**คู่มือสำหรับการสอน jQuery**

**jQuery** เป็นการบรรจุเอาคำสั่งต่างๆของ JavaScript Library มาสร้างเป็นฟังก์ชั่นใหม่ในรูปแบบของ Framework เพื่อให้ง่ายและเพิ่มประสิทธิภาพต่อการใช้งาน และไม่ต้องมาเขียนใหม่เองทั้งหมดตั้งแต่ต้น อย่างเช่น การเขียนโปรแกรมในรูปแบบ Ajax ก็สามารถทำได้แบบง่ายๆ เพียง code ไม่กี่บรรทัด หรือจะเขียนโปรแกรมดัก Event (เหตุการณ์) ต่างๆ เช่น การ click, rollover, mouse ก็เขียนได้ง่ายขึ้นและลดปัญหาความไม่เข้ากันของเบราเซอร์ที่ต่างกัน

**jQuery Course Outline**

**หลักสูตร jQuery ระยะเวลาการอบรม 16 ชั่วโมง**

**พื้นฐานผู้เข้าอบรม**

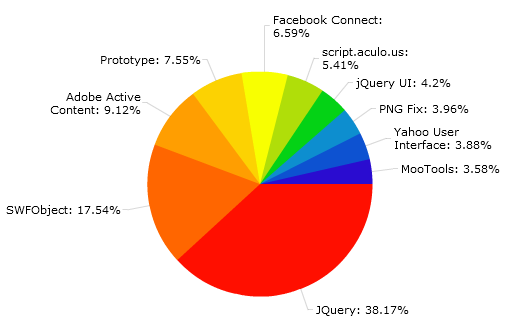
* **มีประสบการณ์ในการพัฒนาเว็บไซต์**
* **เขียนภาษา HTML ได้**
* **มีความเข้าใจ Cascading Style Sheets(css) เบื้องต้น**
* **อาจจะสามารถพัฒนา Web Application ด้วยภาษา PHP**

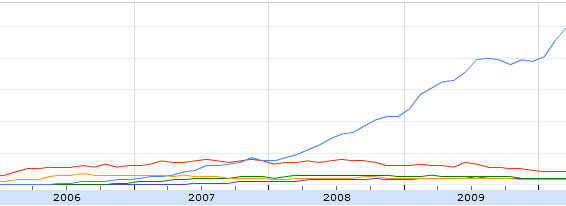
**เนื้อหาการอบรม**

* ทำความเข้าใจเกี่ยวกับ JavaScript Library
* รู้จักกับความสามารถของ jQuery
* รู้จักกับ XHTML และ DOM (Document Object Model)
* รู้จักกับ CSS
* การมองโค้ด XHTML ในรูปแบบของ DOM
* เรียนรู้พื้นฐาน JavaScript
* การใช้งาน jQuery เบื้องต้น
* ติดตั้ง firebug เพื่อเอาไว้ตรวจสอบ Error ของ jQuery
* โครงสร้างของ jQuery
* ความแตกต่างระหว่าง OnLoad Event กับ ready method
* ทำความเข้าใจเกี่ยวกับ Selector ในแบบต่าง ๆ
* ข้อพิจารณาการใช้งาน Selector
* Event ของ jQuery
* การเขียน Function กับ jQuery
* การเข้าถึงและจัดการข้อมูลในรูปแบบของ DOM ด้วย jQuery
* jQuery กับการจัดการ CSS
* เอฟเฟ็กต์(Effects) การสร้างแอนิเมชั่นด้วย jQuery
* Ajax การรับส่งค่าทั้งในแบบ GET and POST
* jQuery with XML
* jQuery with json
* การพัฒนา Web Application ในรูปแบบ AJAX ด้วย jQuery
* jQuery , PHP , Mysql , Ajax (Select, Update, Insert, Delete)
* พัฒนา Plugin ของ jQuery ไว้ใช้งาน
* การใช้งาน Plugin
* การใช้งาน jQuery Ui
* การอบรมเป็นแบบ WorkShop คือ ผู้สอนและผู้เรียนเน้นการปฏิบัติเป็นสำคัญ!

1. การเขียนการตรวจสอบข้อมูลของ Input Form
2. การสร้าง Gallery รูปภาพ
3. การสร้าง Search Auto complete (jquery UI)
4. การสร้าง Slide News (jquery UI)
5. การแบ่งหน้าแสดงผล (jquery UI)
6. การเรียงลำดับข้อมูลในตาราง (jquery UI)
7. การสร้างตะกร้าสินค้า

**ทำความเข้าใจเกี่ยวกับ JavaScript Library**

****

****

Javascript library คือ การนำภาษา javascript มาเขียนเป็นฟังก์ชั่นใหม่ให้ใช้งานง่ายขึ้นมีประสิทธิภาพมากขึ้นเรียกว่า API (Application Programming Interface) เป็นฟังก์ชั่นที่เอาไว้ให้ผู้พัฒนาสามารถเรียกใช้ได้เลยโดยไม่ต้องเขียนเองทั้งหมด็้นgdd JavaScript library มีหลากหลายค่ายมากมายยกตัวอย่างเช่น [Prototype](http://en.wikipedia.org/wiki/Prototype_JavaScript_Framework), [MooTools](http://en.wikipedia.org/wiki/MooTools), [jQuery](http://en.wikipedia.org/wiki/JQuery), [Dojo](http://en.wikipedia.org/wiki/Dojo_Toolkit), [YUI](http://en.wikipedia.org/wiki/YUI_Library) โดยแต่ละค่ายก็จะมีข้อดีข้อเสียต่างกัน javascript library ที่มาแรงและมีคนใช้กันมากที่สุดในขณะนี้คือ javascript library jQuery ซึ่งมีแนวโน้มการใช้งานสูงขึ้นเรื่อยๆ เมื่อเทียบกับ javascript library อื่นๆ

jQuery JavaScript library ก็คือการนำภาษา JavaScript มาสร้างเป็นฟังก์ชั่นใหม่ในรูปแบบของ framework เพื่อให้ง่ายต่อการใช้งาน และไม่ต้องมาเขียนใหม่เองทั้งหมดตั้งแต่ต้น อย่างเช่นการเขียนโปรแกรมในรูปแบบ ajax ก็สามารถทำได้แบบง่ายๆ เพียง code ไม่กี่บรรทัด หรือจะเขียนโปรแกรมดัก Event (เหตุการณ์) ต่างๆ เช่น การ click, rollover, mouse ก็เขียนได้ง่ายขึ้นและลดปัญหาความไม่เข้ากันของบราวเซอร์ที่ต่างกัน

**รู้จักกับความสามารถของ jQuery**

การนำ jQuery มาประยุกต์ใช้งานนั้นมีหลากหลายรูปแบบโดยส่วนมากจะเน้นการใช้งาน ในส่วนของ client side User interface ที่ให้ user สามารถใช้งานเว็บไชต์ได้ง่ายยิ่งขึ้นและเร็วขึ้น และยังทำให้ผู้เขียนโปรแกรมก็สามารถเขียนโปรแกรมยากๆ ได้ง่ายขึ้นเพราะ jqueryได้รวบรวม library ที่สำคัญต่างๆ ที่มีความซับซ้อนรวมเข้าด้วยกันและจัดการผูกฟังก์ชั่นต่างๆ ของ JavaScript ให้สามารถทำงานได้อย่างมีประสิทธิภาพมากกว่าเดิม ความสามารถที่สำคัญในไลบรารี jQuery มีดังต่อไปนี้

**1. เข้าถึงอิลิเมนต์ในเอกสาร (Selector)**

jQuery มีกลไกลในการทำงานของซีเลกเตอร์ที่มีประสิทธิภาพในการดึงข้อมูลจากเอกสาร โดยดึงเอาความสามารถในการเข้าถึงอิลิเมนต์ของ CSS มาจัดการ จึงทำให้ง่ายในการจัดการกับอิลิเมนต์ต่างๆ มากกว่าการใช้ JavaScript แบบเดิม

**2. การแสดงผลของเว็บ (Cross Browser)**

jQuery สามารถแก้ปัญหาของการแสดงผลบนเบราเซอร์ที่ต่างกัน ให้แสดงผลที่ถูกต้อง(ความไม่เข้ากันของบราวเซอร์เป็นปัญหาใหญ่มากสำหรับการนักพัฒนาเว็บไชต์เพราะแต่ละบราวเซอร์ก็ใช้มาตรฐานของตัวเองในการพัฒนา)

**3. แก้ไขคอนเท็นต์ของเอกสาร (DOM)**

jQuery สามารถเปลี่ยนข้อความ, แทรกรูปภาพ, เรียงลำดับลิสใหม่ หรือสามารถแก้โครงสร้าง html ใหม่ทั้งหมด หรือต่อเติมจากโครงสร้างเดิมได้อย่างง่ายๆ

**4. โต้ตอบกับการทำงานของผู้ใช้ (Event)**

jQuery มีวิธีการดักจับอีเวนต์ที่หลากหลายเช่น อีเวนต์คลิก, อีเวนต์ดับเบิลคลิก สามารถจัดการฟังก์ชั่นดักจับอีเวนต์ต่างๆ อย่างเป็นระเบียนมากขึ้น

**5. แสดงภาพเคลื่อนไหวให้เอกสาร (Effect)**

jQuery มีไลบรารีในการจัดการเรื่องเอฟแฟ็กต์ไว้อย่างสมบรูณ์ เช่น การกำหนดอิลิเมนต์ให้ภาพจางๆ แล้วค่อยๆ ชัดขึ้นเป็นต้น

**6. ดึงข้อมูลจากเชิร์ฟเวอร์โดยไม่ต้องรีเฟรช (AJax)**

รูปแบบการเขียนโค้ดแบบนี้เป็นที่รู้จักกันดีในชื่อเรียกว่า Asynchronous JavaScript And XML (AJAX) ซึ่งช่วยให้นักพัฒนาเว็บไชต์ สร้างเว็บไชต์ที่เต็มไปด้วยคุณลักษณะที่โดดเด่น ในการโต้ตอบกับผู้ใช้ได้อย่างรวดเร็ว มีประสิทธาภาพ เขียนง่าย ไม่มีการ Refresh โดยไลบรารีได้ขจัดความยุ่งยากในกระบวนการทำงานนี้ และความแตกต่างของแต่ละเบราร์เซอร์ให้หมดไป

**รู้จักกับ HTML และ DOM (Document Object Model)**

**HTML**

## HTML คือภาษาสำหรับออกแบบหน้าเว็บไซต์

+ HTML ย่อมาจาก Hyper Text Markup Language  
+ HTML ไม่ใช่ภาษาที่ใช้ในการเขียนโปรแกรม แต่เป็นภาษาที่ใช้ตบแต่ง (Markup Language)  
+ Markup Language คือชุดของแท็ก (Tags) เช่น <html> , <p> เป็นต้น

**คำสั่งเริ่มต้น**  
**รูปแบบ**

|  |
| --- |
| **<HTML>.....</HTML>** |

คำสั่ง <HTML> เป็นคำสั่งเริ่มต้นในการเขียนโปรแกรม และ </HTML>เป็นคำสั่งจุดสิ้นสุดโปรแกรม

**คำสั่งการทำหมายเหตุ (Comment)  
รูปแบบ**

|  |
| --- |
| **<!-- ..... -->** |

**ตัวอย่าง**

|  |
| --- |
| **<!-- END WEBSTAT CODE -->** |

ข้อความที่อยู่ในคำสั่งจะปรากฏอยู่ในโปรแกรมแต่ไม่ถูกแสดงบนจอภาพ  
**ส่วนหัว**   
**รูปแบบ**

|  |
| --- |
| **<HEAD>.....</HEAD>** |

ใช้กำหนดข้อความ ในส่วนที่เป็น ชื่อเรื่อง ภายในคำสั่งนี้ จะมีคำสั่งย่อย อีกหนึ่งคำสั่ง คือ <TITLE>

**กำหนดข้อความในไตเติลบาร์   
รูปแบบ**

|  |
| --- |
| **<TITLE>.....</TITLE>** |

**ตัวอย่าง**

|  |
| --- |
| **<TITLE> บทเรียน HTML </TITLE>** |

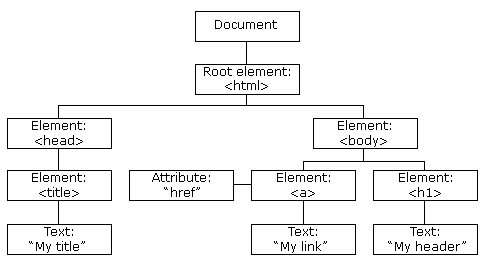
เป็นส่วนแสดงชื่อของเอกสาร จะปรากฏ ขณะที่ไฟล์ HTML ทำงานอยู่ ข้อความ ที่กำหนดในส่วนนี้ จะไม่ถูกนำไปแสดง ผลของ เว็บเบราเซอร์แต่จะปรากฏในส่วนของไตเติลบาร์ (Title bar) ที่เป็นชื่อของวินโดว์ข้างบน ไม่ควรให้ยาวเกินไป เพียงให้รู้ว่าเว็บเพจที่กำลังใช้งานอยู่เกี่ยวข้องกับอะไร

**ส่วนของเนื้อหา  
รูปแบบ**

|  |
| --- |
| **<BODY>.....</BODY>** |

ส่วนเนื้อหาของโปรแกรมจะเริ่มต้นด้วย คำสั่ง <BODY> และจบลงด้วย </BODY> ภายในคำสั่งนี้คือ ส่วนที่จะแสดงทางจอภาพ

**อธิบาย Document Object Model(DOM)**



DOM ก็คือการประกาศ object เพื่อให้สามารถเข้าถึงได้ (เช่นเราเขียน JavaScript เพื่อเรียก object ต่างๆ ที่อยู่หน้าเว็บ) จากหน้าเว็บนี้ ถ้าเราเอามาเขียนผังโครงสร้างแบบ tree เพื่อใช้ในการอธิบาย DOM เราก็จะได้ออกมาหน้าตาดังรูปด้านบน จากรูปเราจะเห็นชัดว่าใครอยู่เหนือใคร ใครเป็น parent ใครเป็น child และจากพื้นฐานของ DOM ที่ทำให้เราแยกแยะ object ออกมาได้แล้วนั้น จะทำให้เวลาที่เราจะ access เข้าถึงแต่ละ object เราก็จะทำได้อย่างง่ายขึ้นนั่นเอง

เช่น ซ้ายมือจะเห็นว่า title เป็น child ของ head และกลับกัน head เป็น parent ของ childโดย DOM นี้จะถูกประกาศเมื่อหน้าเว็บโหลดเสร็จแล้วเท่านั้นครับและ element ใดที่ไม่ปรากฏเมื่อตอนที่หน้าเว็บโหลดเสร็จแล้ว ก็จะไม่มีใน DOM ด้วยเช่นกัน (ส่วนที่สร้างโดยใช้ JavaScript)

**jQuery มีความสามารถในการเข้าถึง** element ใน DOM ซึ่งเป็นความสามารถที่มีประสิทธิภาพอย่างหนึ่งของ jQuery โดยนำความสามารถของซีเลกเตอร์ Cascading Style Sheets(CSS) เพื่อช่วยให้เข้าถึง element ของ DOM

**อธิบายโปรแกรมดักจับข้อผิดพลาด (Error) ในการเขียนโปรแกรม JavaScript**

**Firebug คืออะไร?**

Firebug เป็นส่วนเสริมของ Firefox Browser ที่สามารถใช้เป็นเครื่องมือที่ช่วยในการปรับแต่ง Blogger Layout ได้เป็นอย่างดี เพราะเป็นเครื่องมือที่ทำให้เราทดลองปรับแต่งโค้ดต่างๆ ของหน้าเว็บที่เราต้องการ โดยที่การทดลองนั้นเป็นเพียงภาพสมมติ จะไม่ส่งผลเสีย หรือเกิดการเปลี่ยนแปลงใด ๆ ต่อหน้าเว็บจริง

**Firebug ทำงานอย่างไร?**

หลักการทำงานของ Firebug คือ เมื่อเราติดตั้ง Firebug แล้วเปิด Firefox Browser ก็สามารถเรียกใช้งาน Firebug กับหน้าเว็บที่ต้องการได้เลย ซึ่งมุมมองในการแก้ไข และใช้งาน ก็เหมือนกับเราเปิดดูหน้าเว็บบน Firefox Browser ตามปกติ เมื่อเราทดลองแก้โค้ด CSS หรือ HTML หรือ จาวาสคริปต์ของหน้าเว็บนั้นๆ เราก็จะเห็นผลลัพธ์จากการปรับแต่งทันที  แต่ผลการปรับแต่งจะไม่ได้ถูกบันทึกเก็บไว้แต่อย่างใด ดังนั้นเมื่อเรา refresh หน้าเว็บนั้นใหม่ สิ่งที่เราได้ทดลองปรับแต่งก็จะกลับคืนสู่สภาพเดิมทันทีเพราะฉะนั้นหลักการทำงานของ Firebug คือใช้ช่วยในการทดลอง ปรับแต่งหน้าเว็บ ไม่ใช่เครื่องมือปรับแต่งโค้ดของเว็บ หรือบล็อกโดยตรง

**Firebug ช่วยปรับแต่ง Blogger ได้อย่างไร?**

ด้วยหลักการตามที่กล่าวมาแล้วถ้าเราอยากจะปรับความกว้าง ปรับ CSS หรืออยากจะลบส่วนที่ไม่ต้องการออกจากหน้าบล็อกเราก็สามารถใช้เครื่องมือนี้ ค้นหาโค้ดเหล่านั้นได้ และเมื่อปรับแต่งลงตัวแล้วก็คัดลอกวิธีปรับแต่งนั้นไปปรับแต่งโค้ดใน Blogger อีกทีโดยไม่ต้องแก้ไปแก้มา เพราะโค้ดที่เรานำมาแก้นั้นเป็นโค้ดที่เราได้ทดลองจนเป็นที่พอใจแล้ว

**จะใช้ Firebug ต้องมีอะไรบ้าง?**

การจะใช้เครื่องมือนี้ได้นั้นจะต้องติดตั้ง Firefox Browser ดังนั้นหากท่านใดจะใช้เครื่องมือนี้ก็จะต้องติดตั้ง Firefox Browser บนคอมพิวเตอร์ก่อน โดยสามารถเข้าไปดาวน์โหลดและติดตั้งจากลิงค์ด้านล่างเลย

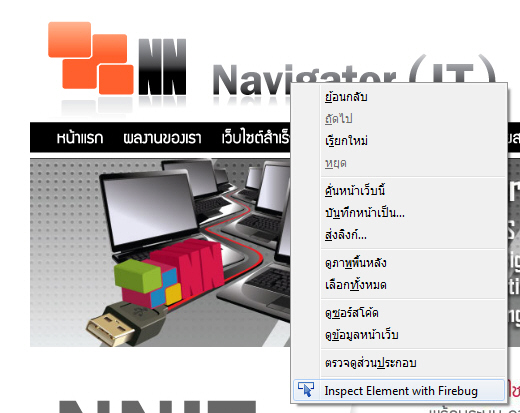
[http://www.mozilla.com/th/](http://www.mozilla.com/th/)

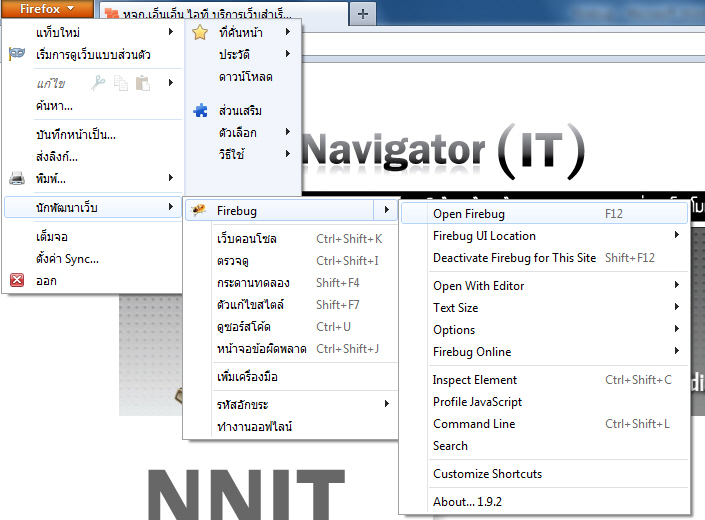
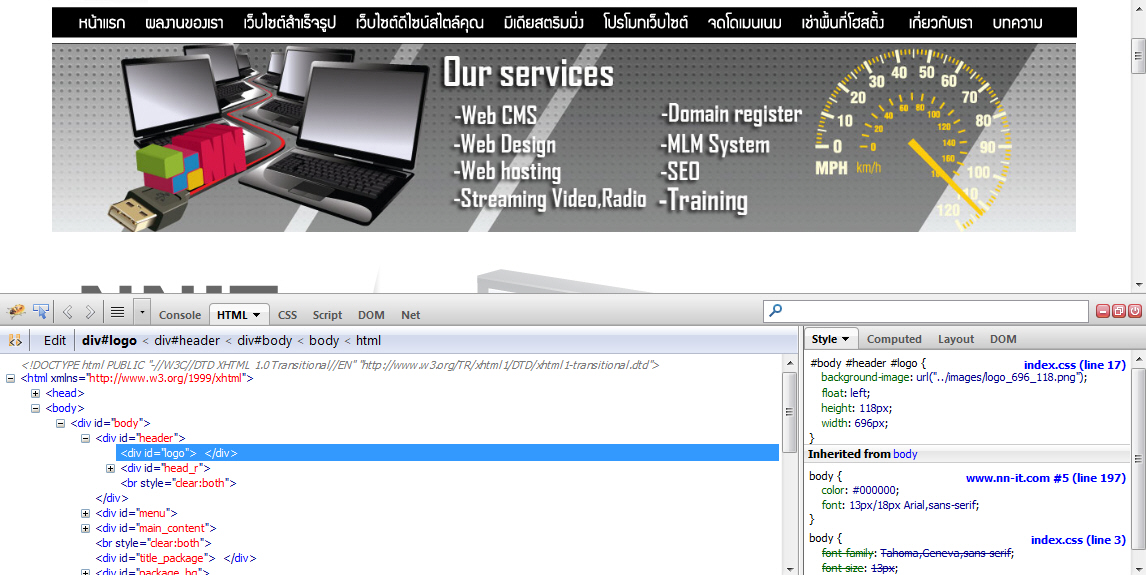
**วิธีติดตั้ง Firebug**

เมื่อติดตั้ง Firefox Browser ที่คอมพิวเตอร์แล้วให้ไปที่

<https://addons.mozilla.org/en-US/firefox/addon/1843/> และทำการติดตั้ง Firebug ลงบน Firefox  
**การเรียกใช้งาน Firebug ?**

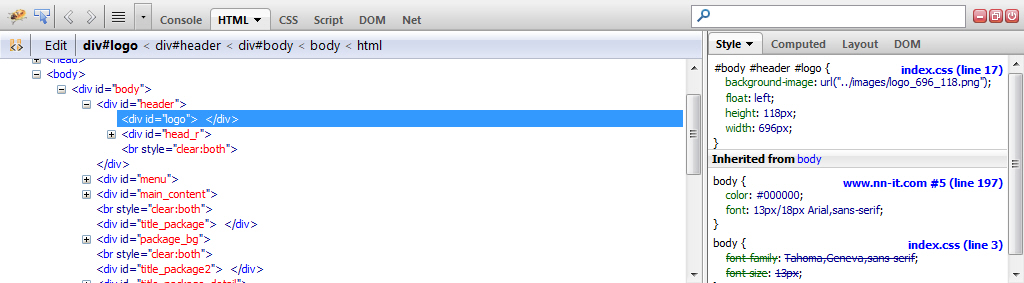
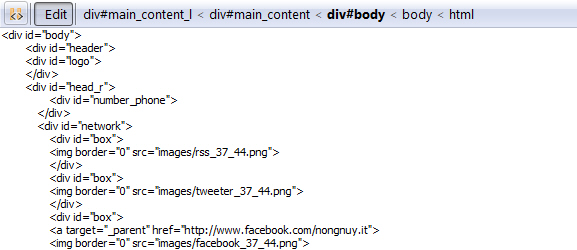
หลังจากติดตั้ง Firebug ให้เป็นส่วนเสริมของ Firefox Browser เรียบร้อยแล้ว การเรียกใช้ทำได้ 3 วิธีคือ  
1. เรียกใช้โดยการ **กด F12 บน Keyboard**

2. เรียกใช้โดยการคลิกขวาและเลือก สัญลักษณ์ Inspect Element

3. เรียกใช้จากเมนูส่วนเสริม   
  
  
และไม่ว่าจะเรียกใช้ด้วยวิธีใดก็ตามแถบเครื่องมือของ Firebug จะปรากฎอยู่ที่ด้านล่างของ Browser ดังรูปด้านล่าง  
  


**เครื่องมือของ Firebug**

**ก่อนดูตัวอย่างการใช้ควรรู้จักแถบเครื่องมือของ Firebug ดังนี้**

  
**ส่วนที่ 1**  เป็นเครื่องมือที่ใช้ระบุตำแหน่งที่เราต้องการปรับแก้ หรือระบุตำแหน่งที่เราต้องการทราบโค้ด ใช้งานโดยคลิกที่ [2010-06-28_123055](http://lh3.ggpht.com/_uJ9ptu1GqQ0/TChJg-7jWvI/AAAAAAAAEBc/XP5sFXHEt-g/s1600-h/2010-06-28_123055%5b3%5d.png)และนำไปคลิกยังเป้าหมายบนหน้าเว็บที่ต้องการปรับแต่งอีก 1 ครั้ง   
**ส่วนที่ 2**  เมื่อใช้เครื่องมือในส่วนที่ 1 แล้วเราสามารถเลือกได้ว่าเราต้องการแสดงโค้ดชนิดใด เช่น HTML หรือ CSS หรือ จาวาสคริปต์เป็นต้น ซึ่งโดยค่าดังเดิมก็จะแสดงผลเป็น HTML โค้ด   
**ส่วนที่ 3** เป็นส่วนที่ใช้ทดลองปรับแต่ง HTML โค้ด โดยการคลิกที่ข้อความ EDIT ทางซ้ายสุด หรือคลิกที่ Class หรือ id ที่ต้องการ ตามที่แสดงผลบน bar ในส่วนที่ 3 ก็ได้  


**6**

**5**

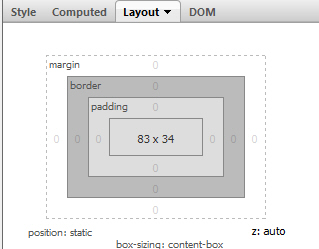
**4**

**3**

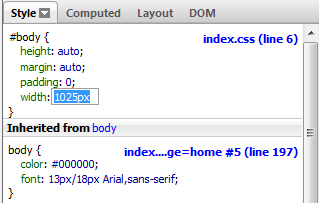
**2**

**1**

**ส่วนที่ 4**  เป็นส่วนที่ใช้ในการแสดง HTML โค้ดที่เราต้องการปรับแต่ง สามารถค้นหา ชื่อของ Class หรือ id ที่ต้องการ จากการแสดงผลในส่วนนี้ได้และนำชื่อ Class หรือ id ไปปรับแต่งแม่แบบได้ง่ายขึ้นด้วย  
[ทำบล็อก แต่งบล็อก](http://lh3.ggpht.com/_uJ9ptu1GqQ0/TChJj3jPrBI/AAAAAAAAEBs/SIsFTJXsF8E/s1600-h/2010-06-28_131343%5b18%5d.png)  
**ส่วนที่ 5** เป็นตัวเลือกในการแสดงรายละเอียดของโค้ดในส่วนที่ 4 เช่นแสดง CSS แสดง Layout และยังสามารถทดลองปรับแต่งได้จากส่วนนี้ด้วย



**ส่วนที่ 6** เป็นส่วนที่แสดงผลสืบเนื่องจากส่วนที่ 5 สามารถทดลองปรับแต่งโค้ดในส่วนนี้ได้



**ตัวอย่างการใช้งาน Firebug**  
ในตัวอย่างต้องการเพิ่มความสูงของ Logo ในหน้าแรกก็นำเมาส์คลิกที่ Inspect Element ไอคอน  [2010-06-28_123055](http://lh3.ggpht.com/_uJ9ptu1GqQ0/TChJg-7jWvI/AAAAAAAAEBc/XP5sFXHEt-g/s1600-h/2010-06-28_123055%5b3%5d.png)และนำไปคลิกที่ส่วน Logo ดังรูป

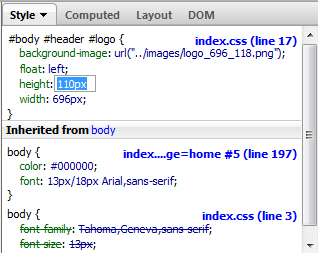


**3**

**2**

**1**

เมื่อปรากฏโค้ดของส่วนที่ต้องการแก้แล้ว ก็ลงมือทดลองปรับแต่งโดยใช้เครื่องมือที่ได้อธิบายการทำงานไปแล้ว เช่น ในตัวอย่างต้องการแก้ความสูงให้น้อยกว่าเดิม จากเดิมที่เห็นในรูปคือ height: 118px; โดยจะลดเป็น height: 110px;  ก็แก้ CSS ทางขวามือ



ผลที่ได้จากการแก้ไขก็จะแสดงให้เราเห็นทันที

**118 px**

**110 px**



ทดลองปรับและดูผลลัพธ์ว่าจะพอดีกับแม่แบบหรือสวยงามหรือไม่ ถ้าไม่ก็ปรับแต่งไปเรื่อยๆ จนเป็นที่พอใจและจดจำโค้ดที่แก้ไขแล้วไปใช้แก้ไขจริง ๆ ในแม่แบบ

**รู้จักกับ CSS**

****

**CSS (Cascading Style Sheets)** คือ ชุดคำสั่งที่ใช้สำหรับการกำหนดการแสดงผลข้อมูลหน้าเว็บเพจ เป็นมาตราฐานหนึ่งของ W3C ที่กำหนดขึ้นมา เพื่อใช้ในการจัดการรูปแบบเอกสารเว็บเพจโดยเฉพาะ การใช้งาน CSS จะเข้ามาช่วยเพิ่มความสามารถให้กับ HTML เดิมที่เราใช้งานกันอยู่ในปัจจุบัน

**ประโยชน์ของ CSS**  
-  ลดขนาดแฟ้มเอกสารเว็บไซต์ ทำให้โหลดเว็บไซต์ได้ไว  
-  ลดเวลาในการออกแบบพัฒนาเว็บ เพราะสามารถนำ CSS ไปใช้กับเอกสารเว็บอื่นๆ ได้ง่าย สะดวก  
-  ลดเวลาในการปรับแก้ไขเอกสารเว็บ  
-  ทำงานได้ดีกับเทคโนโลยีใหม่ๆ เช่น การแสดงผลบนเบราว์เซอร์โมบายต่างๆ

**External CSS**

**เป็นการเรียกใช้ไฟล์ CSS จากภายนอกเข้ามาใช้งานภายในเพจที่ต้องการ**

|  |
| --- |
| **<link href="file\_css.css" rel="stylesheet" type="text/css">** |

โดยระบุไว้ใน head element

**Internal CSS**

การเรียกใช้งานไฟล์ CSS ภายในเพจเดียวกันสามารถเรียกใช้ได้ดังนี้

|  |
| --- |
| **<style type=”text/css”>**  **/\*Code define CSS\*/**  **</style>** |

โดยระบุไว้ใน head element

**รูปแบบคำสั่งของ CSS**

|  |
| --- |
| **Selector { property : value ; property : value ; }** |

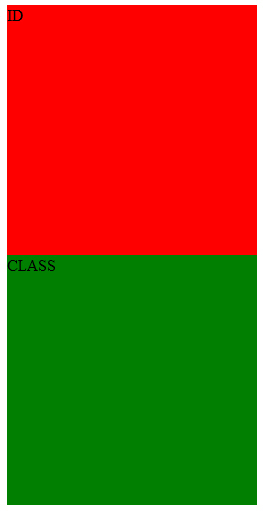
รูปแบบการใส่ Comment

|  |
| --- |
| **/\* Comment \*/** |

**ตัวอย่างการเรียกใช้งาน CSS จากภายใน**

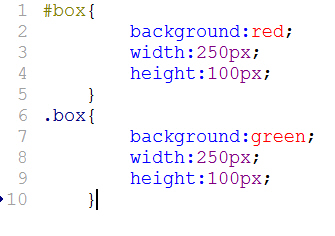


**ผลลัพธ์**

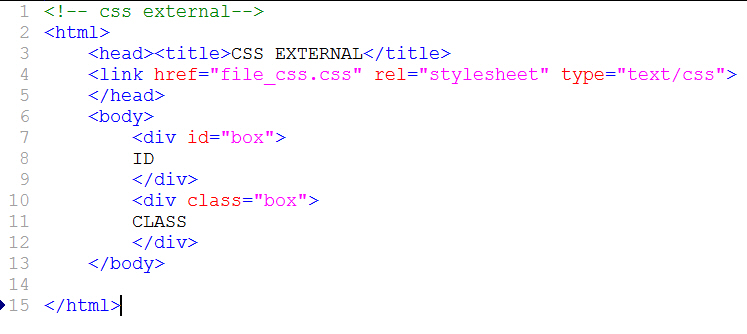


**ตัวอย่างการเรียกใช้งาน CSS จากภายนอก**

ไฟล์ file\_css.css

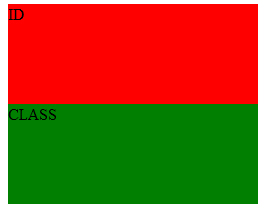


ไฟล์ css\_external.html



rel = เป็นตัวระบุความสำพันธ์ระหว่างหน้าเพจปัจจุบันกับหน้าเพจที่ลิงค์มาจากข้างนอก

**ผลลัพธ์**

****

**เรียนรู้พื้นฐาน JavaScript**

**Java Script คืออะไร?**

Java Script เป็นภาษาโปรแกรม (programming language) ประเภทหนึ่ง ที่เรียกกันว่า "สคริปต์" (script) ซึ่งมีวิธีการทำงานในลักษณะ "แปลความและดำเนินงานไปทีละคำสั่ง" (interpret) ภาษานี้เดิมมีชื่อว่า LiveScript ได้รับการพัฒนาขึ้นโดย Netscape ด้วยวัตถุประสงค์ เพื่อที่จะช่วยให้เว็บเพจสามารถแสดงเนื้อหา ที่มีการเปลี่ยนแปลงไปได้ ตามเงื่อนไขหรือสภาพแวดล้อมต่างๆ กัน หรือสามารถโต้ตอบกับผู้ชมได้มากขึ้น ทั้งนี้เพราะภาษา HTML แต่เดิมนั้น เหมาะสำหรับใช้แสดงเอกสาร ที่มีเนื้อหาคงที่แน่นอน และไม่มีลูกเล่นอะไรมากมายนัก  
 เนื่องจาก JavaScript ช่วยให้ผู้พัฒนาสามารถสร้างเว็บเพจได้ตรงกับความต้องการ และมีความน่าสนใจมากขึ้น ประกอบกับเป็นภาษาเปิด ที่ใครก็สามารถนำไปใช้ได้ ดังนั้นจึงได้รับความนิยมเป็นอย่างสูง มีการใช้งานอย่างกว้างขวาง รวมทั้งได้ถูกกำหนดให้เป็นมาตรฐานโดย ECMA ซึ่งเราจะพบว่าปัจจุบัน บราวเซอร์ส่วนใหญ่จะรองรับ JavaScript กันหมดแล้ว  
 การทำงานของ JavaScript จะต้องมีการแปลความคำสั่ง ซึ่งขั้นตอนนี้จะถูกจัดการโดยบราวเซอร์ดังนั้น JavaScript จึงสามารถทำงานได้ เฉพาะบนบราวเซอร์ที่สนับสนุน ซึ่งปัจจุบันบราวเซอร์เกือบทั้งหมดก็สนับสนุน JavaScript แล้ว อย่างไรก็ดี สิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชั่นใหม่ๆ ออกมาด้วย (ปัจจุบันคือรุ่น 1.5) ดังนั้น ถ้านำโค้ดของเวอร์ชั่นใหม่ ไปรันบนบราวเซอร์รุ่นเก่าที่ยังไม่สนับสนุน ก็อาจจะทำให้เกิด error ได้

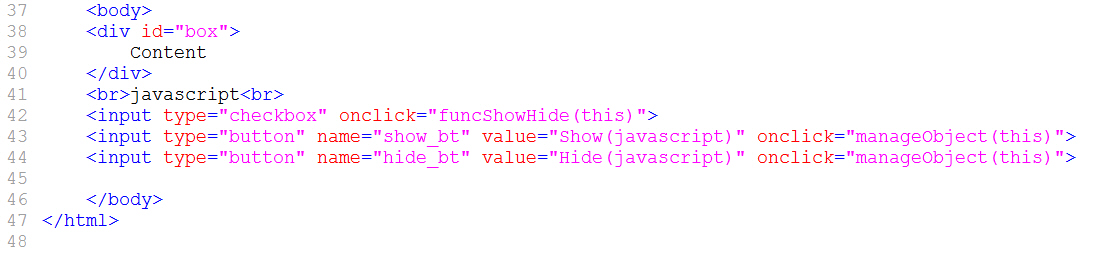
การทำงานของ JavaScript เกิดขึ้นบนบราวเซอร์ (เรียกว่าเป็น client-side script) ดังนั้นไม่ว่าจะใช้เซิร์ฟเวอร์อะไร หรือที่ไหน ก็ยังคงสามารถใช้ JavaScript ในเว็บเพจได้ ต่างกับภาษาสคริปต์อื่น เช่น Perl, PHP หรือ ASP ซึ่งต้องแปลความและทำงานที่ตัวเครื่องเซิร์ฟเวอร์ (เรียกว่า server-side script) ดังนั้นจึงต้องใช้บนเซิร์ฟเวอร์ ที่สนับสนุนภาษาเหล่านี้เท่านั้น อย่างไรก็ดี จากลักษณะดังกล่าวก็ทำให้ JavaScript มีข้อจำกัด คือไม่สามารถรับและส่งข้อมูลต่างๆ กับเซิร์ฟเวอร์โดยตรง เช่น การอ่านไฟล์จากเซิร์ฟเวอร์ เพื่อนำมาแสดงบนเว็บเพจ หรือรับข้อมูลจากผู้ชม เพื่อนำไปเก็บบนเซิร์ฟเวอร์ เป็นต้น ดังนั้นงานลักษณะนี้ จึงยังคงต้องอาศัยภาษา server-side script อยู่ (ความจริง JavaScript ที่ทำงานบนเซิร์ฟเวอร์ก็มี ซึ่งต้องอาศัยเซิร์ฟเวอร์ที่สนับสนุนโดยเฉพาะเช่นกัน แต่ไม่เป็นที่นิยมนัก)

การทำงานของ JavaScript จะมีประสิทธิภาพมาก เพราะมันสามารถดัดแปลงคุณสมบัติ ขององค์ประกอบต่างๆ บนเว็บเพจ (เช่น สี หรือรูปแบบของข้อความ) และสามารถรับรู้เหตุการณ์ ที่ผู้ชมเว็บเพจโต้ตอบกับองค์ประกอบเหล่านั้น (เช่น การคลิก หรือเลื่อนเมาส์ไปวาง) ได้ ดังนั้นจากภาษา HTML เดิม ที่มีลักษณะเนื้อหาแบบคงที่ (static) ใน HTML เวอร์ชั่นใหม่ๆ จึงได้มีการพัฒนาให้มีคุณสมบัติบางอย่างเพิ่มขึ้น และมีลักษณะเป็นอ็อบเจ็ค "object" มากขึ้น การทำงานร่วมกันระหว่างคุณสมบัติใหม่ของ HTML ร่วมกับ JavaScript นี้เอง ทำให้เกิดเป็นสิ่งที่เรียกว่า Dynamic HTML คือภาษา HTML ที่สามารถใช้สร้างเว็บเพจที่มีลักษณะพลวัต (dynamic) ได้นั่นเอง

นอกจากนี้ อีกองค์ประกอบหนึ่งที่เกี่ยวข้อง ก็คือ Cascading Style Sheet (CSS) ซึ่งเป็นภาษาที่ช่วยให้เราควบคุมรูปแบบ ขององค์ประกอบต่างๆ บนเว็บเพจ ได้อย่างมีประสิทธิภาพมากกว่าคำสั่ง หรือแท็ก -(tag) ปกติของ HTML เนื่องจาก JavaScript สามารถดัดแปลงคุณสมบัติของ CSS ได้เช่นกัน ดังนั้นมันจึงช่วยให้เราควบคุมเว็บเพจ ได้อย่างมีประสิทธิภาพมากมากยิ่งขึ้นไปอีก

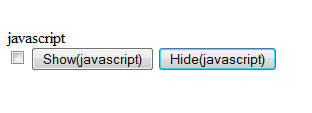
**ตัวอย่างการเขียนโปรแกรม Hide Show Object ด้วย javasript**

****

****

**ผลลัพธ์**

เมื่อกดปุ่ม Hide (JavaScript) ก็จะเป็นการซ่อนกล่องสีแดงที่เราได้เขียนโปรแกรมไว้



เมื่อกดปุ่ม Show (JavaScript)ก็จะเป็นการแสดงกล่องสีแดง



**การใช้งาน jQuery เบื้องต้น**

สำหรับเว็บไซต์ในการดาวน์โหลด ไฟล์ jquery.js ได้ที่ [http://jquery.com](http://jquery.com/)

การเรียกใช้งาน jquery.js ก็เหมือนเป็นการเรียกใช้งาน ไฟล์ JavaScript ทั่วไปถ้าเครื่องคอมพิวเตอร์ต่ออินเตอร์เน็ตอยู่สามารถเรียกใช้ไฟล์ jquery.js ได้โดยเรียก **Content Delivery Network or Content Distribution Network (CDN)** ที่ให้บริการ JavaScript library jQuery ต่างๆ ออนไลน์ เช่น เว็บของ jQuery(<http://code.jquery.com/jquery-1.7.1.min.js>) หรือ google (<http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js>) การเรียกใช้แบบออนไลน์จะมีข้อดีตรงที่ความเร็วในการใช้งานมากขึ้นลดแบนวิดในการดึงข้อมูลพร้อมกัน เพราะ server จะมีหลายเครื่องให้บริการและมีการ updateอยู่ในเวอร์ชั่นที่ใหม่ตลอด หรือจะใช้งานออฟไลน์ก็สามารถโหลดมาเก็บไว้ที่เครื่องแล้วก็เรียกใช้งานตามพาททีเก็บไฟล์ JavaScript library jQuery

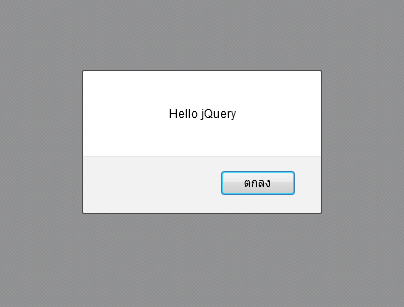
ในการใช้งาน jquery เราไม่จำเป็นต้องติดตั้งโปรแกรมอะไรเลย แค่นำไฟล์ jquery ไปวางไว้ในพื้นที่ ที่สามารถอ้างอิงได้เนื่องจาก JavaScript เป็นภาษาแบบ interpreted (เป็นภาษาที่ใช้ตัวแปรภาษาอ่านคำสั่งทีละคำสั่ง) ไม่มีขั้นตอนการคอมไพล์ หรือขั้นตอนการ build

**ทดสอบเขียนโปรแกรมเพื่อตรวจสอบว่าสามารถใช้งาน jQuery ได้หรือไม่**

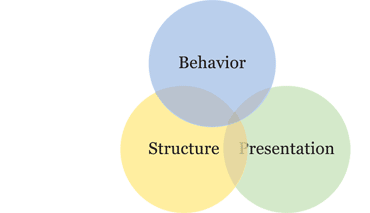
**HELLO JQUERY**

****

**ผลลัพธ์**

****

**โครงสร้างของ jQuery**



**อธิบาย ฟังก์ชั่น $()**

ฟังก์ชั่น $() เป็นฟังก์ชั่นเริ่มต้นของ jQuery ต้องประกาศฟังก์ชั่น $() ทุกครั้งที่เรียกใช้งาน jQuery  $()เป็นฟังก์ชั่นในการสร้าง object jQuery โดยปกติจะเรียกใช้งานฟังก์ชั่นนี้ได้ 2 รูปแบบ คือ $() หรือ $(document) เพราะภายใน object jQuery จะห่อหุ้มด้วย อิลิเมนต์ Dom ของเอกสารนั้นๆ เพื่อที่จะจัดการกับอิลิเมนต์ได้ เมื่อเราประกาศฟังก์ชั่น $(document) ในหน้าเอกสาร html ก็จะทำการห่อหุ้มอิลิเมนต์ Dom ต่างๆ ทั้งหมด เช่น

**ตัวอย่างโค้ด**

|  |
| --- |
| <div>,<h1>,<body>,<input>,<p> |

เพื่อที่จะใช้ในการจัดการกับ element เหล่านี้ได้ง่ายขึ้น เช่น การเพิ่ม css ให้กับ element การใส่ effect การเพิ่ม element การลบ element รวมถึงการอ่านค่าต่างของอิลิเมนต์ได้ง่ายๆ โดยไม่ต้องใช้ ฟังก์ชั่น วนลูปในการเขียนโค้ดยาวๆ เพราะ jQuery มีคุณสมบัติ implicit iteration มีฟังก์การวนลูปในตัว jQuery ไม่ต้องเขียนโค้ดเอง

**การประการฟังก์ชั่นแบบไม่ระบุชื่อ** **ปกติจะเรียกฟังก์ดังนี้**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | function runhighlight(){              $("#content").addClass("highlight");    }    $(document).ready(runhighlight); | | | | |
|  |  |

jQuery สามารถเขียนโด้ดให้สั้นลงได้โดยใช้ ฟังก์ชั่น **แบบไม่ระบุชื่อ (anonymous function หรือ lamba function)**

**สามารถเขียนโค้ดให้สั้นลงได้ดังนี้**

|  |  |  |  |
| --- | --- | --- | --- |
| $(document).ready(function(){    $(‘#content’).addClass(‘highlight’);  }); | | | |
|  |  |
|  |  |

เราจะใช้ฟังก์ชั่นนี้ในกรณีที่ เป็นฟังก์ชั่นที่ไม่ค่อยได้นำกลับมาใช้ใหม่

**ความแตกต่างระหว่าง OnLoad Event กับ ready method**

เมื่อต้องการให้ ฟังก์ชั่น ทำงานทันทีที่โหลดเพจเสร็จสามารถทำได้ 2 วิธีคือ

**วิธีที่ 1** รูปแบบเก่าจะใส่ ฟังก์ชั่น onload ใน tag body เช่น

|  |
| --- |
| <body onload=” runhighlight();”> |

และมีฟังก์ชั่น runhighlight ประกาศฟังก์ชั่น ไว้ที่ tag head เช่น

****

**แต่มีข้อเสียคือ**

เราต้องไปแก้ไขเอกสาร html เพื่อให้ฟังก์ชั่นทำงาน ทำให้โค้ดผูกติดกับเอกสาร html โค้ดไม่เป็นระเบียบ อาจเกิดการทำงานซ้ำๆ ในเพจต่างๆ หรือถูกเรียกในอีเวนต์ (event handlers)ต่างๆ เช่น การคลิกซึ่งเกิดขึ้นได้ตลอดทุกๆ อิลิเมนต์ของเพจและเพิ่มการทำงานใหม่ๆ เข้าไป จำเป็นต้องแก้โค้ดหลายๆ ที่ จึงเป็นการเพิ่มโอกาสความผิดพลาดสูงในการทำงาน และ**ไม่สามารถเรียกได้ 2 ฟังก์ชั่นให้ทำงานพร้อมกันได้ในขณะที่โหลด DOM เสร็จ**

**วิธีที่ 2**



วิธีนี้จะไม่ต้องจัดการแก้ไข html โดยตรง แต่จะเป็นการจัดการเขียนโค้ดในส่วนของ JavaScript ทั้งหมด จึงหมดปัญหาจากข้อเสียที่ได้กล่าวไว้ข้างต้น

**ทำความเข้าใจเกี่ยวกับ Selector ในแบบต่างๆ**

Selector คือการเลือกอิลิเมนต์ที่ต้องการจาก DOM (Document Object Model) เพื่อนำอิลิเมนต์มาทำการผูกฟังก์ชั่นต่างๆ หรือ กำหนด event ส่วนประกอบพื้นฐานของ selector มี 3 อย่างคือ

|  |  |  |  |
| --- | --- | --- | --- |
| **Selector** | **Css** | **jQuery** | **คำอธิบาย** |
| ชื่อแท็ก | p | $(“p”) | เลือกทุกอิลิเมนต์ที่อยู่ภายใต้แท็ก<p></p> |
| ID | #idName | $(“#idName”) | เลือกหนึ่งอิลิเมนต์ที่มี ID เท่ากับ idName |
| Class | .className | $(“.ClassName”) | เลือกทุกอิลิเมนต์ที่มี Class เท่ากับ ClassName |

**ตัวอย่างการใช้งาน Tag selector**

<h2>Ttitle</h2>

<strong>Ttitle<strong>

<p>Paragraph Content1</p>

<p>Paragraph Content1</p>

<p>Paragraph Content1</p>

$(document).ready(function(){

$(“p”).css(“font-size”,”15px”);

});

กำหนดให้ข้อมูลที่อยู่ภายใต้ Paragraph<p> มีขนาดตัวอักษรเท่ากับ 15px โดยใช้ Method การจัดการ CSS นั้นก็คือ .css()

**ตัวอย่างการใช้งาน ID selector**

<div id=”idName”>

Content…

</div>

$(document).ready(function(){

$(“#idName”).css(“font-size”,”20px”);

});

**ตัวอย่างการใช้งาน Class selector**

<ul>

<li class=”className”>menu1</li>

<li class=”className”>menu2</li>

<li class=”className”>menu3</li>

<li class=”className”>menu4</li>

</ul>

$(document).ready(function(){

$(“. className”).css({“font-size”:”20px”,”float”:”left”});

});

นอกจาก selector พื้นฐานแล้วก็ยังมี selector อื่นๆ ดังนี้

**Attribute Selector** เป็น selector ย่อยของ CSS ช่วยเลือก element ที่ต้องการโดยใช้ค่า properties ของ HTML เช่น Attribute title(<a href=”link.html” title=”value”></a>)

**ตัวอย่างการใช้งาน Attribute selector**

<style type=”text/css”>

.httpLink{ Background-color:#cccccc;}

.mailLink{Background-color:#ffccffcc;}

.pdfLink{ Background-color:red;}

</style>

<ul class=”menu”>

<li><a href=”http://www.workphp.com”>Link to Workphp</a></li>

<li>><a href=”file.pdf” title=”ducument”>Download PDF </a></li>

<li>><a href=”mailto:nn.it@hotmail.com” >Contact Us</a></li>

<li><a href=”map.html” title=”map”>Map</a></li>

</ul>

$(document).ready(function(){

$(“a[href^=http]”).addClass(“httpLink”);

$(“a[href$=.pdf]”).addClass(“pdfLink”);

$(“a[href^=mailto]”).addClass(“mailLink”);

$(“[title= map]”).css(“background-color”,”bule”);

});

**คัสตอมซีเลกเตอร์**

Custom Selector เป็นselectorที่ jQuery สร้างขึ้นมาเพื่อช่วยให้เลือก element ได้แม่นยำขึ้นโดยสามารถระบุตำแหน่ง element ตามต้องการได้เลย เช่น

**ตัวอย่างการใช้งาน Custom selector**

<ul id=”menu”>

<li>HOME</li>

**<li>ABOUT US</li>**

<li>CONTACT US</li>

</ul>

$(ducument).read(function(){

$(“ul#menu li:eq(**1**)”).css(“font-weight”,”bold”);

});

การนับข้อมูลใน array ของ javascript เป็นการนับในรูปแบบ zero-based คือเริ่มต้นที่ 0

**ชีเล็กเตอร์สำหรับฟอร์ม**

Selector สำหรับ Form เป็น selector ในการเลือก element ของ Form โดยเฉพาะ เพื่อความสะดวกในการเข้าถึงelement มากขึ้น โดยมี selector ที่ใช้กับ Formมีดังนี้

|  |  |
| --- | --- |
| **Selector** | **Elementที่ถูกเลือก** |
| **:text, :checkbox, :radio, :image, :submit, :reset , :password, :file** | เลือกอิลิเมนต์ที่มีแอตทริบิวต์ type เท่ากับชื่อซีเลกเตอร์ที่กำหนด เช่น :text เป็นการเลือกอิลิเมนต์<input type=”text”> |
| **:input** | เลือกอิลิเมนต์ทั้งหมดที่เป็นอิลิเมนต์<input>,<select><textarea><button>  (input form ทั้งหมด) |
| **:button** | เลือกอิลิเมนต์ <input> ที่มี type=”button” และอิลิเมนต์ <button> |
| **:enabled** | เลือกอิลิเมนต์ของฟอร์มที่สามารถป้อนข้อมูลได้ |
| **:disable** | เลือกอิลิเมนต์ของฟอร์มที่ไม่สามารถป้อนข้อมูลได้ |
| **:checked** | เลือกอิลิเมนต์เช็คบ็อกซ์ที่ถูกเลือกหรือ อิลิเมนต์เรดิโอทีถูกเลือก |
| **:selected** | เลือกอิลิเมนต์<option> ทั้งหมด |

ตัวอย่างการใช้งาน ชีเล็กเตอร์สำหรับฟอร์ม

<form>

<input type=”text” / >

<input type=”radio” />

<input type=”checkbox” />

<textarea></textarea>

<select >

<option>1</option>

<option>2</option>

</select>

<input type=”submit” value=”submit”>

</form>

$(ducment).ready(function(){

var allValue = $(“:input”). serialize ();

//หากไม่ใช้ซีเลกเตอร์ Form สามารถใช้ซีเลกเตอร์แบบปกติก็คือใช้ attribute selector

var allValue = $(“[type=input]”).serialize();

});

**การทำ Chaining (การเชื่อมเมธอด)**

Chaining คือการเชื่อมเมธอดต่อกันโดยไม่ต้องระบุselectorเพี่อสร้างออบเจ็กต์ใหม่ทุกครั้งที่เรียกใช้เมธอดเป็นการเพิ่มประสิทธิภาพในการทำงานของสคริปต์

**ตัวอย่าง**

<ul>

<li>row1</li>

<li>row2</li>

<li>row3</li>

</ul>

$(ducument).ready(function(){

$(“ul li:eq(1)”).css(“font-weight”,”bold”).end().find(“ul li:eq(0)”).css(“font-size”,”10px”);

});

**อีเวนต์ (Events)**

**jQuery Events รูปแบบคำสั่งการใช้ jQuery กับ Events บน jQuery Library Framework**

**jQuery กับ Events** ในภาษาของ **JavaScript** การใช้งาน **Events** นั้นมีความจำเป็นอย่างยิ่ง แทบจะเรียกได้ว่าเป็นหัวใจของการพัฒนาโปรแกรมของฝั่ง **Client** ได้เลยทีเดียว เพราะเหตุการณ์ต่างๆ เหล่านี้ที่เกิดขึ้นจากฝั่งของ **Client** จะเป็นตัวควบคุมและบ่งบอกวัตถุประสงค์ของผู้ใช้ ว่าต้องการทำอะไร ไปที่ไหน และอย่างไร โดย **Events** ต่างๆ ที User ได้กระทำนั้น เราสามารถทำการดักตรวจสอบและควบคุม เพื่อให้โปรแกรมนั้นไปในทิศทางที่ได้ออกแบบไว้ และสามารถทำงานได้อย่างถูกต้อง และการนำ jQuery มาใช้เกี่ยวกับการควบคุม **Events** ก็เป็นทางเลือกที่น่าสนใจอย่างยิ่ง เพราะ **jQuery** สามารถดักจับ **Events** ต่างๆ ที่เกิดขึ้นในฝั่ง Client ได้อย่างง่ายดายด้วย Code อันสั้น ๆ และสามารถใช้งานได้สะดวกกว่าในการเขียน **JavaScript** ด้วยระบบเดิม ๆ

**รูปแบบคำสั่งของ Events**  
jQuery Events Syntax

|  |
| --- |
| $("selector").event(); |

**เปรียบเทียบรูปแบบคำสั่งของ Events JavaScript ระบบเดิม กับ Events ของ jQuery**  
Events ของ JavaScript ระบบเดิม

|  |
| --- |
| <input type="button" id="btn1" OnClick="alert('Sawatdee');"> |

Events ของ jQuery

|  |
| --- |
| $("#btn1").click(alert('Sawatdee')); |

ในการกำหนด **Events ของ jQuery** สามารถเรียกใช้งาน method เพื่อกำหนดเหตุการณ์ให้กับ **Events** นั้นๆ ได้ในทันที หรือจะเรียก **Callback ของ function** เพื่อให้ทำงานต่างๆ ตามที่เราต้องการ โดยสามารถสร้างคำสั่งการทำงานลงใน **Callback ของ function** เพิ่มเติมได้

Events ที่ไมีมี Callback function

|  |
| --- |
| $("#btn1").click();  $("#div1").hide(); |

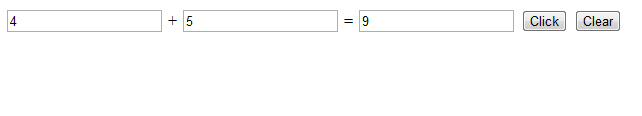
Events ที่มี Callback function

|  |
| --- |
| $("#btn1").click(function(){  $("#txt3").val(parseFloat($("#txt1").val())+parseFloat($("#txt2").val()));  }); |

**ตัวอย่างการใช้งาน Events ของ jQuery**

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Event jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  var $value = parseFloat($("#txt1").val())+parseFloat($("#txt2").val());  $("#txt3").val($value).show();  });  $("#btn2").click(function(){  $("#txt1").val("");  $("#txt2").val("");  $("#txt3").val("").hide();  });  });  </script>  <input type="text" id="txt1" />+<input type="text" id="txt2" />=<input type="text" id="txt3" />  <input type="button" id="btn1" value="Submit"/>  <input type="button" id="btn2" value="Clear"/>  </body>  </html> |

**Screenshot**  
  
**คำอธิบาย**  
ตัวอย่างการใช้งาน **Events** ของ **jQuery** แบบไม่มี Callback $("#txt3").hide(); และที่แบบมี Callback $("#btn1").click(function(){ และ $("#btn2").click(function(){ และจากตัวอย่างจะเห็นว่าต่อให้เราเปลี่ยนไปใช้ **jQuery Library Framework** แต่พื้นฐานของคำสั่งนั้นยังเป็นภาษา JavaScript ซึ่งบางคำสั่งของ **JavaScript** สามารถเรียกใช้งานได้ทันที เช่น ฟังก์ชั่นเกี่ยวกับการแปลงค่าหรือตรวจสอบชนิดของ Data Type

**คำสั่ง Event อื่น ๆ ที่น่าสนใจ**  
**$(document).ready(function)** ความหมายคือ มีมีการโหลดหน้าเว็บเพจเสร็จสิ้น ถ้ามีการใส่คำสั่ง JavaScript ที่ไม่ผ่าน Events จะทำงานทันทีหลังจากโหลดหน้าเว็บเพจเรียบร้อย เช่น $("#txt3").hide();  
**$(selector).click(function)** ความหมายคือ เมื่อมีการคลิกยัง Selectors ที่กำหนดขึ้น  
**$(selector).dblclick(function)** ความหมายคือ เมื่อมีการดับเบิ้ลคลิกยัง Selectors ที่กำหนดขึ้น  
**$(selector).focus(function)** ความหมายคือ เมื่อมีการคลิก focus() ยัง Selectors ที่กำหนดขึ้น เช่น textbox  
**$(selector).mouseover(function)** ความหมายคือ เมื่อมีเมาส์เลื่อนผ่าน Selectors ที่กำหนดขึ้น  
**$(selector).error(function)** ความหมายคือ เมื่อมีการ error จาก Selectors ที่กำหนดขึ้น เช่น Tags img โหลดภาพไม่สมบูรณ์  
**$(selector).focusout (function)** ความหมายคือ focusout ความหมายคือ เมื่อมีการย้ายตำแหน่ง focus()ออกจาก Selectors ที่กำหนดขึ้น เช่น เลื่อน focus() ออกจาก textbox  
**$(selector).keydown(function)** ความหมายคือ เมื่อมีการกดปุ่มลงในทันทีไปยัง Selectors ที่กำหนดขึ้น  
**$(selector).keypress(function)** ความหมายคือ เมื่อมีการกดปุ่มลงจนสุด ไปยัง Selectors ที่กำหนดขึ้น  
**$(selector).keyup(function)** ความหมายคือ เมื่อมีการปล่อยปุ่ม ที่ได้กดลงใน Selectors ที่กำหนดขึ้น

**Effect**

**jQuery Effects รูปแบบคำสั่งการใช้ jQuery กับ Effects บน jQuery Library Framework**

**jQuery กับ Effects** อีกหนึ่งในความสามารถของ **jQuery** ก็คือการกำหนด **Effects ให้กับ element** ที่สามารถกระทำและกำหนดได้ง่ายๆ ด้วยคำสั่งอันสั้นๆ เช่น $("#div1").hide(1000); จะเป็นการ hide ที่ element โดยจะค่อยๆ เลื่อนจางหายไปใน 1 วินาที หรือ $("div1").toggle(); ใช้ซ่อนและแสดง element ต่างๆ สลับไปมา และก็ยงมี **Effect** อื่นๆ อีกมากมายที่สามารถดัดแปลงการใช้งานจาก **jQuery Library**   
  
**รูปแบบคำสั่งของ Effects**  
**jQuery Effects**

|  |
| --- |
| $("selector").effect(); |

**1. ตัวอย่างการใช้งาน Effects ของ .toggle(); ใช้ซ่อนและแสดง element ต่างๆ สลับไปมา**

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Effect01 jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("#div1").toggle();  });  });  </script>  </head>  <body>  <div id="div1">Test Effect Toggle</div>  <input type="button" id="btn1" value="Click">  </body>  </html> |

**Screenshot**

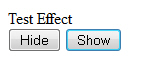
effect01.html.jpg  
**คำอธิบาย**

เมื่อคลิกที่ปุ่ม ข้อความจะหายไป และ คลิกอีกครั้ง ข้อความจะแสดงกลับมาเหมือนเดิม ซึ่งสามารถทำได้อย่างง่ายๆ จริงๆ หรือจะลองกำหนด $("#div1").toggle(1000); หน่วยวินาทีเพื่อหน่วงเวลาก็ได้เช่นเดียวกัน โดย 1000 = 1 วินาที  
นอกจาก method ชื่อว่า .toggle() แล้ว method ของ hide() และ show() ก็สามารถสร้าง Effect ได้เช่นเดียวกัน โดยการกำหนดหน่วงเวลาเป็นหน่วยวินาที เช่น

|  |
| --- |
| $("#div1").hide(1000);  $("#div1").show(1000); |

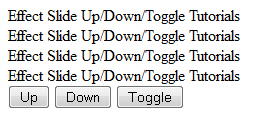
**2. ตัวอย่างการใช้งาน Effects ของ .hide(); และ .show(); ใช้ซ่อนและแสดง element ต่าง ๆ**

|  |
| --- |
| <html>  <head>  <title>jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("#div1").hide(1000);  });  $("#btn2").click(function(){  $("#div1").show(1000);  });  });  </script>  </head>  <body>  <div id="div1">Test Effect</div>  <input type="button" id="btn1" value="Hide">  <input type="button" id="btn2" value="Show">  </body>  </html> |

**Screenshot**  
  
**คำอธิบาย**  
เมื่อคลิกที่ปุ่ม Hide และ Show จะซ่อนและแสดงข้อความ โดยมีการหน่วงเวลาเพื่อสร้าง Effect ให้กับ element

**3. ตัวอย่างการใช้งาน Effects ของ .slideUp(); , .slideDown(); และ .slideToggle(); ใช้ Slide element ต่าง ๆ**

|  |
| --- |
| <html>  <head>  <title>Effect Slide Down Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("#div1").slideUp();  });  $("#btn2").click(function(){  $("#div1").slideDown();  });  $("#btn3").click(function(){  $("#div1").slideToggle();  });  });  </script>  </head>  <body>  <div id="div1">Effect Slide Up/Down/Toggle Tutorials<br />Effect Slide Up/Down/Toggle Tutorials <br />Effect Slide Up/Down/Toggle Tutorials <br />Effect Slide Up/Down/Toggle Tutorials<br /></div>  <input type="button" id="btn1" value="Up">  <input type="button" id="btn2" value="Down">  <input type="button" id="btn3" value="Toggle">  </body>  </html> |

**Screenshot**  
  
**คำอธิบาย**  
เมื่อคลิกที่ปุ่ม Up จะทำการ Slide ข้อความเก็บไว้ด้านบน  
เมื่อคลิกที่ปุ่ม Down จะทำการ Slide ข้อความลงมาด้านล่างเหมือนเดิม  
เมื่อคลิกที่ปุ่ม Toggle จะทำการ Slide ข้อความขึ้นลงสลับไปมา  
  
**สรุป**  
**$("selector").show();** ความหมายคือ แสดง Selectors ของ element ที่กำลังอ้างถึง  
**$("selector").hide();** ความหมายคือ ซ่อน Selectors ของ element ที่กำลังอ้างถึง  
**$("selector").toggle();** ความหมายคือ ซ่อนและแสดง Selectors ของ element ที่กำลังอ้างถึง โดยทำงานสลับกันไปมา  
**$("selector").slideUp();** ความหมายคือ เลือน Selectors ของ element ที่กำลังอ้างถึง โดยเลื่อนขึ้นด้านบน  
**$("selector").slideDown();** ความหมายคือ เลือน Selectors ของ element ที่กำลังอ้างถึง โดยเลื่อนขึ้นลงล่างเหมือนเดิม  
**$("selector").slideToggle();** ความหมายคือ เลือน Selectors ของ element ที่กำลังอ้างถึง โดยทำงานสลับกันไปมา เลือนขึ้น-เลือนลง และจากตัวอย่างจะเห็นว่าต่อให้เราเปลี่ยนไปใช้ **jQuery Library Framework** แต่พื้นฐานของคำสั่งนั้นยังเป็นภาษา **JavaScript** ซึ่งบางคำสั่งของ **JavaScript** สามารถเรียกใช้งานได้ทันที เช่น ฟังก์ชั่นเกี่ยวกับการแปลงค่าหรือตรวจสอบชนิดของ Data Type

**คำสั่ง Event อื่น ๆ ที่น่าสนใจ**  
**$(selector).fadeIn()** ความหมายคือ fade element ค่อย ๆ แสดงผลจางลง  
**$(selector).fadeOut()** ความหมายคือ fade element ค่อย ๆ แสดงผลชัดขึ้น  
**$(selector).fadeTo()** ความหมายคือ fade element การปรับภาพจากเข้มไปจาง  
**$(selector).animate()** ความหมายคือ การกำหนดเอนิเมชั่นตามความต้องการโดยกำหนด css  
**$(selector).fadeToggle()** ความหมายคือ สลับการะหว่าง faceIn และ fadeOut  
**$(selector).stop()** ความหมายคือ หยุดการทำงานที่กำลังทำงานอยู่ในขณะนั้น

\

**DOM Document Object Model**

**jQuery HTML รูปแบบคำสั่งการใช้ jQuery กับ HTML บน jQuery Library Framework**

**jQuery กับ HTML** จากเรื่องบทก่อนหน้านี้ จะเห็นว่า **jQuery** สามารถใช้ **Selector** ในการอ้างถึง **element** ในหน้าเว็บเพจได้อย่างง่ายดาย และหลังจากที่สามารถอ้างถึง **element** เหล่านั้น ได้แล้ว สิ่งที่จะทำต่อไปก็คือ การอ่านค่า หรือ กำหนดค่าให้กับ **element** นั้นๆ ซึ่งในบทนี้ จะเป็นการกำหนดค่าให้กับ **HTML element** ที่อยู่ในรูปแบบ innetText หรือ innetHTML (ในรูปแบบ JavaScript แบบดั่งเดิม) ส่วนการกำหนด **Events หรือ Attribute** อื่นๆ จะได้กล่าวถึงในหัวข้อถัดไป

**1. การใช้งาน method ของ $(selector).text(content) และ $(selector).html(content)**  
**inner HTML กับ jQuery**  
**Syntax ใน JavaScript ดังเดิม**

|  |
| --- |
| document.getElementById('div1').innerHTML = 'My Text'; |

**Syntax ใน jQuery จะใช้ method สั้น ๆ ว่า html()**

|  |
| --- |
| $("#div1").html("My Text"); |

**innerText กับ jQuery**  
**Syntax ใน JavaScript ดังเดิม**

|  |
| --- |
| document.getElementById('div1').innerText = '<b>My Text</b>'; |

**Syntax ใน jQuery จะใช้ method สั้น ๆ ว่า text()**

|  |
| --- |
| $("#div1").text("My Text"); |

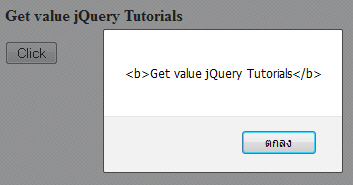
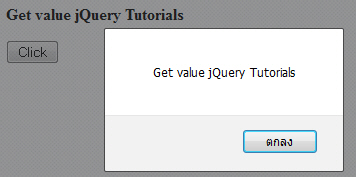
**ตัวอย่างการใช้ method ของ text() และ html()**   
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Dom01 jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("p").html($("#txt1").val());  $("#div1").text($("#txt1").val());  });  });  </script>  </head>  <body>  <p></p>  <div id="div1"></div>  <input type="text" id="txt1">  <input type="button" id="btn1" value="Click">  </body>  </html> |

**Screenshot**  
  
**คำอธิบาย**  
เมื่อมีการรับค่า string ที่เป็น HTML Tags และเมื่อใช้ .text และ .html จะได้ผลลัพธ์ต่างกัน เหมือนในรูป

**นอกจากนี้ในทางกลับกัน method ก็สามารถทำการอ่านค่าจาก .text() และ .html() ได้เช่นเดียวกัน**  
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Get value jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  alert($("p").html());  alert($("p").text());  });  });  </script>  </head>  <body>  <p><b>Get value jQuery Tutorials</b></p>  <input type="button" id="btn1" value="Click">  </body>  </html> |

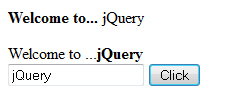
**Screenshot**  
  
  
  
ใช้ .html() จะได้ค่าเป็น HTML Tags  
  
  
ใช้ .text() จะได้ค่าเป็น Text เท่านั้น

**สรุป**  
**$(selector).text(content)** ใช้ในการอ่านค่าและส่งค่าในรูปแบบของ Text  
**$(selector).html(content)** ใช้ในการอ่านค่าและส่งค่าในรูปแบบของ HTML Tags  
**คำอธิบาย**  
ใช้ method ของ .html() จะได้ค่าเป็น **HTML Tags** และ ใช้ .text() จะได้ค่าเป็น Text เท่านั้น

**2. การใช้งาน method ของ $(selector).append(content)**

สำหรับ **method append()** ใช้สำหรับการเพิ่มเนื้อหาเข้าไปกับ element ที่ได้เลือกไว้ เช่น **<p>Welcome to</p>** เมื่อใช้ **.append("jQuery")** จะได้ string ใหม่มีค่าเป็น **<p>Welcome to jQuery</p>**  
**ตัวอย่าง**  
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Dom03 using append jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("p").append($("#txt1").val());  $("#div1").append("<b>"+$("#txt1").val()+"</b>");  });  });  </script>  </head>  <body>  <p><b>Welcome to... </b></p>  <div id="div1">Welcome to ...</div>  <input type="text" id="txt1">  <input type="button" id="btn1" value="Click">  </body>  </html> |

  
**คำอธิบาย**  
ทั้ง <p> และ <div id="div1"> ใช้ method append() ซึ่งได้ผลลัพธ์เหมือนกัน โดย append() สามารถรับค่าที่เป็น HTML Tags ได้เช่นเดียวกัน  
**ผลลัพธ์ทีได้ในมุมมองของ HTML Tags**

|  |
| --- |
| <p><b>Welcome to... jQuery</b></p> <div id="div1">Welcome to ... jQuery </div> |

**3. การใช้งาน method ของ $(selector).prepend(content)**

สำหรับ **method prepend()** ใช้สำหรับการเพิ่มเนื้อหาเข้าไปกับ element ที่ได้เลือกไว้ โดยเพิ่มเข้าไปข้างหน้าเนื้อหาเดิมที่มีอยู่ เช่น **<p>Welcome to</p>** เมื่อใช้ **.prepend("**Dom using prepend jQuery Tutorials **")** จะได้ string ใหม่มีค่าเป็น **<p>** Dom using prepend jQuery Tutorials **Welcome to</p>**  
**ตัวอย่าง**  
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Dom using prepend jQuery Tutorials </title>  <script type="text/javascript" src="jquery-1.6.4.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("p").prepend($("#txt1").val());  $("#div1").prepend("<b>"+$("#txt1").val()+"</b>");  });  });  </script>  </head>  <body>  <p><b>Welcome to... </b></p>  <div id="div1">Welcome to ...</div>  <input type="text" id="txt1">  <input type="button" id="btn1" value="Click">  </body>  </html> |

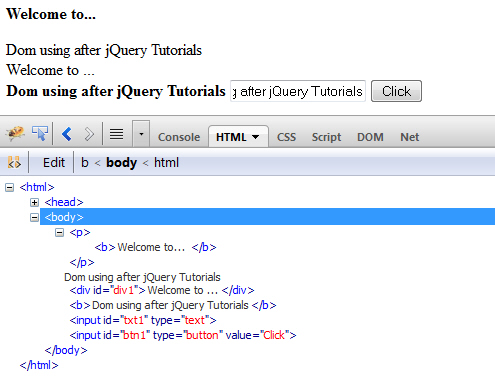
  
**คำอธิบาย**  
ทั้ง <p> และ <div id="div1"> ใช้ method prepend() ซึ่งได้ผลลัพธ์เหมือนกัน โดย prepend() สามารถรับค่าที่เป็น HTML Tags ได้เช่นเดียวกัน  
**ผลลัพธ์ทีได้ในมุมมองของ HTML Tags**

|  |
| --- |
| <p><b> Dom using prepend jQuery Tutorials Welcome to... </b></p> <div id="div1"> Dom using prepend jQuery Tutorials Welcome to ...</div> |

**4. การใช้งาน method ของ $(selector).after (content)**

สำหรับ **method append()** ใช้สำหรับการเพิ่มเนื้อหาหลังจากสิ้นสุด element ที่ได้เลือกไว้ เช่น **<p>Welcome to</p>** เมื่อใช้ **.after("Dom using after jQuery Tutorials")** จะได้ string ใหม่มีค่าเป็น **<p>Welcome to</p> Dom using after jQuery Tutorials**  
**ตัวอย่าง**  
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Dom using after jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("p").after($("#txt1").val());  $("#div1").after("<b>"+$("#txt1").val()+"</b>");  });  });  </script>  </head>  <body>  <p><b>Welcome to... </b></p>  <div id="div1">Welcome to ...</div>  <input type="text" id="txt1">  <input type="button" id="btn1" value="Click">  </body>  </html> |



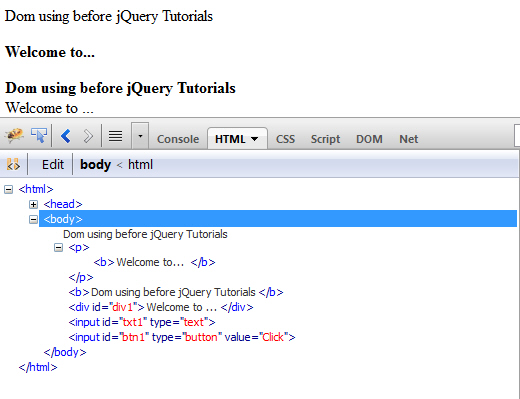
**คำอธิบาย**  
ทั้ง <p> และ <div id="div1"> เมื่อใช้ method after() ซึ่งได้ผลลัพธ์เหมือนกัน คือมีการแสดงผลขึ้นบรรทัดใหม่ เพราะมีการเพิ่มเนื้อหาหลังจากจบ Tags นั้นแล้ว  
**ผลลัพธ์ทีได้ในมุมมองของ HTML Tags**

|  |
| --- |
| <p><b>Welcome to... </b></p> Dom using after jQuery Tutorials  <div id="div1">Welcome to ...</div><b> Dom using after jQuery Tutorials </b> |

**5. การใช้งาน method ของ $(selector).before(content)**

สำหรับ **method before()** ใช้สำหรับการเพิ่มเนื้อหาก่อนเริ่ม element ที่ได้เลือกไว้ เช่น **<p>Welcome to</p>** เมื่อใช้ **.before("**Dom using before jQuery Tutorials **")** จะได้ string ใหม่มีค่า Dom using before jQuery Tutorials **<p>Welcome to</p>**   
  
**ตัวอย่าง**  
**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Dom using before jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("p").before($("#txt1").val());  $("#div1").before("<b>"+$("#txt1").val()+"</b>");  });  });  </script>  </head>  <body>  <p><b>Welcome to... </b></p>  <div id="div1">Welcome to ...</div>  <input type="text" id="txt1">  <input type="button" id="btn1" value="Click">  </body>  </html> |

  
**คำอธิบาย**  
ทั้ง <p> และ <div id="div1"> เมื่อใช้ method before() ซึ่งได้ผลลัพธ์เหมือนกัน คือมีการแสดงผลขึ้นบรรทัดใหม่ เพราะมีการเพิ่มเนื้อหาก่อนเริ่ม Tags นั้น  
**ผลลัพธ์ทีได้ในมุมมองของ HTML Tags**

Dom using before jQuery Tutorials<p><b>Welcome to... </b></p>  
**<b> Dom using before jQuery Tutorials </b>** <div id="div1">Welcome to ...</div>

**สรุป**  
**$(selector).text(content)** ใช้ในการอ่านค่าและส่งค่าในรูปแบบของ Text  
**$(selector).html(content)** ใช้ในการอ่านค่าและส่งค่าในรูปแบบของ HTML Tags  
**$(selector).append(content)** เพิ่มเนื้อหาเข้าไปใน Tags โดยต่อจากเนื้อหาที่มีอยู่แล้ว  
**$(selector).prepend(content)** เพิ่มเนื้อหาเข้าไปใน Tags โดยเพิ่มแทรกเนื้อหาก่อนเนื้อหาที่มีอยู่แล้ว  
**$(selector).after(content)** เพิ่มเนื้อหาหลังจากจบ Tags  
**$(selector).before(content)** เพิ่มเนื้อหาก่อนเริ่ม Tags

จะเห็นว่า jQuery ได้ช่วยให้การเขียน Code สั้นลงและง่ายขึ้นมาก Syntax ที่เขียนก็เข้ากันได้อย่างดี และเข้าใจง่าย ซึ่งถ้าหากอ่านและลองปฏิบัติตามให้เข้าใจทุกส่วน ก็จะพอเข้าใจหลักการทำงานพื้นฐานของ jQuery ได้เป็นอย่างดี

**Ajax**

|  |
| --- |
|  |
|  |

**jQuery Ajax รูปแบบคำสั่งการใช้ jQuery กับ Ajax บน jQuery Library Framework**

**jQuery กับ Ajax** ใน **jQuery** เองก็ได้ออกแบบ function ที่ทำงานร่วมกับ **Ajax** ได้อย่างมีประสิทธิภาพ ด้วย Code คำสั่งที่สั้นมากๆ และยังไม่ต้องกังวลเรื่องปัญหา **Cross ของ Web Browser** อีกด้วย ด้วยประสิทธิภาพและการออกแบบที่เป็นมาตรฐานและถูกต้องตามหลักของ **W3C CSS Validation** รองรับการทำงานทุกเว็บบราวเซอร์ และสามารถทำงานได้รวดเร็ว ด้วย Library ไฟล์ขนาดเล็ก องค์กรหลายๆ แห่งได้ประกาศใช้ **jQuery เป็น Core** หลักในการทำงานร่วมกับ **Ajax** เช่น Google ได้ประกาศใช้ Library ของ **jQuery** ใช้ร่วมกับ Application หลายตัวทีได้พัฒนาขึ้น Open Source ดังๆ หลายตัว หรือแม้กระทั้ง Microsoft ก็ได้นำ jQuery มาเป็นส่วนหนึ่งของ Framework ที่ใช้ในการพัฒนา ร่วมกับ ASP.NET (Web Application) ที่ทำงานร่วมกับ **Ajax** ด้วย เหตุผลและความสามารถของ jQuery นี้เอง คาดว่าในอนาคต jQuery นั้นมีความจำเป็นมากๆ ที่จะต้องเรียนรู้ และศึกษาการใช้งานอย่างจริงจัง เพื่อสร้างเสริมประสบการณ์และความรู้ใหม่ๆ ในการพัฒนา **Web Application** ในรูปแบบต่างๆ

**พื้นฐาน jQuery Ajax Syntax**  
**jQuery Ajax**

|  |
| --- |
| $(selector).load(url,data,callback); |

**1. ตัวอย่างการใช้งาน jQuery แบบง่ายๆ ผ่าน method load()**  
**Code (server\_side.php)**

|  |
| --- |
| <?php  for($i=1;$i<=10;$i++)  {  echo $i."<br>";  }  ?> |

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Ajax Load jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $("#div1").load(‘server\_side.php');  });  });  </script>  </head>  <body>  <div id="div1"></div>  <input type="button" id="btn1" value="Load">  </body>  </html> |

**Screenshot**  
  
**คำอธิบาย**  
เมื่อคลิกที่ปุ่ม Load จะมีการ load ข้อมูลจากไฟล์ **04\_ajax\_load**.php ซึ่งเป็นการทำงานง่ายๆ ด้วยคำสั่งสั้นๆ ของ jQuery   
  
**2. ตัวอย่างการใช้งาน jQuery แบบง่าย ๆ ผ่าน method ajax()** โดยวิธีนี้สามารถตรวจสอบสถานะการทำงาน และกำหนดคุณสมบัติต่างๆ ได้  
**Code (server\_side.php)**

|  |
| --- |
| <?php  for($i=1;$i<=10;$i++)  {  echo $i."<br>";  }  ?> |

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>ajax method jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $.ajax({ url: "server\_side.php" })  .success(function(data) { $("#div1").html(data); })  .error(function() { $("#div1").html("error"); })  .complete(function() { $("#div1").after("Ajax load finished"); });  });  });  </script>  </head>  <body>  <div id="div1"></div>  <input type="button" id="btn1" value="Load">  </body>  </html> |

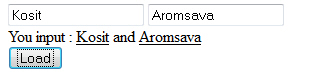
**Screenshot**  
  
**คำอธิบาย**  
เมื่อคลิกที่ปุ่ม Load จะมีการ load ข้อมูลจากไฟล์ server\_side.php โดยมีการตรวจสอบผลการทำงานว่า   
**.success** เมื่อทำงานสมบูรณ์ไม่มีข้อผิดพลาด  
**.error** เมื่อมีการทำงานผิดพลาด เช่น หาไฟล์ไม่เจอ 404   
**.complete** เมื่อ Ajax มีการทำงานเสร็จสิ้น ไม่ว่าจะ suscess หรือ error ก็ตาม  
  
**3. ตัวอย่างการใช้งาน jQuery Ajax ส่งค่าแบบ $\_POST**  
**Code (**server\_side\_post.PHP**)**

|  |
| --- |
| <?php  echo "You input : <u>".$\_POST["data1"]."</u> and <u>".$\_POST["data2"]."</u>";  ?> |

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Ajax Post jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $.post("server\_side\_post.php", {  data1: $("#txt1").val(),  data2: $("#txt2").val()},  function(data){  $("#div1").html(data);  }  );  });  });  </script>  </head>  <body>  <input type="text" id="txt1">  <input type="text" id="txt2">  <div id="div1"></div>  <input type="button" id="btn1" value="Load">  </body>  </html> |

**Screenshot**

  
**คำอธิบาย**  
จะมีการส่งค่าตัวแปรแบบ $\_POST โดยส่งค่า #txt1 และ #txt2 หรือถ้าต้องการส่งค่ามากกว่านี้ก็สามารถเพิ่มตัวแปรได้แบบง่ายๆ เช่น

|  |
| --- |
| <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $.post("webpage.php", {  data1: $("#txt1").val(),  data2: $("#txt2").val(),  data3: $("#txt3").val(),  data4: $("#txt4").val(),  data5: $("#txt5").val()},  function(result){  $("#div1").html(result);  }  );  });  });  </script> |

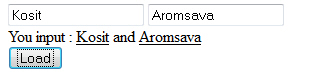
**4. ตัวอย่างการใช้งาน jQuery Ajax ส่งค่าแบบ $\_GET**  
**Code (webpage.php)**

|  |
| --- |
| <?php  echo "You input : <u>".$\_GET["data1"]."</u> and <u>".$\_GET["data2"]."</u>";  ?> |

**Code (jQuery)**

|  |
| --- |
| <html>  <head>  <title>Ajax Get jQuery Tutorials</title>  <script type="text/javascript" src="http://code.jquery.com/jquery-latest.js"></script>  <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $.get("server\_side\_get.php", {  data1: $("#txt1").val(),  data2: $("#txt2").val()},  function(data){  $("#div1").html(data);  }  );  });  });  </script>  </head>  <body>  <input type="text" id="txt1">  <input type="text" id="txt2">  <div id="div1"></div>  <input type="button" id="btn1" value="Load">  </body>  </html> |

**Screenshot**

  
**คำอธิบาย**  
จะมีการส่งค่าตัวแปรแบบ $\_GET โดยส่งค่า #txt1 และ #txt2 หรือถ้าต้องการส่งค่ามากกว่านี้ก็สามารถเพิ่มตัวแปรได้แบบง่ายๆ เช่น

|  |
| --- |
| <script type="text/javascript">  $(document).ready(function(){  $("#btn1").click(function(){  $.get("webpage.php", {  data1: $("#txt1").val(),  data2: $("#txt2").val(),  data3: $("#txt3").val(),  data4: $("#txt4").val(),  data5: $("#txt5").val()},  function(result){  $("#div1").html(result);  }  );  });  });  </script> |

**ภาคผนวก**

ภาคผนวกนี้ใช้สำหรับเป็น Quick reference สำหรับ API ของ jQuery ซึ่งประกอบด้วย selector expression และเมธอดต่างๆ

**Selector expressions**

เราใช้ฟังก์ชั่นแฟกทอรี่ $() ของไลบรารี jQuery เพื่อค้นหาอิลิเมนต์ที่อยู่ในเว็บเพจโดยฟังก์ชั่นนี้จะรับค่าอาร์กิวเมนต์เป็นสตริงที่อยู่ในรูปแบบที่คล้ายกับ syntax ของภาษา CSS ที่เรียกว่า selector expreccion

|  |  |
| --- | --- |
| **ซีเลกเตอร์** | **อิลิเมนต์ที่ถูกเลือก** |
| \* | เลือกทุกอิลิเมนต์ |
| #id | เลือกอิลิเมนต์ที่ตรงกับ id ที่ระบุ |
| element | เลือกอิลิเมนต์ที่ตรงกับชนิด(ชื่อแท็ก)ของอิลิเมนต์ที่ระบุ |
| .class | เลือกอิลิเมนต์ทุกตัวที่ตรงกับคลาสที่ระบุ |
| a, b | เลือกอิลิเมนต์ b ที่อยู่ภายในอิลิเมนต์ a |
| a > b | เลือกอิลิเมนต์ b ที่เป็นลูกของอิลิเมนต์ aนอิลิเมนตลกข |
| a + b | เลือกอิลิเมนต์ b ที่มีอิลิเมนต์ a นำหน้า |
| a - b | เลือกอิลิเมนต์ b ที่เป็นอิลิเมนต์พี่น้องกับอิลิเมนต์ a |
| : first | เลือกอิลิเมนต์แรกในกลุ่มอิลิเมนต์ถูกเลือก |
| : last | เลือกอิลิเมนต์สุดท้ายในกลุ่มอิลิเมนต์ถูกเลือก |
| : not (a) | เลือกอิลิเมนต์ทุกตัวในกลุ่มที่ไม่ใช่อิลิเมนต์ a |
| : even | เลือกอิลิเมนต์ที่มีลำดับเลขคู่จากกลุ่มอิลิเมนต์ (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| : odd | เลือกอิลิเมนต์ที่มีลำดับเลขคี่จากกลุ่มอิลิเมนต์ (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| **ซีเลกเตอร์** | **อิลิเมนต์ที่ถูกเลือก** |
| : eq (index) | เลือกอิลิเมนต์ตัวเดียวที่มีค่าอินเด็กซ์เท่ากับ index จากกลุ่มอิลิเมนต์ที่ถูกเลือก (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| : qt (index) | เลือกอิลิเมนต์ทุกตัวที่มีค่าอินเด็กซ์มากกว่า index จากกลุ่มอิลิเมนต์ที่ถูกเลือก (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| : lt (index) | เลือกอิลิเมนต์ทุกตัวที่อยู่ในกลุ่มอิลิเมนต์ที่ถูกเลือก ที่มีค่าอินเด็กซ์น้อยกว่า index (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| : header | เลือกอิลิเมนต์ที่เป็นเฮดเดอร์ (เช่น <hl>, <h2>) |
| : animated | เลือกอิลิเมนต์ที่กำลังแสดงแอนิเมชั่นอยู่ |
| : contains (text) | เลือกอิลิเมนต์ที่มีข้อความตรงกับที่ระบุอยู่ภายในอิลิเมนต์ |
| : empty | เลือกอิลิเมนต์ที่ไม่มีอิลิเมนต์ลูกหรือข้อความอยู่ภายใน |
| : has (a) | เลือกอิลิเมนต์ที่มีอิลิเมนต์ภายในอิลิเมนต์ a |
| : parent | เลือกอิลิเมนต์ที่มีอิลิเมนต์ลูก หรือมีข้อความอยู่ภายใน |
| : hidden | เลือกอิลิเมนต์ที่ถูกซ่อนจากการกำหนดสไตล์ใน CSS หรือเป็นอิลิเมนต์  <input type=”hidden” /> |
| : visible | เลือกอิลิเมนต์ที่แสดงบนหน้าจอทั้งหมด |
| [attr] | เลือกอิลิเมนต์ที่แอตทริบิวต์ชื่อ attr |
| [attr=value] | เลือกอิลิเมนต์ที่มีชื่อแอตทริบิวต์ attr และมีค่าเท่ากับ value |
| [attr!=value] | เลือกอิลิเมนต์ที่มีชื่อแอตทริบิวต์ชื่อ attr ที่มีค่าไม่เท่ากับ value |
| [attr^=value] | เลือกอิลิเมนต์ที่มีแอตทริบิวต์ชื่อ attr และค่าในแอตทริบิวต์เริ่มด้วยคำว่า value |
| [attr$=value] | เลือกอิลิเมนต์ที่มีแอตทริบิวต์ชื่อ attr และมีค่าในแอตทริบิวต์ลงท้ายด้วยคำว่า value |
| **ซีเลกเตอร์** | **อิลิเมนต์ที่ถูกเลือก** |
| [attr\*=value] | เลือกอิลิเมนต์ที่มีแอตทริบิวต์ชื่อ attr และค่าในแอตทริบิวต์มีคำว่า value อยู่ภายใน |
| :nth-child(index) | เลือกอิลิเมนต์ลูกที่อยู่ในลำดับที่ index ของอิลิเมนต์แม่ (อินเด็กซ์มีค่าเริ่มต้นที่เลข 1) |
| :nth-child(even) | เลือกอิลิเมนต์ลูกที่มีลำดับเป็นเลขคู่ของอิลิมเนต์แม่ (อินเด็กซ์มีค่าเริ่มต้นที่เลข 1) |
| :nth-child(odd) | เลือกอิลิเมนต์ลูกที่มีลำดับเป็นเลขคี่ของอิลิเมนต์แม่ (อินเด็กซ์มีค่าเริ่มต้นที่เลข 1) |
| :nth-child(formula) | เลือกอิลิเมนต์ลูกที่มีลำดับที่ได้ระบุในสมการ โดยสมการจะอยู่ในรูปแบบ an+b โดย a และ b เป็นเลขจำนวนเต็ม (อินเด็กซ์มีค่าเริ่มต้นที่เลข 1)ด์ |
| :first-child | เลือกอิลิเมนต์ลูกตัวแรกของอิลิเมนต์แม่ |
| :last-child | เลือกอิลิเมนต์ลูกตัวสุดท้ายของอิลิเมนต์แม่ |
| :only-child | เลือกอิลิเมนต์ลูกที่ไม่มีอิลิเมนต์พี่น้องภายในอิลิเมนต์แม่ |
| :input | เลือกอิลิเมนต์ทั้งหมดที่เป็นอิลิเมนต์ <input>, <select>, <textarea> และอิลิเมนต์ <button> |
| :text | เลือกอิลิเมนต์ <input> ที่มี type=”text” |
| :password | เลือกอิลิเมนต์ <input> ที่มี type=”password” |
| :radio | เลือกอิลิเมนต์ <input> ที่มี type=”radio” |
| :checkbox | เลือกอิลิเมนต์ <input> ที่มี type=”checkbox” |
| :submit | เลือกอิลิเมนต์ <input> ที่มี type=”submit” |
| :image | เลือกอิลิเมนต์ <input> ที่มี type=”image” |
| :reset | เลือกอิลิเมนต์ <input> ที่มี type=”reset” |
| :button | เลือกอิลิเมนต์ <input> ที่มี type=”button”, และเลือกอิลิเมนต์ <button> |
| :file | เลือกอิลิเมนต์ <input> ที่มี type=”file” |
| :enabled | เลือกอิลิเมนต์ของฟอร์มที่สามารถป้อนข้อมูลได้ |
| **ซีเลกเตอร์** | **อิลิเมนต์ที่ถูกเลือก** |
| :disabled | เลือกอิลิเมนต์­­ของฟอร์มที่ไม่สามารถป้อน­ข้อมูล |
| :checkd | เลือกอิลิเมนต์เซ็กบ็อกที่ถูกเลือก และอิลิเมนต์ปุ่มเรดิโอที่ถูกเลือก |
| :selected | เลือกอิลิเมนต์ <option> ที่ถูกเลือกในฟอร์ม |

**เมธอดสำหรับค้นหาอิลิเมนต์ DOM**

หลังจากที่สร้างออบเจ็กต์ jQuery โดยใช้ฟังก์ชั่น $ () เราสามารถแก้ไขกลุ่มอิลิเมนต์ที่เรากำลังทำงานอยู่ โดยการเรียกเมธอดสำหรับค้นหาอิลิเมนต์ DOM

|  |  |
| --- | --- |
| **เมธอดสำหรับค้นหาอิลิเมนต์** | **รีเทิร์นออบเจ็กต์ jQuery ที่มี...** |
| .filter (selector) | เลือกอิลิเมนต์ที่ตรงกับซีเลกเตอร์ที่ระบุ |
| .filter(callback) | เลือกอิลิเมนต์ที่ฟังก์ชั่นอลล์แบ็กรีเทิร์นค่าเป็น true |
| .eq(index) | เลือกอิลิเมนต์ที่มีค่าอินเด็กซ์เท่ากับ index (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| .slice(start,[end]) | เลือกอิลิเมนต์ที่อยู่ในช่วงค่าอินเด็กซ์ที่กำหนด (อินเด็กซ์มีค่าเริ่มต้นที่เลข 0) |
| .not(selector) | เลือกอิลิเมนต์ที่ไม่ตรงกับซีแลกเตอร์ที่ระบุ |
| .add(selector) | นำอิลิเมนต์จากซีแลกเตอร์ที่ระบุมาเพิ่มในกลุ่มอิลิเมนต์ที่ได้เลือกไว้ |
| .find(selector) | เลือกอิลิเมนต์ภายในที่ตรงกับซีเลกเตอร์ที่ระบุ |
| .contents() | เลือกอิลิเมนต์ลูกทั้งหมด (รวมทั้งข้อความที่อยู่ภายในอิลิเมนต์ด้วย) |
| .children([selector]) | เลือกอิลิเมนต์ลูก (สามารถคัดกรองอิลิเมนต์ที่ถูกเลือกด้วยการระบุซีเลกเตอร์) |
| .next([selector]) | เลือกอิลิเมนต์พี่น้องที่อยู่ถัดจากอิลิเมนต์แต่ละตัวที่ถูกเลือกทันที (สามารถทำการฟิลเตอร์อิลิเมนต์ตัวถัดไปด้วยการระบุซีเลกเตอร์) |
| **เมธอดสำหรับค้นหาอิลิเมนต์** | **รีเทิร์นออบเจ็กต์ jQuery ที่มี...** |
| .nextAll([selector]) | เลือกอิลิเมนต์พี่น้องทุกตัวที่อยู่ถัดจากอิลิเมนต์แต่ละตัวที่ถูกเลือก (สามารถทำการฟิลเตอร์อิลิเมนต์ตัวถัดไปทุกตัวด้วยการระบุซีเลกเตอร์) |
| .prev([selector]) | เลือกอิลิเมนต์พี่น้องที่อยู่ก่อนหน้าอิลิเมนต์แต่ละตัวที่ถูกเลือกทันที (สามารถทำการฟิลเตอร์อิลิเมนต์ตัวก่อนหน้าด้วยการระบุซีเลกเตอร์ |
| .prevAll([selector]) | เลือกอิลิเมนต์พี่น้องทุกตัวที่อยู่ก่อนหน้าอิลิเมนต์แต่ละตัวที่ถูกเลือก (สามารถทำการฟิลเตอร์อิลิเมนต์ตัวก่อนหน้าทุกตัวด้วยการระบุซีเลกเตอร์) |
| .siblings([selector]) | เลือกอิลิเมนต์พี่น้องทุกตัวทั้งที่อยู่ก่อนหน้าและหลังอิลิเมนต์แต่ละตัวที่ถูกเลือก (สามารถทำการฟิลเตอร์เมนต์ด้วยการระบุซีเลกเตอร์) |
| .parent([selector]) | เลือกอิลิเมนต์แม่ของอิลิเมนต์แต่ละตัวที่ถูกเลือก (สามารถทำการฟิลเตอร์อิลิเมนต์แม่ด้วยการระบุซีเลกเตอร์) |
| .parents([selector]) | เลือกอิลิเมนต์ระดับบนทุกตัวของอิลิเมนต์ของอิลิเมนต์ที่ถูกเลือก (สามารถทำการฟิลเตอร์อิลิเมนต์รดับบนด้วยการระบุซีเลกเตอร์) |
| .closest([selector]) | เลือกอิลิเมนต์ตัวแรกที่ตรงกับซีเลกเตอร์ที่ระบุ โดยเริ่มต้นค้นหาจากอิลิเมนต์ที่ถูกเลือกไปยังอิลิเมนต์ที่อยู่ระดับบน |
| .offsetParent() | เลือกอิลิเมนต์แม่ที่มีการจัดวางตำแหน่ง (เช่น relative, absolute) ของอิลิเมนต์ที่ถูกเลือกตัวแรก |
| .andSelf() | เลือกกลุ่มอิลิเมนต์ที่กำลังถูกเลือกอยู่ในปัจจุบัน บวกกับกลุ่มอิลิเมนต์ก่อนหน้านี้ที่อยู่ในสแต็กของ jQuery |
| .end() | เลือกกลุ่มอิลิเมนต์ก่อนหน้านี้ที่อยู่ในสแต็ก­ของ jQuey |
| .map(callback) | ได้ผลลัพธ์จากการเรียกฟังก์ชั่น callback ให้ทำงานกับอิลิเมนต์ที่ถูกเลือกแต่ละตัว |

**เมธอดสำหรับอีเวนต์**

ในการโต้ตอบกับผู้ใช้ เราจำเป็นต้องกำหนดฟังซ์ชั่นจัดการโดยการใช้ เมธอดสำหรับอีเวนต์ (event method) เหล่านี้ สังเกตว่าอีเวนต์ของ DOM จำนวนมากจะบังคับใช้กับชนิดของอิลิเมนต์ที่แน่นอนเท่านั้น

|  |  |
| --- | --- |
| **เมธอดสำหรับอีเวนต์** | **คำอธิบาย** |
| .ready(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อโหลด DOM และ CSS เสร็จแล้ว |
| .bind(type,[data],handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเกิดชนิดอีเวนต์ที่ระบุกับอิลิเมนต์ที่ถูกเลือก |
| .one(type,[data],handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกเกิดชนิดอีเวนต์ที่ระบุ และจะยกเลิกการกำหนดฟังก์ชั่นจัดการอีเวนต์นี้ทันทีที่จบการทำงาน |
| .unbind([type],[handler]) | ยกเลิกการกำหนดฟังก์ชั่นจัดการอีเวนต์ที่อยู่ในอิลิเมนต์ที่ถูกเลือก (สำหรับยกเลิกชนิดอีเวนต์ที่ระบุ, ยกเลิกเฉพาะฟังก์ชั่นที่ระบุชื่อหรือยกเลิกฟังก์ชั่นจัดการอีเวนต์ทั้งหมด) |
| .live(type,handler) | ใช้ตัวแทนจัดการอีเวนต์ (event delegation) กำหนดให้ฟังก์ชั่น handler ถูกเลือก เมื่ออิลิเมนต์ที่ถูกเลือกเกิดชนิดอีเวนต์ที่ระบุ |
| .die(type,[handler]) | ยกเลิกการกำหนดฟังก์ชั่นจัดการอีเวนต์ให้กับอิลิเมนต์ที่ถูกเลือกที่ได้มีการกำหนดฟังก์ชั่นจัดการอีเวนต์ด้วยเมธอด .live () |
| .blur(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกเกิดอีเวนต์เสียโฟกัสแป้นคีย์บอร์ด |
| .change(handler) | กำหนดให้ฟังชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกถูกเปลี่ยนแปลงค่า |
| .click(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อคลิกเมาส์ให้อิลิเมนต์ที่ถูกเลือก |
| .dbclick(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อดับเบิลคลิกเมาส์ในอิลิเมนต์ที่ถูกเลือก |
| .error(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกเกิดอีเวนต์ error (ขึ้นอยู่กับเบราเซอร์แต่ละรุ่น) |
| .focus(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกได้รับโฟกัสของคีย์บอร์ด |
| .keydown(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อมีการกดปุ่มบนแป้นคีย์บอร์ดในอิลิเมนต์ที่ถูกเลือก (อิลิเมนต์ต้องได้รับโฟกัสแป้นคีย์บอร์ด) |
| **เมธอดสำหรับอีเวนต์** | **คำอธิบาย** |
| .keypress(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อมีการเคาะแป้นคีย์บอร์ด (กดปุ่มบนแป้นคีย์บอร์ดแล้วปล่อย) ในอิลิเมนต์ที่ถูกเลือก (อิลิเมนต์ต้องได้รับโฟกัสแป้นคีย์บอร์ด) |
| .keyup(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อปล่อยปุ่มกดบนแป้นคีย์บอร์ดในอิลิเมนต์ที่ถูกเลือก (อิลิเมนต์ต้องได้รับโฟกัสแป้นคีย์บอร์ด) |
| .load(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเลือกโหลดคอนเทนต์เสร็จเรียบร้อย |
| .mousedown(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อคลิกปุ่มเมาส์ภายในพื้นที่ของอิลิเมนต์ที่ถูกเลือก |
| .mouseenter(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนเข้าสู่พื้นที่ของอิลิเมนต์ที่ถูกเลือก และไม่ทำให้เกิดการลอยขึ้นของอีเวนต์ (event bubbling) |
| .mouseleave(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนออกจากพื้นที่ของอิลิเมนต์ที่ถูกเลือก และไม่ทำให้เกิดการลอยขึ้นของอีเวนต์ (event bubbling) |
| .mousemove(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนอยู่ภายในพื้นที่ของอิลิเมนต์ที่ถูกเลือก |
| .mouseout(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนออกจากพื้นที่อิลิเมนต์ที่ถูกเลือก |
| .mouseover(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนเข้าสู่พื้นที่อิลิเมนต์ที่ถูกเลือก |
| .mouseup(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อปล่อยปุ่มกดเมาส์ภายในอิลิเมนต์ที่ถูกเลือก |
| .resize(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อปรับขนาดหน้าต่างของเบราเซอร์ |
| .scroll(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อมีการเลื่อนสกรอลล์บาร์ |
| .select(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเลือกข้อความภายในอิลิเมนต์ที่ถูกเลือก |
| .submit(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ฟอร์มที่ถูกเลือกถูกซับมิตฟอร์ม |
| .unload(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่ออิลิเมนต์ที่ถูกเรียกถูกลบออกจากหน่วยความจำ |
| **เมธอดสำหรับอีเวนต์** | **คำอธิบาย** |
| .hover(enter,leave) | กำหนดให้ฟังก์ชั่น enter ถูกเรียกเมื่อเคอร์เซอร์ของเมาส์เลื่อนอยู่เหนือพื้นที่อิลิเมนต์ที่ถูกเลือก และเรียกฟังก์ชั่น leave เมื่อเคอร์เซอร์ของเมาส์เลื่อนออกจากพื้นที่อิลิเมนต์ที่เลือก |
| .toggle(handler1,handler2) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อคลิกเมาส์บนอิลิเมนต์ตามด้วยเรียกฟังก์ชั่น handler2 ในการคลิกเมาส์ครั้งต่อไป และเรียกฟังก์ชั่นอื่น (ถ้ามี) ในการคลิกครั้งต่อไปอีก (เมื่อเรียกครบแล้วก็วนกลับไปที่ฟังก์ชั่นแรกใหม่) |
| .trigger(type,[data]) | ระบุชนิดอีเวนต์เพื่อเรียกฟังก์ชั่นจัดการอีเวนต์ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน และทำให้อิลิเมนต์เกิดการทำงานแบบดีฟอลต์ |
| .triggerHandler(type,[data]) | ระบุชนิดอีเวนต์เพื่อเรียกฟังก์ชั่นจัดการอีเวนต์ของอิลิเมนต์ที่ถูกเลือกทำงาน แต่อิลิเมนต์จะไม่เกิดการทำ งานแบบดีฟอลต์ |
| .blur() | เรียกฟังก์ชั่นจัดการอีเวนต์ blur ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .change() | เรียกฟังก์ชั่นจัดการอีเวนต์ change ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .click() | เรียกฟังก์ชั่นจัดการอีเวนต์ click ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .dbclick() | เรียกฟังก์ชั่นจัดการอีเวนต์ dbclick ของอิลิเมนตีถูกเลือกให้ทำงาน |
| .error() | เรียกฟังก์ชั่นจัดการอีเวนต์ error ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .focus() | เรียกฟังก์ชั่นจัดการอีเวนต์ focus ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .keydown() | เรียกฟังก์ชั่นจัดการอีเวนต์ keydown ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .keypress() | เรียกฟังก์ชั่นจัดการอีเวนต์ keypress ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .keyup() | เรียกฟังก์ชั่นจัดการอีเวนต์ keyup ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .select() | เรียกฟังก์ชั่นจัดการอีเวนต์ select ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |
| .submit() | เรียกฟังก์ชั่นจัดการอีเวนต์ submit ของอิลิเมนต์ที่ถูกเลือกให้ทำงาน |

**เมธอดสำหรับเอฟเฟ็กต์**

เมธอดสำหรับเอฟเฟ็กต์ (effect method) เหล่านี้อาจถูกนำมาใช้สำหรับสร้างแอนิเมชั่นให้กับอิลิเมนต์ได้

|  |  |
| --- | --- |
| **เมธอดสำหรับเอฟเฟ็กต์** | **คำอธิบาย** |
| .show() | แสดงอิลิเมนต์ที่ถูกเลือกบนหน้าจอ |
| .hide() | ซ่อนอิลิเมนต์ที่ถูกเลือกจากหน้าจอ |
| .show(speed,[callback]) | แสดงอิลิเมนต์ที่ถูกเลือกบนหน้าจอ โดยแสดงภาพเคลื่อนไหวจากการเปลี่ยนค่าพร็อปเพอร์ตี้ height, width และ opacity |
| .hide(speed,[callback]) | ซ่อนอิลิเมนต์ที่ถูกเลือกโดยแสดงภาพเคลื่อนไหวจากการเปลี่ยนค่าพร็อปเพอร์ตี้ height, width และ opacity |
| .toggle([speed],[callback]) | แสดงหรือซ่อนอิลิเมนต์ที่ถูกเลือกบนหน้าจอ |
| .slideDown([speed],[callback]) | แสดงอิลิเมนต์ที่ถูกเลือกด้วยเอฟเฟ็กต์ภาพเลื่อนลง |
| .slideUp([speed],[callback]) | ซ่อนอิลิเมนต์ที่ถูกเลือกด้วยเอฟเฟ็กต์ภาพเลื่อนขึ้น |
| .slideToggle([speed],[callback]) | แสดงหรือซ่อนอิลิเมนต์ที่ถูกเลือกด้วยเอฟเฟ็กต์ภาพเลื่อน |
| .fadeIn([speed],[callback]) | แสดงอิลิเมนต์ที่ถูกเลือกโดยเปลี่ยนจากภาพจางเป็นภาพปกติ |
| .fadeOut([speed],[callback]) | ซ่อนอิลิเมนต์ที่ถูกเลือกโดยเปลี่ยนจากภาพปกติเป็นภาพโปร่งใส |
| .fadeTo(speed,opacity,[callback]) | ปรับค่าความทึบแสงให้อิลิเมนต์ที่ถูกเลือก |
| .animate(attributes,[speed],[easing],[callback]) | ทำให้อิลิเมนต์ที่ถูกเลือกแสดงคัสตอมแอนิเมชั่นโดยระบุแอตทริบิวต์ CSS |
| .animate(attributes,options) | เป็นอินเตอร์เฟซระดับล่างของเมธอด .animate () ที่เราสามารถระบุออปชั่นควบคุมการจัดคิวแอนิเมชั่นได้ |
| .stop([clearQueue],[jumpToEnd]) | หยุดแสดงแอนิเมชั่นที่กำลังทำงานอยู่ และรันแอนิเมชั่นถัดไป (ถ้ามี) |
| .queue() | อ่านข้อมูลคิวแอนิเมชั่นของอิลิเมนต์ตัวแรกที่ถูกเลือก |
| **เมธอดสำหรับเอฟเฟ็กต์** | **คำอธิบาย** |
| .queue(callback) | เพิ่มฟังก์ชั่น callback ไปต่อท้ายคิวแอนิเมชั่น |
| .queue(newQueue) | แทนที่คิวเดิมด้วยคิวใหม่ให้อิลิเมนต์ที่ถูกเลือก |
| .dequeue() | รันแอนิเมชั่นตัวถัดไปที่อยู่ในคิว |

**เมธอดสำหรับจัดการ DOM**

|  |  |
| --- | --- |
| **เมธอดสำหรับจัดการ DOM** | **คำอธิบาย** |
| .attr(key) | อ่านค่าของแอตทริบิวต์ที่ชื่อ key |
| .attr(key,value) | กำหนดค่าของแอตทริบิวต์ key ให้เป็น value |
| .attr(key,fn) | กำหนดค่าของแอตทริบิวต์ key ให้เป็นค่าที่ฟังก์ชั่น fn รีเทิร์น (เรียกฟังก์ชั่นแยกจากกันในแต่ละอิลิเมนต์ที่ถูกเลือก) |
| .attr(map) | กำหนดค่าของแอตทริบิวต์ด้วยคู่ของคีย์-ค่าข้อมูล |
| .removeAttr(key) | ลบแอตทริบิวต์ที่ชื่อ key |
| .addClass(class) | เพิ่มคลาส class ไปยังอิลิเมนต์ที่ถูกเลือก |
| .removeClass(class) | ลบคลาส class ออกจากอิลิเมนต์ที่ถูกเลือก |
| .toggleClass(class) | ลบคลาส class ออกจากอิลิเมนต์ที่ถูกเลือก (ถ้ามีคลาสนี้), เพิ่มคลาส class ไปยังอิลิเมนต์ที่ถูกเลือก (ถ้าไม่มีคลาสนี้) |
| .hasClass(class) | หากอิลิเมนต์ที่ถูกเลือกมีการกำหนดคลาสชื่อ class ให้รีเทิร์น true |
| .html() | อ่านคอนเทนต์ HTML ที่อยู่ภายในอิลิเมนต์แรกที่ถูกเลือก |
| .html(value) | กำหนดคอนเทนต์ HTML ให้แต่ละอิลิเมนต์ที่ถูกเลือกเป็น value |
| .text() | อ่านคอนเทนต์ข้อความของอิลิเมนต์ทั้งหมดที่ถูกเลือกในรูปแบบสตริงตัวเดียว |
| **เมธอดสำหรับจัดการ DOM** | **คำอธิบาย** |
| .text(value) | กำหนดคอนเทนต์ข้อความให้แต่ละอิลิเมนต์ที่ถูกเลือกมีค่าเป็น value |
| .val() | อ่านค่าแอตทริบิวต์ value จากอิลิเมนต์ตัวแรกที่ถูกเลือก |
| .val(value) | กำหนดค่าแอตทริบิวต์ value ให้แต่ละอิลิเมนต์ที่ถูกเลือก |
| .css(key) | อ่านค่าแอตทริบิวต์ CSS ที่ชื่อว่า key จากอิลิเมนต์ที่ถูกเลือก |
| .css(key,value) | กำหนดให้แอตทริบิวต์ CSS ที่ชื่อว่า key มีค่าเป็น value |
| .css(map) | กำหนดค่าแอตทริบิวต์ CSS โดยระบุคู่ของคีย์-ค่าข้อมูล ให้แต่ละอิลิเมนต์ที่ถูกเลือก |
| .offset() | อ่านค่าพร็อปเพอร์ตี้ top และ left ซึ่งเป็นค่าพิกัดตำแหน่งในหน่วยพิกเซลจากอิลิเมนต์แรกที่ถูกเลือก ที่สัมพันธ์กับหน้าจอ |
| .position() | อ่านค่าพร็อปเพอร์ตี้ top และ left ซึ่งเป็นค่าพิกัดตำแหน่งในหน่วยพิกเซลจากอิลิเมนต์ที่ถูกรีเทิร์นโดยเมธอด .offestParent () |
| .scrollTop() | อ่านค่าตำแหน่งของสกรอลล์บาร์ในแนวตั้งจากอิลิเมนต์แรกที่ถูกเลือก |
| .scrollTop(value) | กำหนดตำแหน่งของสกรอลล์บาร์ในแนวตั้งเป็น value ให้อิลิเมนต์ทั้งหมดที่ถูกเลือก |
| .scrollLeft() | อ่านค่าตำแหน่งของสกรอลล์บาร์ในแนวนอนจากอิลิเมนต์แรกที่ถูกเลือก |
| .scrollLeft(value) | กำหนดตำแหน่งของสกรอลล์บาร์ในแนวนอนเป็น value ให้อิลิเมนต์ทั้งหมดที่ถูกเลือก |
| .height() | อ่านค่าความสูงของอิลิเมนต์ตัวแรกที่ถูกเลือก |
| .height(value) | กำหนดความสูงให้อิลิเมนต์ทั้งหมดที่ถูกเลือกเป็น value |
| .width() | อ่านค่าความกว้างจากอิลิเมนต์ตัวแรกที่ถูกเลือก |
| .width(value) | กำหนดความกว้างเป็น value ให้อิลิเมนต์ทั้งหมดที่ถูกเลือก |
| .innerHeight() | อ่านค่าความสูงของอิลิเมนต์ตัวแรกที่ถูกเลือก โดยรวมค่า padding |
| .innerWidth() | อ่านค่าความกว้างที่รวมค่าpaddingแต่ไม่รวมความกว้างของขอบจากอิลิเมนต์แรกที่ถูกเลือก |
| **เมธอดสำหรับจัดการ DOM** | **คำอธิบาย** |
| .outerHeight(includeMargin) | อ่านค่าความสูง ที่รวมค่า padding, ความสูงของขอบ และค่า margin จากอิลิเมนต์แรกที่ถูกเลือก |
| .outerWidth(includeMargin) | อ่านค่าความกว้าง ที่รวมค่า padding, ความกว้างของขอบและค่า margin จากอิลิเมนต์ที่ถูกเลือก |
| .append(content) | แทรก content ไปที่ส่วนท้ายของคอนเท็นต์เดิมที่อยู่ภายในอิลิเมนต์ที่ถูกเลือก |
| .appendTo(selector) | นำอิลิเมนต์ถูกเลือกไปแทรกที่ส่วนท้ายของคอนเท็นตภายในอิลิเมนต์ที่ถูกเลือกโดย selector |
| .prepend(content) | แทรก content ไปที่ส่วนบนของคอนเท็นต์เดิมที่อยู่ภายในแต่ละอิลิเมนต์ที่ถูกเลือก |
| .prependTo(selector) | นำอิลิเมนต์ที่ถูกเลือกไปแทรกที่ส่วนบนของคอนเท็นต์ภายในอิลิเมนต์ที่ถูกเลือกโดย selector |
| .after(content) | แทรก content ไว้หลังอิลิเมนต์ที่ถูกเลือก |
| .insertAffer(selector) | นำอิลิเมนต์ที่ถูกเลือกไปแทรกไว้หลังอิลิเมนต์ที่ถูกเลือกโดย selector |
| .before(content) | แทรก content ไว้ก่อนหน้าอิลิเมนต์ที่ถูกเลือก |
| .insertBefore(selector) | นำอิลิเมนต์ที่ถูกเลือกไปไว้ก่อนหน้าอิลิเมนต์ถูกเลือกโดย selector |
| .wrap(content) | ครอบแต่ละอิลิเมนต์ที่ถูกเลือกไว้ภายใน content |
| .wrapAll(content) | ครอบอิลิเมนต์ทั้งกลุ่มไว้ภายใน content |
| .wrapInner(content) | ครอบคอนเท็นต์ภายในแต่ละอิลิเมนต์ที่ถูกเลือกด้วย content |
| .replaceWith(content) | แทนที่อิลิเมนต์แต่ละตัวที่ถูกเลือกด้วย content |
| .replaceAll(selector) | นำกลุ่มของอิลิเมนต์ที่ถูกเลือกไปแทนที่กลุ่มของอิลิเมนต์ที่ถูกเลือกโดย selector |
| .empty() | ลบอิลิเมนต์ลูกของอิลิเมนต์ที่ถูกเลือก |
| .remove([selector]) | ลบกลุ่มของอิลิเมนต์ที่ถูกเลือกออกจาก DOM (มีออปชั่นให้ทำฟิลเตอร์อิลิเมนต์ที่ถูกเลือกได้โดยระบุ selector) |
| **เมธอดสำหรับจัดการ DOM** | **คำอธิบาย** |
| .clone([withHandlers]) | ก๊อปปี้อิลิเมนต์ทุกตัวที่ถูกเลือก, มีออปชั่นให้กำหนดว่าจะก๊อปปี้ฟังก์ชั่นจัดการอีเวนต์ด้วยหรือไม่ |
| .data(key) | อ่านค่าข้อมูลชื่อ key ที่เก็บไว้ในอิลิเมนต์แรกที่ถูกเลือก |
| .data(key,value) | กำหนดข้อมูลชื่อ key มีค่าเป็น value เก็บไว้ในแต่ละอิลิเมนต์ที่ถูกเลือก |
| .removeData(key) | ลบข้อมูลชื่อ key เก็บไว้ในแต่ละอิลิเมนต์ที่ถูกเลือก |

**เมธอดสำหรับ AJAX**

เราสามารถรับข้อมูลจากเซิร์ฟเวอร์โดยไม่ต้องรีเฟรชหน้าจอ โดยการใช้เมธอดสำหรับ AJAX เหล่านี้

|  |  |
| --- | --- |
| **เมธอดสำหรับ AJAX** | **คำอธิบาย** |
| $.ajax(options) | กำหนดออปชั่นให้กับเมธอดที่สร้างรีเควสต์ AJAX เป็นเมธอดที่ทำงานในระดับล่างซึ่งมักจะถูกเรียกผ่านเมธอดอื่น |
| .load(url,[data],[callback]) | สร้างรีเควสต์ AJAX ไปที่ url และนำข้อมูลที่ตอบกลับมาไปวางไว้ในอิลิเมนต์ที่ถูกเลือก |
| $.get(url,[data],[callback],[returnType]) | สร้างรีเควสต์ AJAX ไปที่ url โดยใช้รูปแบบ GET |
| $.getJson(url,[data],[callback],[callback]) | สร้างรีเควสต์ AJAX ไปที่ url และแปลงข้อมูลที่ตอบกลับมาให้อยู่ในโครงสร้างข้อมูล JSON |
| $.getScript(url,[callback]) | สร้างรีเควสต์ AJAX ไปที่ url และรันข้อมูลที่ตอบกลับมาเป็นโค้ดจาวาสคริปต์ |
| $.post(url,[data],[callback],[returnType]) | สร้างรีเควสต์ AJAX ไปที่ url โดยใช้รูปแบบ POST |
| .ajaxComplete(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อการติดต่อ AJAX เสร็จเรียนร้อย (ไม่สนใจว่าการติดต่อจะสำเร็จหรือว่าข้อมูลที่ได้รับจะอยู่ในรูปแบบที่ถูกต้องหรือไม่) |
| **เมธอดสำหรับ AJAX** | **คำอธิบาย** |
| .ajaxError(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อรีเควสต์ AJAX มีข้อผิดพลาดเกิดขึ้น |
| .ajaxSend(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเริ่มส่งรีเควสต์ AJAX |
| .ajaxStart(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อเริ่มส่งรีเควสต์ AJAX และไม่มีรีเควสต์อื่นกำลังทำงานอยู่ |
| .ajaxStop(handler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อรีเควสต์ AJAX สิ้นสุดลง และไม่มีรีเควสต์อื่นกำลังประมวลผลอยู่ |
| .ajaxSuccess(hanndler) | กำหนดให้ฟังก์ชั่น handler ถูกเรียกเมื่อรีเควสต์ AJAX ทำงานสำเร็จโดยไม่มีข้อผิดพลาดเกิดขึ้น |
| $.ajaxSetup(options) | กำหนดออปชั่นดีฟอลต์สำหรับเมธอด AJAX ที่จะถูกเรียกในภายหลัง |
| .serialize() | แปลงค่าในคอนโทรลฟอร์มให้เป็นคิวรีสตริง |
| .serializeArray() | แปลงค่าในคอนฌทรลฟอร์มให้เป็นโครงสร้างข้อมูล JSON |
| $.param(map) | แปลงค่าที่อยู่ในออปเจ็กต์ให้เป็นิวรีสตริง |

**เมธอดเบ็ดเตล็ด**

เมธอดยูทิลิตี้เหล่านี้ไม่เข้ากับประเภทเมธอดที่ผ่านมา แต่มักจะมีประโยชน์มากเมื่อเขียนสคริปต์โดยใช้ jQuery

|  |  |
| --- | --- |
| **เมธอดหรือชื่อพร็อปเพอร์ตี้** | **คำอธิบาย** |
| $.support | รีเทิร์นออบเจ็กต์แมปสำหรับพร็อพเพอร์ตี้ที่บอกให้ทราบว่าเบราเซอร์สนับสนุนคุณสมบัติและมาตรฐานหลายๆ อย่างหรือไม่ |
| $.each(collection,callback) | วนซ้ำข้อมูลที่อยู่ใน collection และรันฟังก์ชั่น callback ในข้อมูลแต่ละรายการ |
| $.extend(target,addition,…) | แก้ไขออบเจ็กต์ target โดยการเพิ่มพร็อปเพอร์ตี้จากออบเจ็กต์อื่นที่ส่งมาให้ |
| $.grep(array,callback) | ฟิลเตอร์ข้อมูลใน array โดยใช้ฟังก์ชั่น callback เป็นตัวทดสอบ |
| **เมธอดหรือชื่อพร็อปเพอร์ตี้** | **คำอธิบาย** |
| $.makeArray(object) | แปลง object เป็นข้อมูลอาร์เรย์ |
| $.map(array,callback) | สร้างข้อมูลในอาร์เรย์ขึ้นมาใหม่จากการเรียกฟังก์ชั่น callback ในข้อมูลแต่ละรายการ |
| $.inArray(value,array) | ตรวจสอบว่ามีค่า value อยู่ในข้อมูล array หรือไม่ |
| $.merge(array1,array2) | รวมข้อมูลอาร์เรย์ของ array1 และ array2 เข้าด้วยกัน |
| $.unique(array) | ลบอิลิเมนต์ DOM ที่ซ้ำกันออกจาก array |
| $.isFunction(object) | ตรวจสอบว่า object ป็นฟังก์ชั่นหรือไม่ |
| $.trim(sting) | ลบพื้นที่ว่างระหว่างตัวอักษรที่อยู่ด้านหน้าและด้านหลังของ string |
| $.noConflict([extreme]) | คืนการควบคุมการใช้ชื่อย่อ $ ให้ไลบรารีอื่น |
| .hasClass(className) | ตรวจสอบว่าอิลิเมนต์ที่ถูกเลือกนั้นมีการกำหนดคลาสที่ระบุไว้หรือไม่ |
| .is(selector) | ตรวจสอบอิลิเมนต์ที่ถูกเลือกว่าตรงกับ selector expression ที่ระบุไว้หรือไม่ |
| .each(callback) | การวนซ้ำบนอิลิเมนต์ที่ถูกเลือกและรันฟังก์ชั่น callback ในแต่ละอิลิเมนต์ |
| .ength | อ่านจำนวนของอิลิเมนต์ที่ถูกเลือก |
| .get() | อ่านข้อมูลอาร์เรย์ของอิลิเมนต์ DOM ที่ตรงกับอิลิเมนต์ที่ถูกเลือก |
| .get(index) | อ่านข้อมูลอิลิเมนต์ DOM ที่ตรงกับอิลิเมนต์ที่ถูกเลือกในตำแหน่งที่ระบุค่าอินเด็กซ์ |
| .index(element) | อ่านค่าอินเด็กซ์ของอิลิเมนต์ DOM ที่อยู่ภายในกลุ่มอิลิเมนต์ที่ถูกเลือก |