์บอร์ด	keyCode	ปุ่มคีย์บอร์ด	keyCode
left window key	91	single quote	222
ก-ฮ	3585-3630	๐-๙	3664-3673

ในทางตรงข้าม ถ้าทราบค่า keyCode และต้องการทราบว่าตรงกับอักขระใด ให้ใช้เมธอด fromCharCode ( ) ของออบเจ็กต์ String ในการแปลง เช่น

```
var char = String.fromCharCode (65);
```

ตัวอย่างที่ 12.30 การเขียนคำสั่งเพื่อควบคุมการป้อนเลข 0-9 เท่านั้น

```
<html>
<head>
	<script>
		function isValidKey ( ) {
			var keyCode = event.keyCode;
			if (keyCode < 48 || event.keyCode > 57) {
				window.status = "key invalid";
				alert ("กรุณาป้อนตัวเลข 0-9 เท่านั้น");
			} else {
				window.status = String.fromCharCode (keyCode);
			}
		}
	</script>
</head>
<body onkeydown = "isValidKey ( )">
</body>
</html>
```

12.17 ตัวอย่างการสร้าง AJAX dictionary จาก longdo ด้วย jQuery

ตัวอย่างการใช้งานต่อไปนี้ เป็นการนำระบบค้นหาคำศัพท์ จากเว็บไซต์ dict.longdo.com เหมาะสำหรับใครที่ต้องการนำระบบค้นหาคำศัพท์ dictionary รูปแบบ AJAX ไปตกแต่งเว็บไซต์ โดยเฉพาะเว็บไซต์เกี่ยวกับการศึกษา มีขั้นตอนดังต่อไปนี้

12.17.1 สร้างไฟล์ php ชื่อ ajax_translate.php มีคำสั่งดังต่อไปนี้

ตัวอย่างที่ 12.31 การสร้าง AJAX dictionary ในไฟล์ ajax_translate.php

```
1<?php
2       header("Content-type:text/html; charset=UTF-8");
3       header("Cache-Control: no-store, no-cache, must-revalidate");
4       header("Cache-Control: post-check=0, pre-check=0", false);
5       if(isset($_GET['keyword']) && trim($_GET['keyword'])!=""){
6             $keyword=trim($_GET['keyword']);
7             $data=file_get_contents("http://dict.longdo.com/mobile.php?search=".$keyword);
8             echo strip_tags($data,"<a><table><td><tr><font><style><meta><br>");
9       } else { // กรณีไม่มีการส่งคำค้นมา
10                  echo "โปรดระบุคำที่ต้องการแปล";  // แสดงข้อความแจ้งเตือน
11      }
12?>
```

12.17.2 สร้างไฟล์สำหรับทดสอบ demo_ajax_translate.php โดยจะมีอยู่ 3 ส่วน หลักๆ ดังนี้

1) ส่วนของ css สำหรับกำหนดรูปแบบ สีข้อความ พื้นหลัง หรืออื่นๆ ตามต้องการ
2) ส่วนของ element องค์ประกอบในการใช้งาน
3) ส่วนของ javascript โดยใช้งานผ่าน jQuery

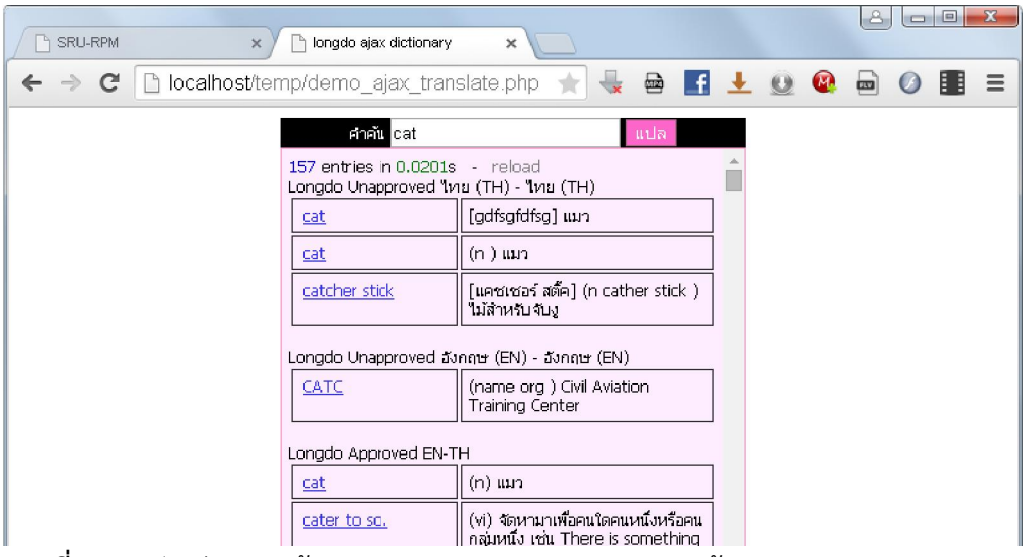
ตัวอย่างที่ 12.32 ไฟล์สำหรับทดสอบ demo_ajax_translate.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=tis-620" />
<title>longdo ajax dictionary</title>
<!--1.ส่วนของ css สำหรับกำหนดรูปแบบ สีข้อความ พื้นหลัง หรืออื่นๆ ตามต้องการ-->
<style type="text/css">
div#myblock_dict{
        font-family:tahoma, "Microsoft Sans Serif", sans-serif, Verdana;
        font-size:12px;
        margin:auto;
        width:350px;
}
```

```
input#translate_it{
	background-color:#F6C;
	color:#FFF;
	border:1px groove #F9C;
	cursor:pointer;
}
div#input_search{
	background-color:#000;
	color:#FFF;
	text-align:center;
}
div#context_search{
	border:1px solid #F9C;
	height:300px;
	overflow:auto;
}
div#context_search{
	font-family:tahoma, "Microsoft Sans Serif", sans-serif, Verdana;
	font-size:12px;
	padding:5px;
	background-color:#FDEDFE;
}
div#context_search a{
	margin-left:3px !important;
	color:#F09;
}
div#context_search td{
	padding:5px !important;
}
</style>
</head>
<body>
```

```
<!--2.ส่วนของ element องค์ประกอบในการใช้งาน-->
<div id="myblock_dict">
<div id="input_search">คำค้น
        <input type="text" name="keyword_q" id="keyword_q" />
        <input type="button" name="translate_it" id="translate_it" value="แปล" />
</div>
<div id="context_search">
</div>
</div>
<!--3.ส่วนของ javascript โดยใช้งานผ่าน jQuery-->
<script type="text/javascript" src="http://code.jquery.com/jquery-latest.min.js"></script>
<script type="text/javascript">
$(function(){
                var loading_img='<center>';
                loading_img+='<img
src="http://static.ak.fbcdn.net/rsrc.php/z5R48/hash/ejut8v2y.gif">';
                loading_img+='</center>';
                $("#keyword_q").click(function(){ // เมื่อคลิกที่ช่องคำค้น
                        $(this).select(); //  ถ้ามีข้อความอยู่ ให้ทำการเลือกข้อความนั้น
                        $("#context_search").html(""); //  ล้างค่าข้อความผลลัพธ์เดิม ถ้ามี
                });
                $("#translate_it").click(function(){  // เมื่อคลิกที่ปุ่มคำว่า แปล
                        $("#context_search").html(loading_img);
                        $.get("ajax_translate.php",{keyword:$.trim($("#keyword_q").val())
},function(data) {
                        $("#context_search").html(data); // แสดงผลลัพธ์จากการค้นหา
                        });
                });
                $("#keyword_q").keyup(function(event){
                        if(event.keyCode==13) { // เมื่อกดปุ่ม Enter ให้เริ่มการค้นหา
                                $("#context_search").html(loading_img);
        $.get("ajax_translate.php",{keyword:$.trim($("#keyword_q").val())},function(data){
```

```
                                        $("#context_search").html(data);
                                });
                        }
                });
                $("div#context_search a").live("click",function(){
                        var text_search=$.trim($(this).text());
                        $("#context_search").html(loading_img);
                        $("#keyword_q").val(text_search);
                        $.get("ajax_translate.php",{keyword:text_search},function(data){
                $("#context_search").html(data); // แสดงผลลัพธ์จากการค้นหา
                        });
                        return false;
                });
});
</script>
</body>
</html>
```



ภาพที่ 12.4 ตัวอย่างการสร้าง AJAX dictionary จาก longdo ด้วย jQuery

สรุป

Ajax เป็นเทคนิคใหม่ที่คนส่วนใหญ่กำลังเริ่มศึกษากัน โดยการนำ Ajax เข้ามาช่วยนั้นทำให้ชิ้นงานเว็บเพจดูมีความน่าสนใจมากยิ่งขึ้น สามารถทำงานร่วมกับภาษา PHP และ HTML ได้อย่างลงตัว อีกทั้งช่วยเพิ่มประสิทธิภาพในการแสดงผลหน้าเว็บเพจได้รวดเร็วขึ้น ไม่ต้องเสียเวลาหยุดรอการโหลดหรือการรีเฟรชหน้าจอ แต่ในการนำเทคนิคนี้มาใช้งานได้นั้นควรศึกษาและมีพื้นฐานภาษา JavaScript หลังจากนั้นจึงค่อยเริ่มศึกษา Ajax เนื้อหาในบทนี้เป็นเพียงแนะนำในเบื้องต้นเท่านั้น ยังไม่ครอบคลุมถึงคำสั่งทั้งหมด

คำถามท้ายบท

1. จงอธิบายหลักการทำงานของ AJAX
2. จงยกตัวอย่างอีเวนต์ที่เกี่ยวกับเมาส์ พร้อมอธิบาย
3. จงยกตัวอย่างอีเวนต์ที่เกี่ยวกับคีย์บอร์ด พร้อมอธิบาย
4. จงอธิบายสคริปต์คำสั่งต่อไปนี้ จะเกิดอีเวนต์เมื่อไหร่ และจะได้ผลลัพธ์อย่างไร

```
<div id="dv" onmousedown="if (confirm ('Are you 18+')) {
location.href='index.html'; } else { window.close ( ) }" > 18+ ? </div>
```

5. จงอธิบายความเหมือนและความแตกต่างระหว่าง JavaScript และ AJAX