

AngularJS

Basic to Professional



Control

C



Tavon Seesenpila

Chapter 0

พื้นฐาน AngularJS และ MVC

แนะนำ AngularJS

AngularJS นั้นเป็น Open Source Java Script Framework ที่ออกแบบ และพัฒนาโดยบริษัท Google มีความโดดเด่น ในเรื่องการจัดการโค้ด ให้เป็นระบบ และทำงานได้อย่างรวดเร็วมาก เมื่อเทียบกับ Java Script Framework อื่นๆ และนอกจากนี้แล้ว AngularJS ก็ยังมีความยืดหยุ่นที่สูงมาก ทำให้เขียนโค้ด ในแบบต่างๆ ได้อย่างง่ายดาย

ในหนังสือเล่มนี้ได้เขียนอธิบายเกี่ยวกับการใช้งาน AngularJS Framework อย่างละเอียด ทีละขั้นตอน โดยเน้นการนำเสนอแบบ “ทำตามได้” ด้วยวิธีที่ง่ายที่สุด ดังนั้นจึงเหมาะสมกับผู้อ่านที่เพิ่งเริ่มต้น และต้องการศึกษาไปจนถึงระดับสูง เพื่อให้เข้าใจถึงการนำไปใช้งานจริง เราจึงได้เขียนส่วนของ Workshop ไว้อีกซึ่งจะอยู่ในท้ายเล่มครับ

พื้นฐานผู้อ่าน

ก่อนอื่นขอบอกก่อนว่า ผู้อ่านจะต้องมีพื้นฐานเรื่องต่างๆ มา ก่อน เพื่อให้สามารถอ่านหนังสือเล่มนี้ได้อย่างรวดเร็ว และมีประสิทธิภาพ โดยมีเรื่องหลักๆ ดังนี้

1. HTML
2. SQL
3. PHP
4. CSS

สิ่งที่จะได้รับเมื่ออ่านหนังสือเล่มนี้จบ

1. มีความรู้ความเข้าใจในการใช้งาน AngularJS
2. นำความรู้ที่ได้ไปต่อยอดในระดับสูงต่อไปได้
3. สามารถพัฒนาระบบ Frontend System ด้วย AngularJS ได้
4. พัฒนาโปรแกรมระบบฐานข้อมูลกับ MySQL ได้
5. ประยุกต์พัฒนาระบบกับภาษา PHP ได้

รูปแบบการนำเสนอของเรา

หนังสือเล่มนี้จะนำเสนอในรูปแบบของการอธิบายทีละขั้นตอน step by step และจากนั้นถ่ายทอดความรู้พื้นฐาน จนถึงระดับสูง โดยคัดเลือกสิ่งที่จำเป็นต้องรู้เท่านั้น (เพราะไม่สามารถอธิบายทุกคำล้วงได้หมด) และตามด้วยการพาประยุกต์ทำระบบแบบ workshop เพื่อให้เห็นภาพของการนำไปใช้งานจริง

และการอธิบาย โค้ดคำสั่งต่างๆ เราจะอธิบายเอาไว้ในโค้ดเลย ไม่ได้แยกมาอธิบายต่างหาก จากนั้นคำสั่งไหนที่ได้อธิบายไปแล้ว เรายังจะยังคงนำมาอธิบายต่อไปอีก ช้าๆ เพื่อให้ผู้อ่าน สามารถข้ามไปมา ระหว่างบทต่างๆ ได้ (ในกรณีที่อ่านจบไปแล้ว อยากกลับมาทบทวน เวลาทำงานจริง) จะได้สะดวกในการทำความเข้าใจ

Download ติดตั้ง และทดสอบ

ไปที่เว็บไซต์ angularjs.org และทำการดาวน์โหลดไฟล์มา

The screenshot shows the official AngularJS website at <https://angularjs.org>. At the top, there's a navigation bar with links for Home, Learn, Develop, Discuss, and a search bar. The main content features the AngularJS logo and the tagline "HTML enhanced for web apps!". Below this are three buttons: "View on GitHub", "Download (1.4.3 / 1.2.28)", and "Design Docs & Notes". A red arrow points from a callout bubble labeled "ดาวน์โหลดที่นี่" (Download here) to the "Download" button. Social sharing links for Google+, Twitter, and Facebook are also present.

จะพบกับหน้าจอให้เราเลือกดาวน์โหลดไฟล์

The screenshot shows the "Download AngularJS" page. It has several configuration options: "Branch" (set to 1.4.x (stable)), "Build" (set to Zip), "CDN" (Unavailable for zip archives), "Bower" (bower install angular#1.4.3), "npm" (npm install angular@1.4.3), and "Extras" (Browse additional modules). A red arrow points from a callout bubble labeled "เลือกเป็นแบบ Zip" (Select as Zip) to the "Zip" button under the Build section. Another red arrow points from a callout bubble labeled "กดดาวน์โหลด" (Press download) to the large blue "Download" button at the bottom right. A link to "Previous Versions" is also visible.

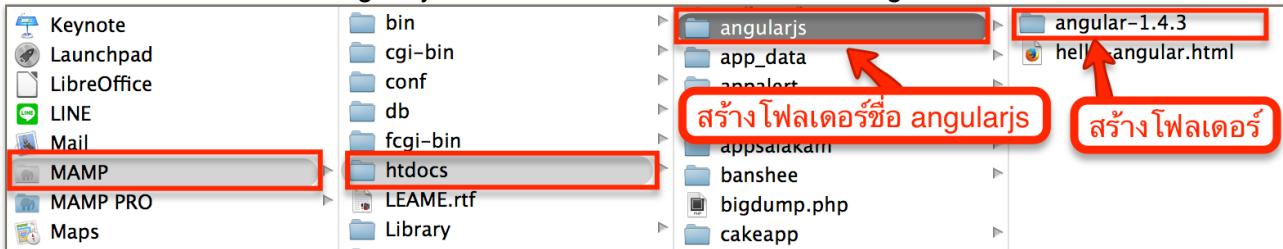
จะได้ไฟล์มาดังนี้

angular-1.4.3.zip	Today, 2:09 AM	9.8 MB	ZIP archive
-------------------	----------------	--------	-------------

ทำการแตกไฟล์ จะพบว่าข้างในมีไฟล์ต่างๆ มากมาย ดังนี้

angular-animate.js	7/14/2558 BE, 6:53 PM	134 KB	JavaScript
angular-animate.min.js	7/14/2558 BE, 6:53 PM	23 KB	JavaScript
angular-animate.min.js.map	7/14/2558 BE, 6:53 PM	64 KB	Document
angular-aria.js	7/14/2558 BE, 6:53 PM	14 KB	JavaScript
angular-aria.min.js	7/14/2558 BE, 6:53 PM	4 KB	JavaScript
angular-aria.min.js.map	7/14/2558 BE, 6:53 PM	8 KB	Document
angular-cookies.js	7/14/2558 BE, 6:53 PM	10 KB	JavaScript
angular-cookies.min.js	7/14/2558 BE, 6:53 PM	1 KB	JavaScript
angular-cookies.min.js.map	7/14/2558 BE, 6:53 PM	3 KB	Document
angular-csp.css	7/14/2558 BE, 6:53 PM	343 bytes	CSS
angular-loader.js	7/14/2558 BE, 6:53 PM	16 KB	JavaScript
angular-loader.min.js	7/14/2558 BE, 6:53 PM	2 KB	JavaScript
angular-loader.min.js.map	7/14/2558 BE, 6:53 PM	4 KB	Document
angular-message-format.js	7/14/2558 BE, 6:53 PM	38 KB	JavaScript
angular-message-format.min.js	7/14/2558 BE, 6:53 PM	10 KB	JavaScript
angular-message-format.min.js.map	7/14/2558 BE, 6:53 PM	28 KB	Document
angular-messages.js	7/14/2558 BE, 6:53 PM	25 KB	JavaScript
angular-messages.min.js	7/14/2558 BE, 6:53 PM	3 KB	JavaScript
angular-messages.min.js.map	7/14/2558 BE, 6:53 PM	7 KB	Document
angular-mocks.js	7/14/2558 BE, 6:53 PM	82 KB	JavaScript
angular-resource.js	7/14/2558 BE, 6:53 PM	27 KB	JavaScript
angular-resource.min.js	7/14/2558 BE, 6:53 PM	4 KB	JavaScript
angular-resource.min.js.map	7/14/2558 BE, 6:53 PM	9 KB	Document
angular-route.js	7/14/2558 BE, 6:53 PM	36 KB	JavaScript
angular-route.min.js	7/14/2558 BE, 6:53 PM	4 KB	JavaScript
angular-route.min.js.map	7/14/2558 BE, 6:53 PM	11 KB	Document
angular-sanitize.js	7/14/2558 BE, 6:53 PM	25 KB	JavaScript
angular-sanitize.min.js	7/14/2558 BE, 6:53 PM	6 KB	JavaScript
angular-sanitize.min.js.map	7/14/2558 BE, 6:53 PM	11 KB	Document
angular-scenario.js	7/14/2558 BE, 6:53 PM	1.4 MB	JavaScript
angular-touch.js	7/14/2558 BE, 6:53 PM	23 KB	JavaScript
angular-touch.min.js	7/14/2558 BE, 6:53 PM	4 KB	JavaScript
angular-touch.min.js.map	7/14/2558 BE, 6:53 PM	10 KB	Document
angular.js	7/14/2558 BE, 6:53 PM	1 MB	JavaScript
angular.min.js	7/14/2558 BE, 6:53 PM	145 KB	JavaScript
angular.min.js.map	7/14/2558 BE, 6:53 PM	392 KB	Document
docs	7/14/2558 BE, 6:54 PM	--	Folder
errors.json	7/14/2558 BE, 6:53 PM	9 KB	JSON
i18n	7/14/2558 BE, 6:54 PM	--	Folder
version.json	7/14/2558 BE, 6:54 PM	302 bytes	JSON
version.txt	7/14/2558 BE, 6:54 PM	5 bytes	Plain Text

ในการติดตั้ง ใช้งานนั้นไม่ยากครับ ให้ผู้อ่านลง โปรแกรม XAMPP หรือ Web Server ได้ ก็ได้ เช่น MAMPP, Appserv, WAMP, AMPP ก็ได้
 จากนั้น ให้ไปที่ web root (ในหนังสือนี้คือ mampp/htdocs/)
 และทำการสร้าง folder ชื่อ angularjs จากนั้นสร้าง folder ย่อยอีก ชื่อว่า angular-1.4.3 ดังภาพนี้



นำไฟล์ทั้งหมดที่โหลดมา (ที่เห็น曳ะๆ ในภาพก่อนหน้านี้) ไปวางใน angular-1.4.3 ดังนี้



สร้างไฟล์ hello-angular.html ดังในภาพนี้ และเขียนโค้ดลงไปไฟล์ดังนี้

hello-angular.html

```
<!DOCTYPE html>
<!-- คำว่า ng-app เป็น directive คือ เป็นการประกาศว่า เอกสารนี้จะเป็น Angular App -->
<html ng-app>
<head>
    <!-- เชื่อมโยง library ของ angularjs เข้ากับงานของเรา -->
    <script src="angular-1.4.3/angular.js"></script>

    <meta charset="UTF-8">
    <title>First App for AngularJS</title>
</head>
<body>
    <h3>Example 1 : Test AngularJS</h3>

    <!-- คำว่า data-ng-model คือการสร้างตัวแปรไว้ให้ผูกกับข้อมูลชื่อ name -->
    <input type="text" data-ng-model="name">

    <!-- แสดงค่าตัวแปร name -->
    Name = {{ name }}
```

```
</body>  
</html>
```

```
<!--  
คือการคอมเม้นต์ ในภาษา HTML ผู้อ่านไม่ต้องพิมพ์ตามก็ได้ครับ เพียงแค่ ผม ใส่คำอธิบายของโค้ดแต่ละส่วน  
ไว้ จึงมีเครื่องหมายนี้  
-->
```

ผลการทำงาน

localhost/angularjs/hello-angular.html

Example 1 : Test AngularJS

 Name =

เมื่อทำการป้อนข้อมูลลงในช่อง input จะสังเกตุว่าค่าตัวแปรถูกนำมาแสดง ดังนี้

localhost/angularjs/hello-angular.html

Example 1 : Test AngularJS

 Tavon Name = Tavon

จุดสังเกต

- * หากเราไม่ใช้ ng-app เข้าไป คำสั่งต่างๆ ของ angular จะไม่ทำงาน
- * การแสดงค่าตัวแปร ใช้ {{ variable_name }}

แนะนำก่อนเข้าเนื้อหาจริง

ตัวอย่างก่อนหน้านี้ เป็นเพียงแค่การทดสอบว่า AngularJS นั้นทำงานในเครื่องเราได้หรือไม่ การติดตั้งถูกต้องตามขั้นตอนใหม่ แต่จากนี้ไป จะเป็นเนื้อหาจริงๆ และครับ ซึ่งผมจะเลือกนำเสนอ ในลักษณะค่าตาม และแนวทางค่าตอบ เพื่อให้เห็นว่า ในแต่ละหัวข้อ เราจะมีวิธีเขียนโค้ดอย่างไร พร้อมกับจะอธิบายไว้ทุknรรทัด และตอบท้ายด้วยการสรุปโค้ด ว่าส่วนไหน คืออะไร ทำอะไร ต้องเขียนค่าสั่งแบบนั้น

เราจะนำเอา Angular.JS มาใช้กับ HTML Page ได้อย่างไร

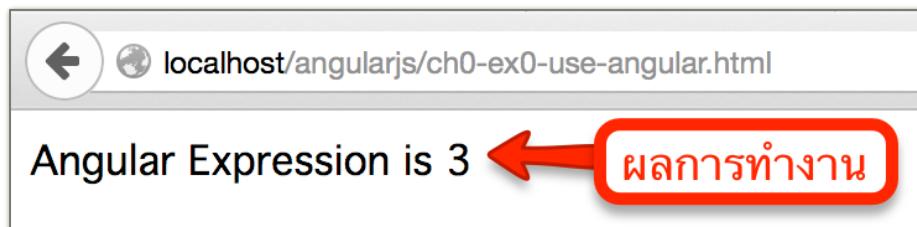
ในการที่เราจะเอาค่าสั่งของ Angular มาทำงานกับหน้าเว็บที่เป็นโค้ด HTML นั้น จะต้องมีการนำเข้าไฟล์ angular.js ก่อน ซึ่งเป็น javascript library จากนั้นเราจะต้องกำหนด ส่วนบนเอกสาร ให้เป็น ng-app เรียกว่า directive ครับ

ตัวอย่างโค้ด

ch0-ex0-use-angular.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
  </head>
  <body ng-app>
    <p>Angular Expression is {{ 1 + 2 }}</p>
  </body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

ในการใช้งาน Expression ของภาษา AngularJS นั้นจะใช้เครื่องหมาย {{ }} มาครอบเอาไว้ ในนี้สามารถทำการคำนวน และแสดงค่าตัวแปรต่างๆ ได้ตามต้องการ แต่มีข้อแม้ว่า ค่าสั่งของ Angular มันจะทำงานก็ต่อเมื่อ เรายังการเรียกใช้งาน ng-app เลยก่อน ซึ่งตัวอย่างนี้ เราได้ใส่เอาไว้ในส่วนของ tag body

หากว่าผู้อ่านເອົາສ່ວນຂອງ ng-app ອອກ คำสั่งของ AngularJS จะໄມ່ທຳງານໄດ້ ແຕ່ຈະແສດງຂໍ້ຄວາມຂອງ Expression นັ້ນໆ ອອກມາແກ່ ດັກພັນຕົວ



การเชื่อมโยง Text Input เข้ากับ Expression

หากต้องการเชื่อมโยงการทำงานระหว่าง Text Input เข้ากับ Expression ของ AngularJS สามารถทำได้ง่ายๆ ด้วยการเติม ng-model ใส่ลงใน Input ช่องนั้นๆ มันก็จะทำการไปอ่านค่า เมื่อถูกป้อนข้อมูล และทำให้ตัวแปรนั้นเปลี่ยนแปลงไป จากนั้นก็เอาไปแสดงผล ในส่วนที่เราวาง Expression {{ }} ไว้

ตัวอย่าง โค้ด

ch0-ex1-binding-text-input.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
  </head>
  <body ng-app>
    <!-- ผูกตัวแปร name เข้ากับช่อง text input -->
    <input type="text" ng-model="name" />

    <!-- แสดงตัวแปร name -->
    {{ name }}
  </body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

เมื่อมีการป้อนข้อมูลลงในช่อง Text Input ค่าตัวแปร name จะเปลี่ยนไป เพราะเราผูกเอาไว้ด้วยคำสั่ง ng-model="name" ดังนั้นส่วนที่เราวางไว้แสดงตัวแปร ก็จะทำการแสดงค่าตัวแปรออกมากันทันที ซึ่งก็คือจุดที่เราวาง {{ name }} ไว้นั่นเองครับ

ทั้งส่วนของ Text Input กับส่วนที่เป็น Expression มันจะเชื่อมกันเอง โดยอัตโนมัติ ซึ่งทำให้เราประหยัดเวลาในการเขียน โค้ดมากๆ ไปมาก เพราะไม่ต้องเขียนเกี่ยวกับการจัด event keypress เอง

Responding Click Event ด้วย Controllers

ตัวอย่างนี้จะเป็นการแยกส่วนการทำงานออกไป ให้มี event onclick และแยกการทำงานไปไว้ในส่วนที่เป็น Controller (การทำงานที่เป็น logic เท่านั้น ไม่เกี่ยวกับการแสดงผล) วิธีการคือจะทำการเติม ng-controller เข้าไปใน tag ที่เราจะกำหนดให้เป็น controller จากนั้นก็จะมีการใช้คำสั่ง ng-click เพื่อสร้าง event onclick ให้กับ tag html นั้นๆ และสุดท้ายกำหนดการซ่อนหรือแสดง ด้วย ng-show, ng-hide ได้เลย

เพื่อความเข้าใจหัวข้อนี้ ผมจะสาธิตตัวอย่างโค้ดง่ายๆ ที่มีการแยกส่วนการทำงานออกไป จากส่วนการแสดงผลนะครับ

ตัวอย่าง โค้ด

ch0-ex2-responding-click-events-using-controllers.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>

    <!-- เชื่อมโยงไปไฟล์ app.js -->
    <script src="app.js"></script>
  </head>

  <!-- ผูกเข้ากับ module ชื่อ myApp -->
  <body ng-app="myApp">

    <!-- เชื่อมเข้ากับ MyController -->
    <div ng-controller="MyController">

      <!-- เมื่อคลิกปุ่ม ให้ไปทำงานที่ function toggle() -->
      <button ng-click="toggle()">Toggle</button>

      <!-- แสดงค่า ตัวยการสั่ง visible -->
      <p ng-show="visible">Hello World!</p>
    </div>
  </body>
</html>
```

app.js

```
// สร้างตัวแปรชื่อ app กำหนดชื่อ module เป็น myApp
var app = angular.module('myApp', []);

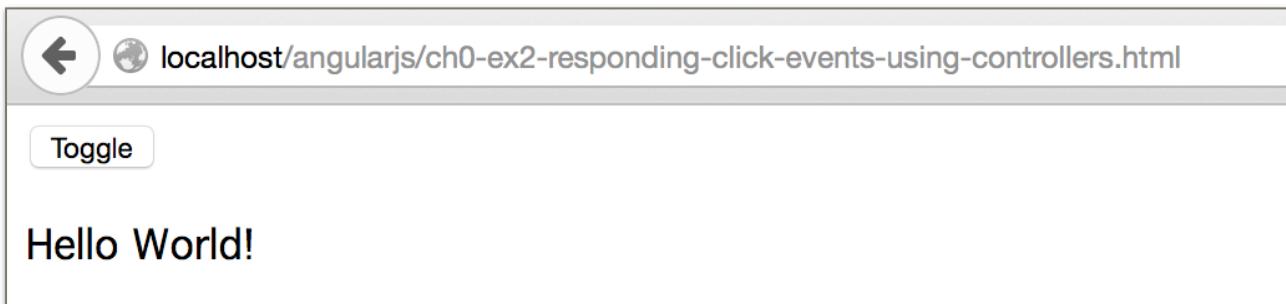
// สร้าง controller ชื่อ MyController
app.controller('MyController', function($scope) {

  // สร้างตัวแปร visible กำหนดค่าเป็น true
  $scope.visible = true;

  // กำหนดการทำงานแบบ toggle
  $scope.toggle = function() {

    // ให้สลับค่า เป็น true, false จากค่าเดิม
    $scope.visible = !$scope.visible;
  }
});
```

ผลการทำงาน



อธิบายเพิ่มเติม

สำหรับตัวอย่างนี้ผมจะอธิบายเป็นส่วนๆ เพราะค่อนข้างมากเหมือนกัน เรามาเริ่มกันที่ ในไฟล์ HTML นั้นเราให้มีการเชื่อมโยงไปที่ไฟล์ app.js จากนั้นลงมาอีกหน่อย ส่วนของ tag body ทำการกำหนดให้เชื่อมเข้ากับ module ชื่อว่า myApp ด้วยคำสั่ง ng-app="myApp" จากนั้นก็เริ่มสร้าง tag div กำหนดให้ผูกเข้ากับ controller ชื่อว่า MyController ด้วยคำสั่ง ng-controller="MyController"

และจากนั้นเราก็สร้าง button ให้มีการคลิก แล้วไปทำงานที่ function toggle ด้วยคำสั่ง ng-click="toggle()" สุดท้ายก่อนจบไฟล์นี้เราจะทำการสร้าง tag p ให้มีการซ่อน และแสดงได้ด้วยการสั่ง ng-show="visible"

ในส่วนของ app.js ผมขออธิบายเพิ่มเติมดังนี้

เริ่มแรกเราทำการสร้างตัวแปรชื่อ app และให้เป็น module ชื่อว่า myApp โดยการประกาศเป็น var app = angular.module('myApp', [])

จากนั้นเราทำการสร้าง controller ด้วยคำสั่ง app.controller('MyController', function(\$scope) โดยที่ \$scope นั้นจะถูกใช้งานใน function ในลำดับต่อไป

ในการทำงานของ function เราสั่งให้มีการสร้างตัวแปร visible = true ไว้ก่อน เพื่อเป็นค่าเริ่มต้น จากนั้นทำการเชื่อมโยงเข้ากับ function toggle โดยการทำงานภายใน function ก็คือ ให้สลับค่าของตัวแปร จาก true เป็น false และสลับจาก false เป็น true ด้วยการเขียน \$scope.visible = !\$scope.visible

แปลง Expression ด้วย Filter

ในการแสดงผลค่าตัวแปรผ่านทาง Expression เราสามารถลั่งงานให้ทำงานอื่นๆ ได้อีกมากมาย โดยผ่านเครื่องหมาย | หลังตัวแปรนั้นๆ เช่น {{ name | uppercase }} เพื่อแปลงให้ค่าตัวแปรกลายเป็นอักษรตัวใหญ่ทั้งหมด

ตัวอย่างโค้ด

ch0-ex3-convert-expression-by-filter.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
  </head>
  <body ng-app>
    <input type="text" ng-model="name">
    {{ name | uppercase }}
  </body>
</html>
```

ผลการทำงาน



สร้าง Custom Element ด้วย Directive

ในการสร้าง element ของ html นั้นสามารถทำได้จ่ายมากๆ เพราะว่า angular.js ออกแบบมาเพื่อการเขียนโปรแกรมด้าน frontend โดยเฉพาะ ซึ่งเราจะใช้ ng-model เข้ามาช่วย ร่วมกับการเขียนโค้ดที่เป็นส่วนของ angular code

ตัวอย่างโค้ด

ch0-ex4-create-custom-element-by-directive.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปรชื่อ app กำหนดเป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สั่งให้สร้าง directive ชื่อ show
      app.directive('show', function() {
        // เริ่มการทำงาน
        return {
          // เริ่มสร้าง function ของ element
          link: function(scope, element, attributes) {
            // กำหนดให้ลับค่าการแสดงผล เป็น '' กับ none
            scope.$watch(attributes.show, function(value) {
              element.css('display', value ? '' : 'none');
            });
          }
        }
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ moudle myApp -->
  <body ng-app="myApp">

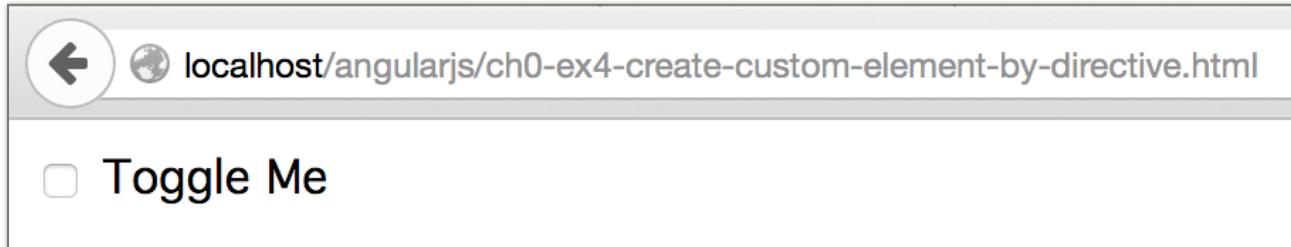
    <!-- สร้าง checkbox -->
    <label for="checkbox">

      <!-- สร้าง checkbox ผูกเข้ากับตัวแปร visible -->
      <input id="checkbox" type="checkbox" ng-model="visible">
      Toggle Me
    </label>
  </body>
</html>
```

```
</label>

<!-- สร้าง div และกำหนด element ชื่อ show ให้เป็นไปตามค่าตัวแปร visible -->
<div show="visible">
    <p>Hello World</p>
</div>
</body>
</html>
```

ผลการทำงาน



เมื่อกดที่ checkbox ค่าจะแสดงออกมา



อธิบายเพิ่มเติม

เป็นอีกตัวอย่างหนึ่งที่อธิบายยาวสักหน่อย เพราะตัวอย่างนี้คือการสร้าง attribute ของ html tag ขึ้นมา (ซึ่งมันไม่มีอยู่ใน html ปกติ) ในที่นี่เราสร้าง attribute ชื่อว่า show ขึ้นมา ให้มีค่าเป็นไปตามตัวแปร visible ที่นี่เรามาดูส่วนของ angular code กันครับ

เริ่มต้นเราทำการสร้างตัวแปรชื่อ app ให้เป็น module ชื่อว่า myApp

จากนั้นเราสร้าง directive ชื่อว่า show และทำการสร้างเป็น function ที่ต้องมีค่า parameter 3 ตัวตามลำดับดังนี้ function(scope, element, attributes)

สุดท้ายเราจะสั่งให้ scope.\$watch เพื่อไปอ่านค่า attribute ที่ชื่อ show โดยที่เราสั่งปรับเปลี่ยน css ของ element นั้นๆ (ในตัวอย่างผมทำการสั่งให้ ซ่อน กับ แสดง โดยอิงค่าตาม checkbox ที่เราคลิก)

อธิบาย Controller

สำหรับ controller ใน angularjs นั้นก็คือส่วนที่เป็นการทำงานในด้าน logic ของโปรแกรม อย่างเช่น การคำนวณ การสั่งแสดงผล หรือซ่อน ลิ้งต่างๆ บนหน้าจอ การควบคุมการ render ทุกอย่าง ไม่ว่าจะเป็น html, css เป็นต้น นอกจากนี้แล้ว controller ก็ยังเป็นสมอ่อน function ที่อยู่ภายใต้ javascript อีกชั้นหนึ่งด้วย

กำหนดค่าเริ่มต้นให้กับตัวแปรของ Controller

เราจะสามารถกำหนดค่าเริ่มต้นของตัวแปรต่างๆ ที่จะถูกใช้งานใน controller ได้อย่างไร โดยให้มันมีข้อมูลการทำงานภายใน controller นั้นๆ เท่านั้น เพื่อความเข้าใจวิธีการ มาลองดูด้วยค่าดังนี้กันครับ

ตัวอย่างโค้ด

ch0-ex5-assign-default-value-to-model.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อว่า myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ MyController
      app.controller('MyController', function($scope) {
        // กำหนดตัวแปรชื่อ value ให้มีค่าเป็น 'Tavon'
        $scope.value = 'Tavon';
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- ผูกเข้ากับ controller ชื่อ MyController -->
    <div ng-controller="MyController">

      <!-- แสดงค่าตัวแปรชื่อ value -->
      <p>{{ value }}</p>
    </div>

    <!-- ตัวแปรนี้จะไม่แสดง เพราะอยู่นอกขอบเขตของ Controller -->
    <p>{{ value }}</p>
  </body>
</html>
```

ผลการทำงาน

Tavon

อธิบายเพิ่มเติม

เริ่มแรกเราทำการสร้างตัวแปร app เป็นชนิด module ก่อนด้วยคำสั่ง var app = angular.module('myApp', []) จากนั้นก็เริ่มสร้าง controller ด้วยการเขียนว่า app.controller('MyController', function(\$scope)

จากนั้นในส่วนของคำสั่งภายใน controller เราจะมีการสร้างตัวแปรชื่อ value และกำหนดค่าเริ่มต้นเป็น 'Tavon' ด้วยการเขียนว่า \$scope.value = 'Tavon'

จากนั้นมาที่ HTML Code

เราทำการผูก tag body เข้ากับ module ของ angular ที่ชื่อว่า myApp ด้วยการเขียนว่า <body ng-app="myApp">

แล้วเราจะทำการสร้าง tag div และผูกเข้ากับ controller ชื่อ MyController

สุดท้ายก็แสดงค่าตัวแปรภายใน Controller ออกมานะ ด้วยการเขียนว่า {{ value }}

เปลี่ยนค่าตัวแปรใน Controller ผ่าน Function

ในการที่เรามีการใช้งานตัวแปรต่างๆ ในส่วนที่เป็น controller แล้วต้องการทำให้ค่าตัวแปรเปลี่ยนแปลงไปเรื่อยๆ จะทำได้อย่างไร โดยเฉพาะหากต้องทำการเรียกผ่านอีก function หนึ่ง (หรือ function ย่อยก็ได้) เราจะมีวิธีการอ้างอิงตัวแปร และสั่งเปลี่ยนค่านั้นได้อย่างไร ลองมาดูตัวอย่างโค้ดกันครับ

ตัวอย่างโค้ด

ch0-ex6-change-variable-by-function.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อว่า myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ MyController
      app.controller('MyController', function($scope) {
        // สร้างตัวแปรชื่อ value กำหนดค่าเป็น 1
        $scope.value = 1;

        // สร้าง function ชื่อ incrementValue ให้มี parameter ชื่อ increment
        $scope.incrementValue = function(increment) {

          // กำหนดตัวแปร value ให้บวกเพิ่มกับตัวแปร increment
          $scope.value += increment;
        }
      });
    </script>
```

```
</head>

<!-- ผูกเข้ากับ module myApp -->
<body ng-app="myApp">

<!-- ผูกเข้ากับ controller MyController -->
<div ng-controller="MyController">

<!-- เรียกใช้ function incrementValue และส่งค่า 5 เข้าไป -->
<p ng-init="incrementValue(5)">

    <!-- แสดงค่าตัวแปร value -->
    {{ value }}
</p>
</div>
</body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

จากตัวอย่างนี้เริ่มแรกเราทำการสร้างตัวแปรชื่อ app เป็น model ชื่อว่า myApp ก่อน จากนั้นเราจะทำการสร้าง controller ชื่อว่า MyController ซึ่งให้การทำงานภายใต้เป็นดังนี้

สร้างตัวแปรชื่อว่า value กำหนดค่าเป็น 1 ก่อน แล้วเราจะทำการสร้าง function ชื่อว่า incrementValue โดยใช้คำสั่ง \$scope.incrementValue = function โดยที่เราจะกำหนดว่า function นี้ให้มีการรับค่า parameter 1 ตัวคือ increment

ส่วนการทำงานภายใต้ function ชื่อ incrementValue ก็คือทำการเพิ่มค่าตัวแปร value ขึ้นไปอีก โดยให้เอกสารเดิม บวกเพิ่มกับค่าของ increment ด้วยคำสั่ง \$scope.value += increment

ในส่วนของ HTML Code เราทำการผูก body เข้ากับ model ชื่อว่า myApp

จากนั้นก็สร้าง tag div ผูกเข้ากับ controller ชื่อว่า MyController

แล้วตามด้วยสร้าง tag p กำหนดค่าล็อกให้ไปเรียกใช้งาน function incrementValue พร้อมส่งค่า 5 เข้าไปทาง parameter จากนั้นก็แสดงค่าตัวแปรของ value ออกมา ด้วยคำสั่ง {{ value }}

Encapsulation Model Value

ในการห่อหุ้มการทำงานบางอย่างของ function นั้น จะใช้วิธีให้ function ส่งค่ากลับคืนไปยังจุดที่เรียกใช้งาน หรือที่เราเรียกว่า return value นั่นเองครับ สำหรับ angular นี้ก็มีวิธีการให้เราทำได้อย่างง่ายดายมาก ผ่าน binding value ภายใต้ controller อีกด้วยหนึ่ง เดียวลองมาดูตัวอย่างโค้ด เพื่อความเข้าใจครับ

ตัวอย่างโค้ด

ch0-ex7-encapsulation-model-value.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปรชื่อ app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ MyController
      app.controller('MyController', function($scope) {

        // สร้างตัวแปรชื่อ value กำหนดค่าเป็น 1
        $scope.value = 1;

        // สร้าง function ชื่อ getIncrementValue
        $scope.getIncrementValue = function() {

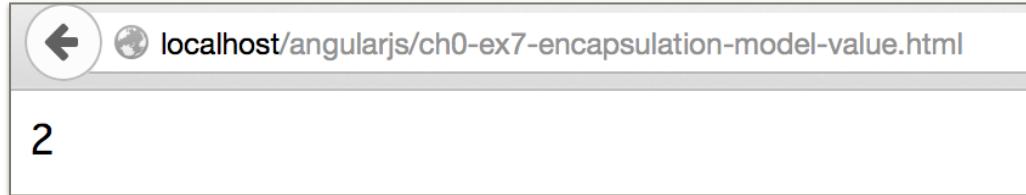
          // ส่งคืนผลการคำนวน ไปยังจุดเรียกใช้
          return $scope.value + 1;
        }
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- ผูกเข้ากับ controller MyController -->
    <div ng-controller="MyController">

      <!-- เรียกใช้ function getIncrementValue() และแสดงค่าที่ return ออกมานะ -->
      {{ getIncrementValue() }}
    </div>
  </body>
</html>
```

ผลการทำงาน



The screenshot shows a browser window with the URL localhost/angularjs/ch0-ex7-encapsulation-model-value.html. The page displays a single digit '2' in a large font.

อธิบายเพิ่มเติม

ในตัวอย่างนี้จะมีการเรียกใช้งาน function getIncrementValue() ในส่วนที่เป็น expression ซึ่งทุกๆ ครั้งที่เราเรียกใช้งาน function จะต้องมีเครื่องหมาย () ไว้ด้วยเสมอ และที่สำคัญ function นั้นจะต้องมีการ return value โดย กลับมาอยังจุดเรียกเรียกใช้ด้วยครับ

Responding to Scope Change

หากว่าผู้อ่านต้องการตรวจสอบการเปลี่ยนแปลงของค่าตัวแปร โดย แล้วทำการไปเรียกใช้งาน function เมื่อค่าตัวแปรเข้าเงื่อนไขต่างๆ สามารถเขียนได้โดยการใช้คำสั่ง \$watch มาช่วย ว่าค่านั้นเปลี่ยนแปลงไปจากเดิมแค่ไหน และค่อยทำการเขียนโปรแกรม ควบคุมการแสดงผล หรือคำนวนอีกที เพื่อความเข้าใจลงมาดูตัวอย่าง โค้ดต่อไปนี้ครับ

ตัวอย่าง โค้ด

ch0-ex8-responding-to-scope-change.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อว่า myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อว่า MyController
      app.controller('MyController', function($scope) {
        // สร้างตัวแปรชื่อ name กำหนดค่าเป็น ""
        $scope.name = "";

        // ตักค่าของตัวแปร name เมื่อมีการเปลี่ยนแปลง
        $scope.$watch('name', function(newValue, oldValue) {
          // เช็คค่าตัวแปร name ถ้ามีขนาดมากกว่า 5 ตัวให้คำสั่ง if ทำงาน
          if ($scope.name.length > 5) {

            // สร้างตัวแปร hello เป็นข้อความว่า Hello : ตามด้วยตัวแปร name
            $scope.hello = "Hello : " + $scope.name
          }
        });
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- ผูกเข้ากับ controller MyController -->
    <div ng-controller="MyController">

      <!-- สร้างช่องรับข้อมูล ผูกเข้าตัวแปร name -->
      <input type="text" ng-model="name" />
```

```
<!-- แสดงค่าตัวแปร hello -->
<p>{{ hello }}</p>
</div>
</body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

ในตัวอย่างนี้ เราเริ่มต้นด้วยการสร้างตัวแปร app เป็น module ชื่อว่า `myApp` จากนั้นก็ทำการสร้าง controller ชื่อ `MyController` และภายใน controller ก็เริ่มสร้างตัวแปร `name` กำหนดค่าเป็น "" (ข้อความว่างเปล่า)

ลำดับต่อมาเราทำการตักค่าของตัวแปร `name` เมื่อมีการเปลี่ยนแปลงไป โดยใช้ `$watch('name', function(newValue, oldValue)` ซึ่ง parameter 2 ตัวนี้ จะต้องส่งเข้าไปตามลำดับ และให้มีชื่อตามตัวอย่างนี้ เลยก็จะ ส่วนการทำงานภายใน `$watch` นี้ก็ไม่มีอะไรมาก เพียงแค่ทำการตักค่าตัวแปรไว้ว่า มีขนาดเกิน 5 ตัว แล้วหรือยัง ถ้าหากเกินก็ทำการสร้างตัวแปร `hello` เพิ่มมาอีก โดยกำหนดค่าของตัวแปร `hello` เป็นข้อความว่า Hello : ตามด้วยตัวแปร `name` ครับ

ส่วนของโค้ด HTML ก็ทำการผูก tag `body` เข้ากับ module `myApp` ก่อน จากนั้นผูก tag `div` เข้ากับ controller ชื่อ `MyController` และตามด้วย สร้างช่องรับข้อมูลให้ผูกค่าไว้กับตัวแปร `name`

สุดท้ายก็แสดงค่าตัวแปร `hello` ออกมา ด้วยคำสั่ง `{{ hello }}` ครับ (ซึ่งค่านี้จะได้ก็ต่อเมื่อ ตัวแปร `name` ถูกป้อนไปแล้วกว่า 5 ตัวอักษรเท่านั้น)

Share ตัวแปร ระหว่าง Controller

หากผู้อ่านต้องการใช้งานตัวแปร ระหว่างหลายๆ controller จะทำได้อย่างไร เพราะบางครั้งตัวแปรเดียวกัน ต้องถูกเรียกใช้ในหลายจุด ดังนั้นจึงขออธิบายในหัวข้อนี้ครับ ด้วยโค้ดง่ายๆ ดังนี้

ตัวอย่างโค้ด

ch0-ex9-share-model-between-controller.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);
```

```
// สร้าง controller ชื่อ MyController
app.controller('MyController', function($scope) {

    // สร้างตัวแปร name กำหนดค่าเป็น 'Tavon'
    $scope.name = 'Tavon';

    // สร้างตัวแปร user กำหนดเป็น object มีสมาชิกคือ name เป็น Kob
    $scope.user = {
        name: 'Kob'
    };
});

// สร้าง controller ชื่อ MyNestedController
app.controller('MyNestedController', function($scope) {

});
```

</script>

</head>

```
<!-- ผูกเข้ากับ module myApp -->
<body ng-app="myApp">

    <!-- ผูกเข้ากับ controller MyController -->
    <div ng-controller="MyController">
        <label>Primitive</label>

        <!-- ผูกเข้ากับตัวแปร name -->
        <input type="text" ng-model="name" />

        <label>Object</label>

        <!-- ผูกเข้ากับตัวแปร user.name -->
        <input type="text" ng-model="user.name" />

        <!-- ผูกเข้ากับ controller MyNestedController -->
        <div class="nested" ng-controller="MyNestedController">
            <label>Primitive</label>

            <!-- ผูกเข้ากับตัวแปร name -->
            <input type="text" ng-model="name" />

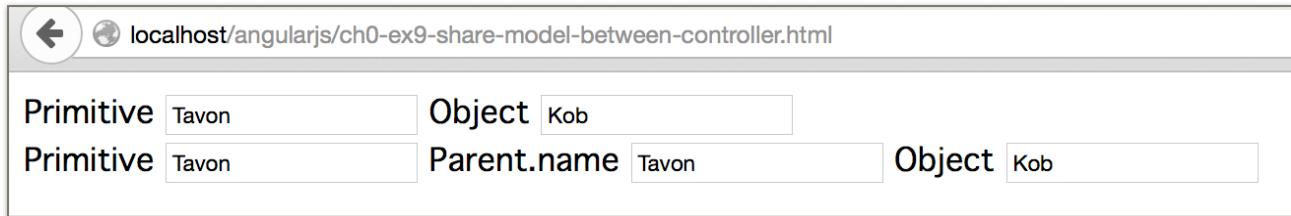
            <label>Parent.name</label>

            <!-- ผูกเข้ากับตัวแปร name ของ controller MyController -->
            <input type="text" ng-model="$parent.name" />

            <label>Object</label>

            <!-- ผูกเข้ากับตัวแปร user.name -->
            <input type="text" ng-model="user.name" />
        </div>
    </div>
</body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

การสร้าง controller นั้นสามารถที่จะผูกในส่วนของ view ให้ช้อนๆ กันหลายชั้นได้ เรียกว่า Nested Controller ครับ และการอ้างอิงตัวแปรของ controller ที่เป็นแม่ เราจะใช้ \$parent.ชื่อตัวแปร เพื่อไปเรียกใช้งานอีกทีหนึ่ง

Share Code ระหว่าง Controller ด้วย Service

ถ้าหากเราต้องการ ให้มีการคำนวนอะไรสักอย่างไว้ใน Service กลาง และเอาค่าตัวแปรนั้นๆ ที่ผ่านการคำนวนเรียบร้อย ไปใช้งานต่อที่ controller อื่น ไม่ว่าจะ 1 controller หรือหลายๆ controller จะทำได้อย่างไร ลองมาดูตัวอย่างการเขียน โค้ดกันครับ โดยผมจะพาใช้งาน Service แบบง่ายๆ ก่อน

ตัวอย่าง โค้ด

ch0-ex10-service.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อว่า myApp
      var app = angular.module('myApp', []);

      // สร้าง service ชื่อ UserService
      app.factory('UserService', function() {

        // สร้างตัวแปร users แบบ array และกำหนดค่าลงไว
        var users = ["Tavon", "Hope", "Porn"];

        // กำหนดการ return แบบต่างๆ
        return {

          // ส่งคืนตัวแปร users ทั้งหมด
          all: function() {
            return users;
          },

          // ส่งคืนเฉพาะตัวแรกเท่านั้น
          first: function() {
            return users[0];
          }
        }
      })
    </script>
  </head>
  <body>
    <div ng-app="myApp">
      <div>Primitive <input type="text" value="Tavon"/> Object <input type="text" value="Kob"/>
      <br/>
      <div>Primitive <input type="text" value="Tavon"/> Parent.name <input type="text" value="Tavon"/> Object <input type="text" value="Kob"/>
    </div>
  </body>
</html>
```

```
        }
    });

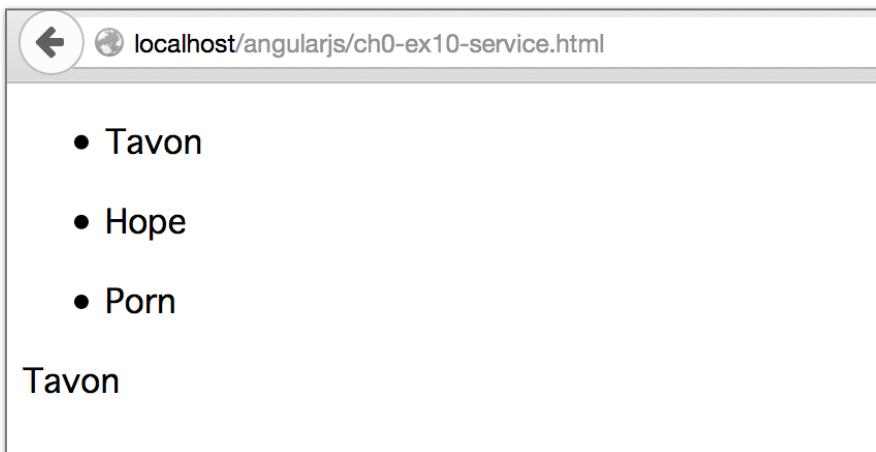
// สร้าง controller ชื่อ MyController ให้เชื่อมกับ UserService
app.controller('MyController', function($scope, UserService) {
    $scope.users = UserService.all();
});

// สร้าง controller ชื่อ AnotherController ให้เชื่อมกับ UserService
app.controller('AnotherController', function($scope, UserService) {
    $scope.firstUser = UserService.first();
});
</script>
</head>

<!-- ผูกเข้ากับ module MyApp --&gt;
&lt;body ng-app="myApp"&gt;
    <!-- ผูกเข้ากับ controller MyController --&gt;
    &lt;div ng-controller="MyController"&gt;
        <!-- ทำการวนรอบ อ่าน array ทั้งหมด เก็บลง user --&gt;
        &lt;ul ng-repeat="user in users"&gt;
            <!-- แสดงตัวแปร user --&gt;
            &lt;li&gt;{{ user }}&lt;/li&gt;
        &lt;/ul&gt;
    &lt;/div&gt;

    <!-- ผูกเข้ากับ controller AnotherController --&gt;
    &lt;div ng-controller="AnotherController"&gt;
        <!-- แสดงตัวแปร firstUser --&gt;
        {{ firstUser }}
    &lt;/div&gt;
&lt;/body&gt;
&lt;/html&gt;</pre>
```

ผลการทำงาน



อธิบายเพิ่มเติม

ผมขออธิบายเพิ่มเฉพาะส่วนที่มีคำสั่งใหม่ๆ เพิ่มมานะครับ ในการสร้าง Service นั้นเราจะใช้คำว่า factory ครับ และกำหนดชื่อของ Service ลงไป เช่น app.factory("UserService", function() {})

ส่วนการ return เราสามารถที่จะ return เป็น object ได้ และใช้ key มาเป็นตัวกำหนดการอ้างอิง เช่น

```
all : function() { }
first : function() { }
```

การทำให้ controller เชื่อมโยงเข้ากับ Service จะใช้วิธีการสร้าง ให้มี parameter แบบ 2 ตัว เช่น

```
app.controller("MyController", function($scope, UserService) { })
```

การแสดงผลแบบวนรอบ (loop) จะใช้ ng-repeat ซึ่งใช้กับ html tag อะไรก็ได้ และทำการอ่านค่ามาลงตัวแปร
ไว้เช่น ng-repeat="user in users"

Directive คืออะไร ?

directive นั้นเป็นจุดเด่นมากๆ ของ angular เพราะทำให้เราสามารถสร้าง element ของ html tag ได้
ตามความต้องการ ซึ่งตรงนี้เองจะทำให้เราสามารถสร้าง component ที่นำเอาไปใช้ซ้ำๆ อีกรอบได้ (reuse)
โดยจะช่วยความซับซ้อนไว้เบื้องหลัง dom อีกทีหนึ่ง นอกจากนี้แล้วก็ยังทำให้มี css, event อยู่ภายใน
directive นั้นๆ ได้ด้วย

Enabling/Disabling DOM Elements

ตัวอย่างนี้ผมจะพำนักระการลั่ง disable button โดยให้เชื่อมกับ checkbox ซึ่งอาศัยการอ่านค่า state
มาทำการเปลี่ยนสถานะ disable button อีกทีหนึ่ง

ตัวอย่างโค้ด

ch0-ex11-use-directive.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
  </head>
  <body ng-app>
    <label>

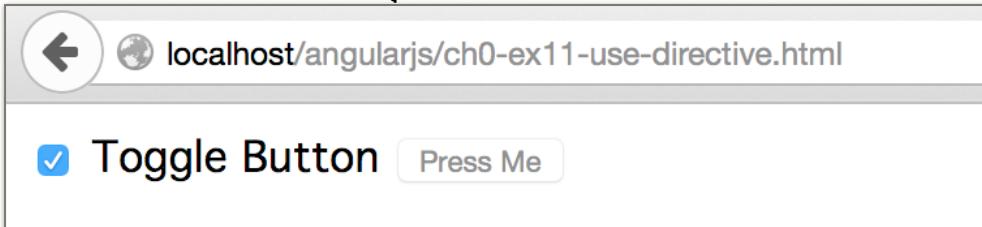
      <!-- เชื่อมเข้ากับ checked -->
      <input type="checkbox" ng-model="checked" />
      Toggle Button
    </label>

      <!-- เชื่อมเข้ากับ checked -->
      <button ng-disabled="checked">
        Press Me
      </button>
    </body>
</html>
```

ผลการทำงาน



เมื่อเราทำการคลิกที่ checkbox ปุ่ม Button จะใช้งานไม่ได้ดังภาพ



อธิบายเพิ่มเติม

ในตัวอย่างนี้เราได้ทำการผูกตัวแปร checked ไว้กับ button และ checkbox โดยการควบคุมสถานะใช้งานของปุ่ม ก็คือ จะใช้ ng-disabled เข้ามาช่วยครับ

เปลี่ยน DOM ใน Response to User Actions

ถ้าคณจะเปลี่ยน css ของ HTML เมื่อทำการเอาเม้าส์ไปคลิกบน component ต่างๆ โดยจะซ่อนความชั้นช้อนไว้เนื่องหลัง เพื่อที่จะนำเอา component นั้นๆ กลับมาใช้งานอีก จะทำได้อย่างไร วิธีการก็คือใช้ tag my-widget ครับ ลองดูโค้ดตัวอย่างกัน

ตัวอย่างโค้ด

ch0-ex12-use-widget.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง directive ชื่อ myWidget
      app.directive('myWidget', function() {

        // สร้างตัวแปรเชื่อมโยงชื่อ linkFunction
        var linkFunction = function(scope, element, attributes) {

          // อ่านค่า element ตัวแรก เก็บลงตัวแปร paragraph
          var paragraph = element.children()[0];

          // กำหนด event onclick
          paragraph.onclick = function() {
```

```
// เปลี่ยนสีพื้นหลังเป็นสีแดง
paragraph.style.backgroundColor = 'red';
}

}

// คืนค่าไปยังจุดเรียกใช้
return {

    // ระบุว่าเป็นแบบ elements
restrict: 'E',

    // เชื่อมกับตัวแปร linkFunction
link: linkFunction
}
});
</script>
</head>

<!-- ผูกเข้า model myApp -->
<body ng-app="myApp">

<!-- เชื่อมเข้ากับ myWidget -->
<my-widget>
    <p>Hello</p>
</my-widget>
</body>
</html>
```

ผลการทำงาน



เมื่อเราคลิกที่ paragraph Hello พื้นหลังจะเปลี่ยนไปดังนี้



อธิบายเพิ่มเติม

ในการสร้าง widget นั้นจะใช้คำว่า directive จากนั้นเราจะสร้างตัวเชื่อมโยง โดยในตัวอย่างได้ทำการสร้างตัวแปรชื่อ linkFunction = function() และตามด้วยเก็บค่า element ลงใน ตัวแปร paragraph และเรียกกำหนด event onclick ให้กับมัน พร้อมสั่งเปลี่ยนค่า css เพื่อให้พื้นหลังเป็นสีแดง เมื่อถูกกด

Render HTML in Directive

ในการที่เราต้องการสั่งให้ directive ทำการ render รหัส HTML นั้น สามารถทำได้โดยการกำหนดเป็นแบบ template ซึ่งจะทำให้เราแทรกคำสั่ง tag html ต่างๆ ลงไปได้ตามต้องการ ลองมาดูตัวอย่างกันครับ

ตัวอย่าง โค้ด

ch0-ex13-render-html-in-directive.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง widget ชื่อ myWidget
      app.directive('myWidget', function() {
        return {

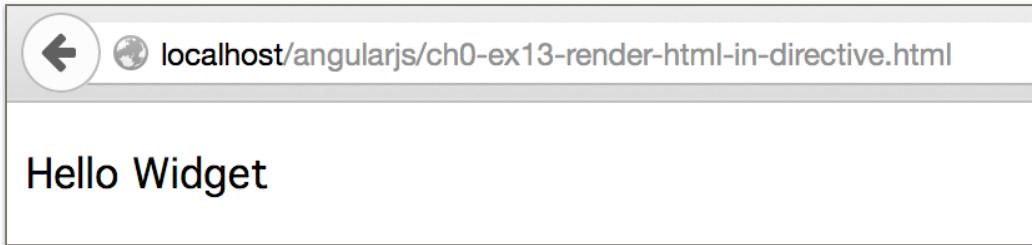
          // กำหนดให้เป็นแบบ element
          restrict: 'E',

          // กำหนดการแสดงผล เป็น HTML Code
          template: '<p>Hello Widget</p>'
        }
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- เรียกแสดงผล widget ของเราระหว่าง myWidget -->
    <my-widget />
  </body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

ตัวอย่างนี้เราสร้าง directive แบบง่ายๆ ครับแล้วก็ return template ซะเลย โดยใช้เป็น html code เพื่อให้ไปแสดงผลบนส่วนที่เป็น widget จากนั้นบริเวณ tag body เรายังเชื่อมเข้ากับ module myApp และสร้าง tag ชื่อว่า <my-widget /> มันก็จะไปเชื่อมโยงกับ directive ที่ชื่อว่า myWidget เองครับ

Render Directive DOM Node Children

หากเราต้องการเรียกใช้งาน directive ภายในส่วนที่เป็น widget จะต้องใช้คำสั่ง transclude กำหนด เป็นค่า true และใช้คำสั่ง ng-transclude เข้ามาช่วยเพื่อเรียกใช้งานส่วนที่เป็น DOM แบบ Node ย่อย (เรียกว่า Children Node) มาลงดูตัวอย่าง โค้ดกันครับ

ตัวอย่างโค้ด

ch0-ex14-render-directive-dom.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง directive ชื่อ myWidget
      app.directive('myWidget', function() {
        // ส่งคืนไปยังจุดเรียกใช้
        return {

          // กำหนดให้เป็น element
          restrict: 'E',

          // กำหนดให้เป็น transclude
          transclude: true,

          // กำหนดรูปแบบ HTML Code
          template: "<h3>Heading</h3><div ng-transclude></div>"
        };
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">
```

```
<!-- ผูกเข้ากับ myWidget -->
<my-widget>
  <p>Paragraph A</p>
</my-widget>
</body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

<div ng-transclude></div> คือการสั่งให้ tag ย่อๆ ภายใน <my-widget> ไปแสดงผล ณ ตำแหน่งนั้น จากการทำงาน จะเห็นว่า Heading มา ก่อน คำว่า Paragraph A นั้นเป็น เพราะเราวาง <div ng-transclude></div> ไว้หลังจาก <h3>...</h3> นั้นเอง

Repeat Rendering DOM Node

หากเราต้องการทำให้ output ของ html มีการทำซ้ำเพื่อแสดงข้อมูลใดๆ ออกมาก เช่นการล็อกวน loop เพื่อแสดง tag ต่างๆ เรียงออกมา จะเขียนได้อย่าง เมื่อต้องสร้างให้เป็นแบบ directive ลองมาดูตัวอย่างโค้ด กันครับ

ตัวอย่างโค้ด

ch0-ex15-repeat-rendering-dom-node.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module("myApp", []);

      // สร้าง directive ชื่อ myRepeat
      app.directive("myRepeat", function() {
        // ส่งคืนไปยังจุดเรียกใช้
        return {

          // กำหนดว่าเป็นแบบ elements
          restrict: "E",
```

```
// สั่ง compile เพื่อส่งคืนไปจุดเรียกใช้
compile: function(element, attributes) {

    // สร้างตัวแปร content โดยอ่านเอา ข้อมูลใน element
    var content = element.children();

    // วนรอบ ตามจำนวนตัวแปร repeat
    for (var i = 1; i < attributes.repeat; i++) {

        // สั่งให้ต่อข้อความเข้าไปใน element โดยทำการ clone ข้อมูลเดิมมาต่อท้าย
        element.append(content.clone());
    }
};

//);
</script>
</head>

<!-- ผูกเข้ากับ module myApp -->
<body ng-app="myApp">

<!-- เรียกใช้ directive myRepeat และกำหนดค่าตัวแปร repeat = 5 -->
<my-repeat repeat="5">
    <h1>Heading</h1>
    <p>Paragraph</p>
</my-repeat>
</body>

</html>
```

ผลการทำงาน

The screenshot shows a web page with the URL `localhost/angularjs/ch0-ex15-repeat-rendering-dom-node.html`. The page contains five identical sections, each consisting of a heading and a paragraph. The headings are bolded and repeated five times. The paragraphs are standard text.

```
<h1>Heading</h1>
<p>Paragraph</p>
<h1>Heading</h1>
<p>Paragraph</p>
<h1>Heading</h1>
<p>Paragraph</p>
<h1>Heading</h1>
<p>Paragraph</p>
<h1>Heading</h1>
<p>Paragraph</p>
```

อธิบายเพิ่มเติม

เริ่มต้นเราทำการเขียนโค้ดสร้างตัวแปร app ขึ้นมา เป็น module ชื่อว่า `myApp` จากนั้นทำการสร้าง directive ชื่อว่า `myRepeat` และ return ค่าโดยกำหนดให้เป็นดังนี้

`restrict: 'E'` คือบอกว่า นี่จะเป็นการ return แบบ element

กำหนด `compile` เป็น function ที่มีการทำงานดังนี้

สร้างตัวแปรชื่อ `content` โดยไปอ่านค่า tag ย่อຍภัยໄດ້ `<my-repeat>` ในที่นี่เลยใช้คำสั่งว่า `element.children()`

สุดท้ายทำการวน loop ตามจำนวน repeat เพื่อให้ทำการ append (ต่อข้อความเข้าไปอีก) การแสดงผล โดยให้ `clone` (คัดลอก) ของเดิมมาทั้งหมด

Chapter 1

Advanced AngularJS + AJAX

Currency Filter

ในกรณีที่เราต้องการจะทำการแปลงค่า output ได้ๆ นั้นสามารถทำได้ง่ายมาก โดยการอ้างอิงไฟล์ i18n ก่อน จากนั้นตามด้วยสิ่งที่แปลงผ่านคำสั่ง currency และยังสามารถกำหนดการแสดงผลข้อความได้อีกด้วย ลองมาดูตัวอย่างโค้ดกันครับ โดยผมจะพาแปลงค่าเงิน แบบง่ายๆ เป็นหน่วย Euro

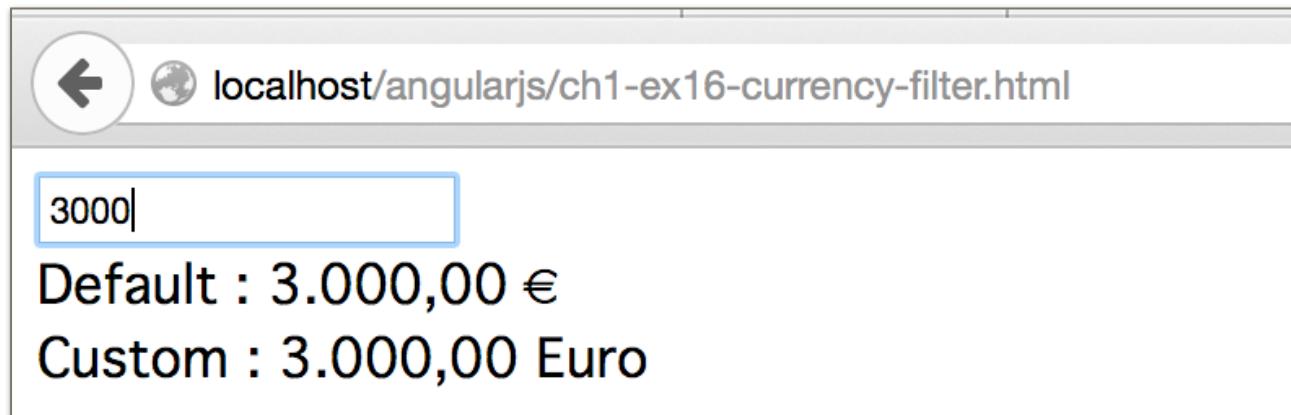
ตัวอย่างโค้ด

ch1-ex16-currency-filter.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script src="angular-1.4.3/i18n/angular-locale_de.js"></script>
  </head>

  <body ng-app>
    <input type="text" ng-model="amount" />
    <div>
      Default : {{ amount | currency}}
    </div>
    <div>
      Custom : {{ amount | currency: "Euro" }}
    </div>
  </body>
</html>
```

ผลการทำงาน



อธิบายเพิ่มเติม

จากตัวอย่างโค้ดจะเห็นว่าเราใช้เครื่องหมาย | เพื่อมาสั่งให้เรียกทำงานเพิ่มเติมอีก โดยในที่นี้เราสั่งเรียกคำสั่ง currency เพื่อเป็นการแปลงตัวเลขนั้นๆ ให้กลายเป็นหน่วยเงินตรา และก็มีการกำหนดเป็นหน่วย euro (ที่ขาดไม่ได้เลยคือการ reference js ไฟล์ `angular-locale_de.js` ที่อยู่ใน i18n ด้วยครับ)

Revert Input String

ตัวอย่างนี้ผมจะพาเขียน filter ขึ้นมาเอง ในกรณีที่เราจะทำ custom filter แล้วให้ตัวแปรของเรามาเป็นตัวอย่างง่ายๆ ครับ แค่ทำการกรอกข้อมูล และกลับด้านตัวอักษร จากหลังไปหน้า เท่านั้นเอง ลองมาดูตัวอย่าง โค้ดกัน

ตัวอย่าง โค้ด

ch1-ex17-revert-input-string.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้างตัวแปร app เป็น module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง filter ชื่อ reverse
      app.filter('reverse', function() {

        // ส่งคืนผลการทำงาน
        return function(input) {

          // สร้างตัวแปร result เป็นค่าว่างไว้ก่อน
          var result = '';

          // กำหนดให้ตัว input เชื่อมกับช่องว่าง
          input = input || '';

          // วนรอบ เท่ากับจำนวนอักษรใน input
          for (var i = 0; i < input.length; i++) {

            // กลับด้านตัวอักษร โดยเชื่อมเข้าไปเก็บไว้ในตัว result
            result = input.charAt(i) + result;
          }

          // ส่งคืนตัวแปร result
          return result;
        }
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- สร้างช่องรับข้อมูล เชื่อมเข้ากับ ตัวแปร text -->
    <input type="text" ng-model="text" />

    <!-- แสดงตัวแปร text -->
    <div>Input : {{ text }}</div>
```

```
<!-- สั่งให้ไปเรียกใช้ filter ชื่อ reverse -->
<div>Filtered : {{ text | reverse }}</div>
</body>
</html>
```

ผลการทำงาน

The screenshot shows a browser window with the URL `localhost/angularjs/ch1-ex17-revert-input-string.html`. Inside the browser, the input field contains the text "Tavon Seesenpila". Below it, the output is displayed in two lines: "Input : Tavon Seesenpila" and "Filtered : alipnesees novaT". The "Filtered" text is displayed in a larger, bold font.

อธิบายเพิ่มเติม

ในตัวอย่างนี้เราทำการสร้าง filter ชื่อว่า `reverse` ขึ้นมา เพื่อแปลงค่า `input` ให้สับซ้อนขึ้นจากตัวแรกไปอยู่ตัวสุดท้าย ไล่ไปทีละตัว จากนั้นส่วนของ `html code` เราทำการผูกเข้ากับ `ng-model` โดยกำหนดเป็น `myApp` และมีการเรียกใช้งาน filter ที่ชื่อ `reverse` ด้วยคำสั่ง `{{ text | reverse }}` เพียงเท่านี้โค้ดก็ทำงานได้แบบง่ายๆ แล้วครับ

Filter a List of DOM Node

เป็นการกรองหาข้อมูลใน DOM Node โดยค้นหาได้ตามต้องการ ส่วนมากจะใช้เพื่อการ loop แสดงผล แล้วจัดรูปแบบการแสดงตามต้องการ ในตัวอย่างนี้ผมจะพามาทำการค้นหาคำ ผ่านทาง parameter ลองมาดูตัวอย่างโค้ดข้างล่างนี้กันครับ

ตัวอย่างโค้ด

ch1-ex18-filter-dom-node.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง filter ชื่อ exclude
      app.filter('exclude', function() {
        // ส่งคืน function มีการรับ parameter 2 ตัว
        return function(input, exclude) {
```

```
// สร้างตัวแปร result เป็น array ว่าง
var result = [];

// ทำการวนรอบ ตามจำนวนอักษรที่ input เข้ามา
for (var i = 0; i < input.length; i++) {

    // เริ่มเช็ค input ใน array ทีละช่อง
    if (input[i] !== exclude) {

        // ถ้าหากไม่ตรงกับคำที่เราคัน ก็ให้เก็บค่าลง array ไว้
        result.push(input[i]);
    }
}

// ส่งคืนตัวแปร array ข้อ result
return result;
}
});
</script>
</head>

<!-- ผูกเข้ากับ myApp -->
<body ng-app="myApp">

<!-- สร้างตัวแปร names มีค่า 3 ตัว ดังนี้ -->
<ul ng-init="names=[ 'Tavon' , 'Java' , 'Ruby' ]">

    <!-- วนรอบ เพื่อแสดงผล โดยส่ง exclude เป็น Java เข้าไป -->
    <li ng-repeat="name in names | exclude: 'Java'">

        <!-- แสดงค่าตัวแปร -->
        {{ name }}
    </li>
</ul>
</body>
</html>
```

ผลการทำงาน

The screenshot shows a web browser window with the following details:

- Address Bar:** localhost/angularjs/ch1-ex18-filter-dom-node.html
- Content Area:** A list of names:
 - Tavon
 - Ruby

อธิบายเพิ่มเติม

เริ่มแรกเราทำการสร้าง module ชื่อว่า `myApp` จากนั้นก็ทำการสร้าง filter ชื่อว่า `exclude` โดยให้ทำการ return function ที่มีการรับ parameter 2 ตัวคือ `input` กับ `exclude` ซึ่งการทำงานภายใน function เป็นดังนี้

สร้างตัวแปร `result` เป็นแบบ array ว่างไว้ก่อน จากนั้นก็วนรอบด้วย `for` เท่ากับจำนวน `input` ที่เข้ามา (อ่านจาก array ที่ชื่อ `input`) ภายใน `loop` แต่ละรอบ เราทำการเช็คด้วย `if` ว่าค่าของช่องนั้นๆ ตรงกับตัวแปร `exclude` หรือไม่ หากว่าไม่ตรงกัน ก็ทำการเก็บค่าลงใน array `result` ไปเรื่อยๆ

และสุดท้ายเมื่อ `loop` ทำงานจบ ก็ทำการส่งคืนไปยังจุดเรียกใช้

ต่อมา ส่วนของ `html code` ทำการผูกเข้ากับ `model` ชื่อ `myApp` และมีการสร้าง `ui` พร้อมกับกำหนดตัวแปร `names` ให้มีค่า sama ชิกดังนี้ `['Tavon', 'Java', 'Ruby']` ตามด้วยจุดที่สำคัญมากของโค้ดนี้ ก็คือ การสั่ง `repeat` เพื่อทำซ้ำ โดยอ่านค่าจากตัวแปร `names` และกำหนด `exclude` เป็น `'Java'`

คำว่า `exclude` ในนี้ก็คือชื่อของ filter นั่นเองครับ

Request JSON with AJAX

ตัวอย่างนี้ผมจะพากำการเขียนโปรแกรมเพื่อเชื่อมต่อกับ External Data โดยจะใช้วิธีง่ายๆ ก่อนคือ การอ่านค่า JSON Data ผ่าน AJAX Programming ซึ่ง AngularJS มีคำสั่งให้เราใช้งานได้อย่างง่ายดาย ผ่านตัวแปร `$http` ครับ โดยดึงได้ทั้งแบบ GET, POST

โค้ดตัวอย่างนี้ผมจะแบ่งออกเป็น 2 ฝั่ง คือด้าน Client และด้าน Server โดยที่ด้านของ Server จะเลือกใช้ภาษา PHP แบบธรรมด้า เพราะหลายคนคุ้นเคยดีอยู่แล้ว ลองมาดูตัวอย่างการเขียนโค้ดกันครับ

ตัวอย่างโค้ด

ch1-ex19-json-with-ajax.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ PostController
      app.controller('PostController', function($scope, $http) {
        // เรียกคำสั่งไปที่ไฟล์ ch1-ex19-json-with-ajax.php
        $http.get('ch1-ex19-json-with-ajax.php').

        // ถ้าตั้งค่าให้สำเร็จ
        success(function(data, status, headers, config) {

          // ให้เก็บค่าจาก server ลงตัวแปร $scope.posts
          $scope.posts = data;
        }).

        // ถ้าหากมีข้อผิดพลาด
        error(function(data, status, headers, config) {
```

```
// ให้แสดงค่าของทาง console
console.log(data);
});
});
</script>
</head>

<!-- ผูกเข้า module myApp -->
<body ng-app="myApp">

<!-- ผูกเข้ากับ controller PostController -->
<div ng-controller="PostController">

<!-- ทำการแสดงค่าจาก posts -->
<ul ng-repeat="post in posts">
<li>

    <!-- 显示เฉพาะส่วนของ title -->
    {{ post.title }}
</li>
</ul>
</div>
</body>
</html>
```

โค้ดฝั่ง server

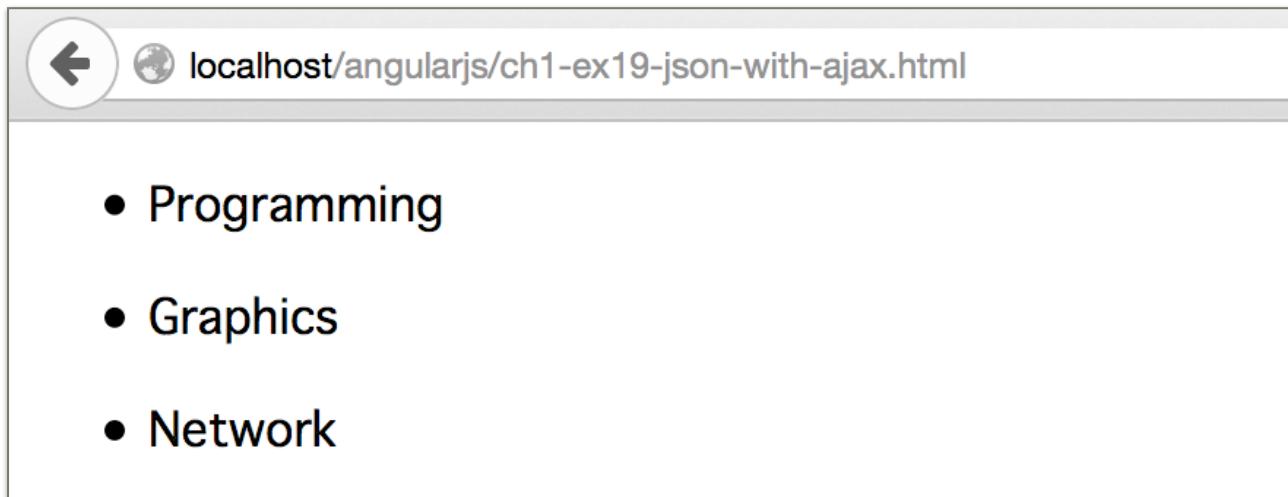
ch1-ex19-json-with-ajax.php

```
<?php

// สร้าง array ชื่อ data
$data = array(
    // กำหนดสมาชิกลงไป 3 ตัว แต่ละตัวมี key ชื่อ title
    array('title' => 'Programming'),
    array('title' => 'Graphics'),
    array('title' => 'Network')
);

// ส่งให้แสดงผลเป็นแบบ json data
echo json_encode($data);
?>
```

ผลการทำงาน



The screenshot shows a web browser window with the URL `localhost/angularjs/ch1-ex19-json-with-ajax.html`. The page displays a list of three items:

- Programming
- Graphics
- Network

อธิบายเพิ่มเติม

ตัวอย่างนี้ถือได้ว่า ใช้โค้ดที่ซับซ้อนไปสักหน่อย ดังนั้นผมขออธิบายอย่างละเอียดดังนี้ เริ่มต้นเราทำการสร้าง module ชื่อว่า `myModule` จากนั้นก็ทำการสร้าง controller ชื่อว่า `PostController` โดยมีการรับ parameter 2 ตัวไว้ใช้งาน ได้แก่ `$scope` และ `$http`

จากนั้นก็สั่งไปเรียกไฟล์ด้าน server ให้ทำการ โดยตัวอย่างนี้ไปเรียกไฟล์ `ch1-ex19-json-with-ajax.php` ตามด้วยเขียนคำสั่งให้ทำงานกรณีที่ `success` เราจะให้เก็บค่าที่ server ส่งกลับมา (ตัวแปร `data`) เอาไปไว้ใน `$scope.posts`

ตามด้วยการกำหนดโค้ด กรณีที่เกิดข้อผิดพลาด ด้วย `method` ชื่อ `error` ให้ทำการแสดงค่าผลการทำงานออกทาง `console` ด้วยคำว่า `console.log(data)`

ในส่วนของ `html` code เราทำการผูก `body` เข้ากับ `module` ชื่อ `myApp` ก่อน ตามด้วยการผูก `div` เข้ากับ `controller` ชื่อว่า `PostController` สุดท้ายก็ทำการ `loop` เพื่ออ่านค่าตัวแปร `posts` มาลงไว้ใน `post` แล้ว แสดงออกมวดด้วยคำสั่ง `{{ post.title }}` โดยคำว่า `.title` นี้หมายถึง `key` ของ `json` data ที่ส่งมาจาก server

อธิบายโค้ดฝั่ง server

ทำการสร้าง array ขึ้นมา ชื่อว่า `$data` และกำหนดให้มีสมาชิกข้างในทั้งหมด 3 ตัว โดยที่แต่ละตัวมีค่า `key` ชื่อว่า `title` และมีข้อมูลต่างกันไป จากนั้นก็สั่งให้แสดงผลออกมาในแบบของ JSON ด้วยคำสั่ง `echo json_encode($data)`

Find by ID

เป็นการส่งค่าไปทาง server และมีการส่งตัวแปรไปให้ด้วย เพื่อ ใช้งานต่างๆ โดยมากนัก ใช้เพื่อลบ แก้ไข คืนหา หรือทำการคำนวนบางอย่างครับ ตัวอย่างนี้ผมจะพาส่งค่าผ่านแบบ GET ด้วยตัวแปร `$http` กัน

ตัวอย่างโค้ด

ch1-ex20-find-by-id.html

```
<html>
<head>
<script src="angular-1.4.3/angular.js"></script>
```

```
<script>
  // สร้าง module ชื่อ myApp
  var app = angular.module('myApp', []);

  // สร้าง controller ชื่อ PostController
  app.controller('PostController', function($scope, $http) {

    // เรียกคำสั่งไปที่ไฟล์ฟัง server พร้อมส่ง id=10 ไปให้
    $http.get('ch1-ex20-find-by-id.php?id=10').

    // ถ้าดึงค่าได้สำเร็จ
    success(function(data, status, headers, config) {

      // ให้เก็บค่าจาก server ลงตัวแปร $scope.output
      $scope.output = data;
    }).

    // ถ้าหากมีข้อผิดพลาด
    error(function(data, status, headers, config) {

      // ให้แสดงค่าของทาง console
      console.log(data);
    });
  });
</script>
</head>

<!-- ผูกเข้า module myApp -->
<body ng-app="myApp">

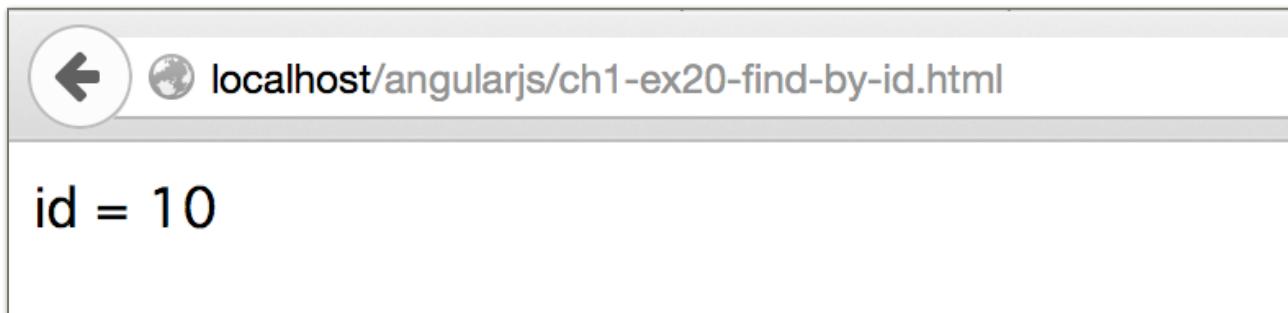
  <!-- ผูกเข้ากับ controller PostController -->
  <div ng-controller="PostController">

    <!-- แสดงค่าตัวแปร output -->
    {{ output }}
  </div>
</body>
</html>
```

โค้ดฟัง server

```
<?php
echo $_GET['id'];
?>
```

ผลการทำงาน



อธิบายเพิ่มเติม

ในตัวอย่างนี้แตกต่างจากตัวอย่างก่อนหน้าเพียงเล็กน้อย นั่นคือมีการส่งตัวแปรไปให้ผ่านทาง url สังเกตุได้ว่าตอนที่เราส่งเรียกการทำงานไปที่ server จะมีการเติม ?id=10 ไปด้วย นั่นหมายความว่า เราส่งตัวแปรชื่อ id มีค่าเป็น 10 ไปให้ server ใช้ทำงาน

ส่วนโค้ดฝั่งของ server นั้นก็จะมีการรับมาด้วย \$_GET['id'] และแสดงผลโดยใช้คำสั่ง echo ผลการทำงานก็จะเป็น id = 10 ตามภาพที่เห็นครับ

Call Service by Event

บางกรณีที่เราต้องการสั่งเรียกการทำงานร่วมกับ code ฝั่ง server side นั้น เชียนได้ง่ายๆ ผ่านทาง event ของ ng-click และไปเรียกผ่าน controller อีกทีหนึ่ง ซึ่งตัวอย่างนี้ผมจะสาธิตทั้งการส่งค่าแบบ GET, POST เพื่อให้ครบถ้วน ในตัวอย่างเดียว

ลองมาดูตัวอย่าง โค้ดนีกันครับ ว่าจะเป็นอย่างไร

ตัวอย่าง โค้ด

ch1-ex21-call-server-by-event.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ MyController
      app.controller('MyController', function($scope, $http) {
        // สร้าง function ชื่อ callService1
        $scope.callService1 = function() {
          // เรียกใช้ตัวแปร $http
          $http({
            // ส่งค่าไปยัง server
            url: 'ch1-ex21-call-server-by-event1.php',
            // กำหนดเป็นแบบ GET
            method: 'GET'
          }).then(function(response) {
            $scope.message = response.data;
          });
        };
      });
    </script>
  </head>
  <body>
    <div ng-app="myApp" ng-controller="MyController">
      <button ng-click="callService1()">Call Server</button>
      {{message}}
    </div>
  </body>
</html>
```

```
method: 'GET',

// ส่งตัวแปรไปให้ด้วย ดังนี้
params: {x: 10, y: 20}
})

// หากทำงานสำเร็จ
.success(function(data) {

    // ให้เก็บผลจาก server ลงตัวแปร output
    $scope.output = data;
});

// สร้าง function ชื่อ callService2
$scope.callService2 = function() {

    // เรียกใช้ตัวแปร $http
    $http({

        // ส่งค่าไปยัง server
        url: 'ch1-ex21-call-server-by-event1.php',

        // กำหนดรูปแบบเป็น POST
        method: 'POST',

        // ส่งตัวแปรไปดังนี้
        params: {x: 15, y: 25}
    })

    // หากทำงานสำเร็จ
    .success(function(data) {

        // เก็บค่าจาก server ลงตัวแปร output
        $scope.output = data;
    });
}

<!-- ผูกเข้ากับ module ชื่อ myApp --&gt;
&lt;body ng-app="myApp"&gt;

<!-- ผูกเข้ากับ controller ชื่อ MyController --&gt;
&lt;div ng-controller="MyController"&gt;

    <!-- แสดงค่าตัวแปร output --&gt;
    &lt;div&gt;{{ output }}&lt;/div&gt;
    &lt;div&gt;

        <!-- สร้างปุ่มกด ให้ไปเรียก function ชื่อ callService1() --&gt;
        &lt;input type="button" value="Service1" ng-click="callService1()"&gt;
    </pre>
```

```
<!-- สร้างปุ่มกด ให้ไปเรียก function ชื่อ callService2() -->
<input type="button" value="Service2" ng-click="callService2()">
</div>
</div>
</body>
</html>
```

โค้ดฝั่ง server

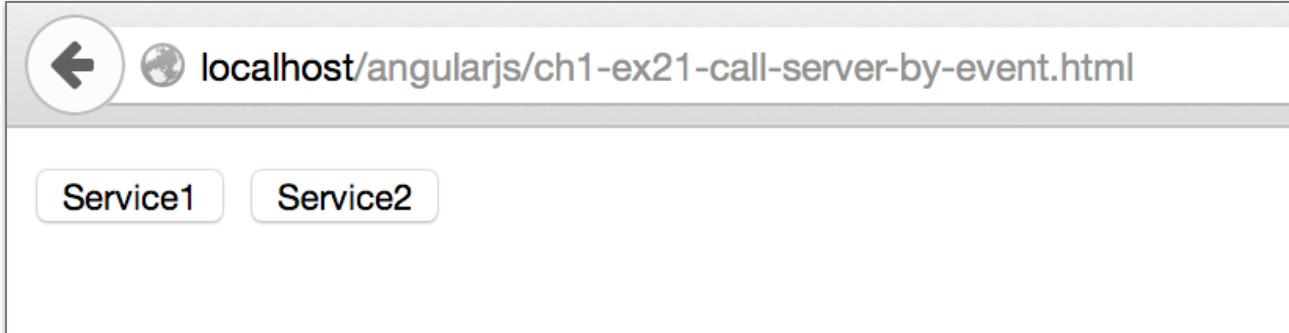
ch1-ex21-call-server-by-event1.php

```
<?php
print_r($_GET);
?>
```

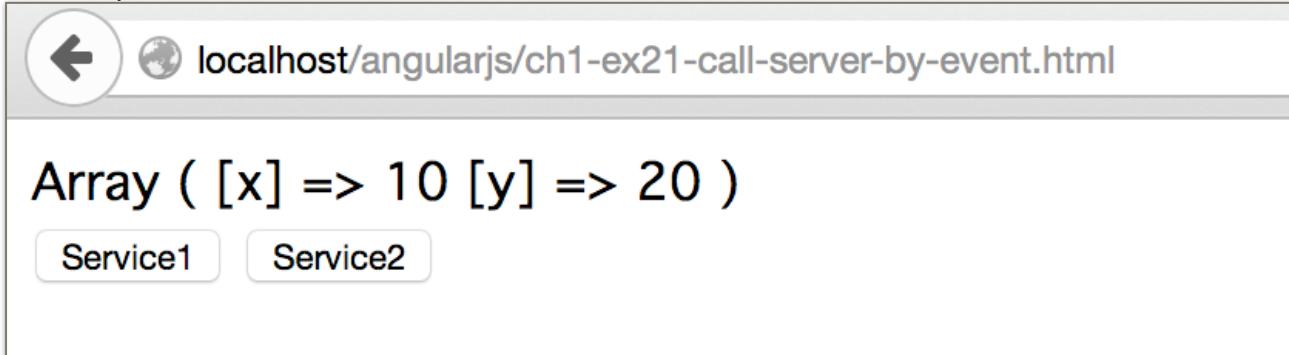
ch1-ex21-call-server-by-even2.php

```
<?php
print_r($_POST);
?>
```

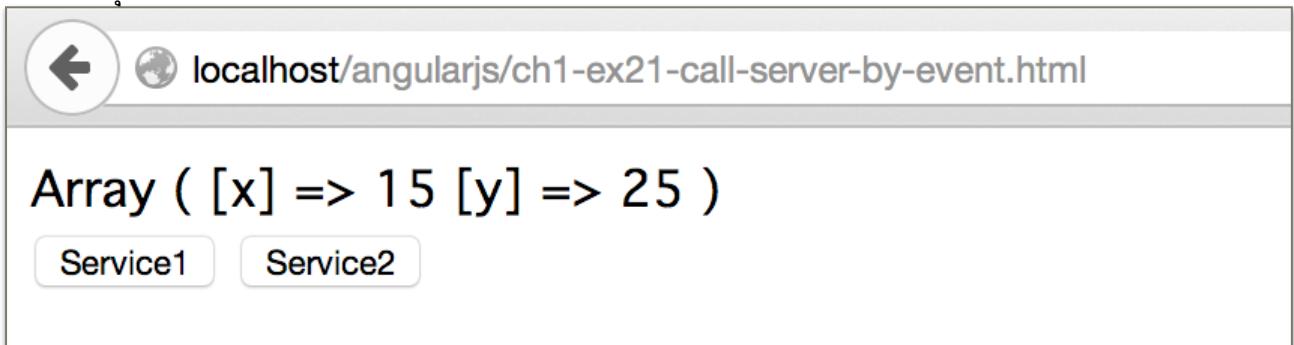
ผลการทำงาน



เมื่อคลิกปุ่มด้านซ้าย



เมื่อคลิกปุ่มด้านขวา



อธิบายเพิ่มเติม

เริ่มต้นเราทำการสร้าง module ชื่อว่า myApp จากนั้นก็ทำการสร้าง controller ชื่อว่า MyController ให้มีการรับ parameter จำนวน 2 ตัวคือ \$scope กับ \$http เพื่อเอาไว้ใช้งานต่อไป

จากนั้นทำการสร้าง function ชื่อ callService1 และเราก็ทำการเรียกใช้งานตัวแปร \$http โดยกำหนดให้ไปเรียกใช้งานโค้ดผัง server ที่ url ch1-ex21-call-server-by-event1.php กำหนดการส่งเป็นแบบ GET และมีการส่ง params x=10 และ y=20 ไปให้ server ด้วย เมื่อฟังก์ชัน server ตอบกลับมา ก็ให้เอาค่า data ที่ได้ไปใส่ลงในตัวแปร output ของ \$scope

เสร็จแล้วก็ทำการสร้าง function ชื่อ callService2 การทำงานก็จะเหมือนกันกับ callService1 เกือบๆ กันอย่าง ต่างก็แค่กำหนดให้ส่งเป็นแบบ POST กับค่าของตัวแปรที่ส่งก็ต่างกันเล็กน้อย

สุดท้ายก็ไม่มีอะไรมาก เราทำการผูก body เข้ากับ module myApp ด้วย ng-app="myApp" และผูก controller จากนั้นก็มีการสั่งแสดงค่าตัวแปร {{ output }} และผูก event ให้ปุ่มกดไปเรียก ng-click เพียงเท่านี้ โปรแกรมของเราก็ทำงานได้เสร็จสมบูรณ์แล้วครับ

AngularJS and Form

การใช้งาน AngularJS ร่วมกับ Form นั้นจะทำให้เราสามารถดักจับแบบข้อมูลที่เข้ามาได้ (การ validation) นอกจากนี้ยังทำให้เขียนโค้ดร่วมกับ model, controller ของ AngularJS ได้ง่ายด้วย โดยทั่วไป จะใช้การ submit ผ่านคำสั่ง ng-submit="functionName()" และผูกค่าของ input แต่ละตัวเข้ากับ model ที่เป็น object ลองมาดูตัวอย่าง โค้ดการสร้างฟอร์มแบบง่ายๆ และรับส่งค่า พร้อมตรวจสอบสถานะการ submit ดังนี้ครับ

ตัวอย่าง โค้ด

ch1-ex22-form.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ User
      app.controller('User', function($scope) {
```

```
// กำหนดตัวแปร user เป็นแบบ object
$scope.user = {};

// กำหนดตัวแปร wasSubmitted เป็น false
$scope.wasSubmitted = false;

// สร้าง function ชื่อ submit
$scope.submit = function() {

    // กำหนดค่าตัวแปร wasSubmitted เป็น true
    $scope.wasSubmitted = true;
}

});

</script>
</head>

<!-- ผูกเข้ากับ module ชื่อ myApp -->
<body ng-app="myApp">

<!-- ผูกเข้ากับ controller ชื่อ User -->
<div ng-controller="User">

    <!-- กำหนด event กรณี submit ให้ไปเรียก function submit() -->
    <form ng-submit="submit()" novalidate>

        <!-- แสดงค่าตัวแปร wasSubmitted -->
        {{ wasSubmitted }}

        <div>
            <label>Firstname</label>

            <!-- ผูกเข้ากับ model user.firstName -->
            <input type="text" ng-model="user.firstName">
        </div>

        <div>
            <label>Lastname</label>

            <!-- ผูกเข้ากับ model user.lastName -->
            <input type="text" ng-model="user.lastName">
        </div>

        <div>
            <label>Age</label>

            <!-- ผูกเข้ากับ model user.age -->
            <input type="text" ng-model="user.age">
        </div>

        <button>Submit</button>
    </form>
</div>
</body>
```

</html>

ผลการทำงาน

false

Firstname

Lastname

Age

Submit

เมื่อกรอกข้อมูล แล้วคลิกปุ่ม Submit

true

Firstname

Lastname

Age

Submit

อธิบายเพิ่มเติม

ตัวอย่างนี้ไม่มีอะไรมากครับ เริ่มต้นเราก็สร้าง module ชื่อว่า myApp จากนั้นก็สร้าง controller ชื่อว่า User ส่วนภายใน controller ก็ทำการสร้างตัวแปร user เป็นแบบ object โดยกำหนดให้เป็น user = {}; ไว้ก่อน จากนั้นก็สร้างตัวแปร wasSubmitted กำหนดค่าเป็น false ไว้ก่อน

ตามด้วยการสร้าง function ชื่อ submit ในนี้จะทำการเปลี่ยนค่าตัวแปร wasSubmitted จาก false ให้เป็น true เท่านั้นครับ ส่วนโค้ด html ที่เหลือไม่มีอะไรมาก แค่ผูกค่าเข้ากับ module myApp และก็ผูกเข้ากับ

controller ชื่อ User และสร้างฟอร์มมาผูก event submit เข้ากับ function submit() ด้วยคำสั่ง ng-submit="submit()" และกำหนดว่า ไม่มีการตักความถูกต้องของข้อมูล เลยใส่คำว่า novalidate เข้าไป ส่วนที่เหลือทำการสร้าง input ต่างๆ มาผูกเข้ากับ model ตัวแปรชื่อ user เพียงเท่านี้โปรแกรมก็ทำงานได้แล้วครับ

Form Validate

ในการตรวจสอบข้อมูลนี้ทาง AngularJS ได้ออกแบบมาให้ใช้งานง่ายมาก และทำงานทันทีเมื่อการกรอกข้อมูลใดๆ ลงไป โดยไม่ต้องรอให้กดปุ่ม submit และที่สำคัญมาก คือทำการ focus ส่วนที่ยังไม่ผ่านการตรวจสอบให้เราได้อีกด้วย ซึ่งสะดวกมากๆ ลองมาดูตัวอย่างโค้ด และผลการทำงานกันครับ

ตัวอย่างโค้ด

ch1-ex23-form-validate.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ User
      app.controller('User', function($scope) {

        // สร้างตัวแปร user เป็นแบบ object
        $scope.user = {};

        // สร้างตัวแปร wasSubmitted เป็น false
        $scope.wasSubmitted = false;

        // สร้าง function ชื่อ submit
        $scope.submit = function() {

          // เปลี่ยนค่าตัวแปร wasSubmitted เป็น true
          $scope.wasSubmitted = true;
        }
      });
    </script>
  </head>

  <!-- ผูกเข้ากับ module myApp -->
  <body ng-app="myApp">

    <!-- ผูกเข้ากับ controller ชื่อ User -->
    <div ng-controller="User">

      <!-- ผูกเข้ากับ event submit -->
      <form ng-submit="submit()">

        <!-- แสดงค่าตัวแปร wasSubmitted -->

```

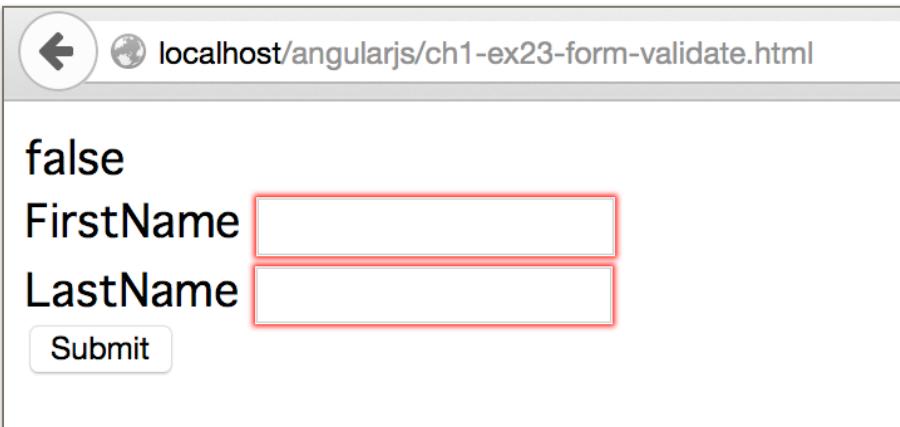
```
{{ wasSubmitted }}
```

```
<div>
  <label>FirstName</label>

  <!-- ผูกเข้ากับตัวแปร user.firstName และกำหนดว่าต้องกรอก -->
  <input type="text" ng-model="user.firstName" required />
</div>
<div>
  <label>LastName</label>

  <!-- ผูกเข้ากับตัวแปร user.lastName และกำหนดว่าต้องกรอก -->
  <input type="text" ng-model="user.lastName" required />
</div>
<div>
  <button>Submit</button>
</div>
</form>
</div>
</body>
</html>
```

ผลการทำงาน



localhost/angularjs/ch1-ex23-form-validate.html

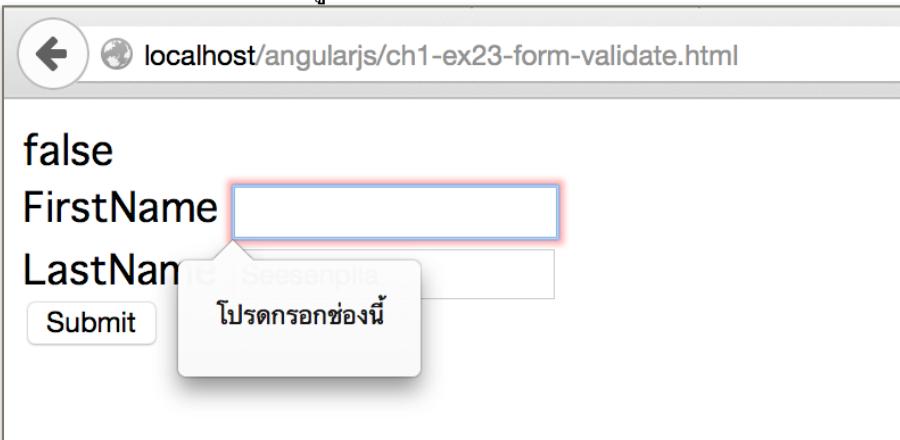
false

FirstName

LastName

Submit

ผลการทำงานเมื่อกรอกข้อมูลไม่ครบถ้วน



localhost/angularjs/ch1-ex23-form-validate.html

false

FirstName

LastNan

Submit

โปรดกรอกช่องนี้

เมื่อกรอกข้อมูลครบถ้วน

localhost/angularjs/ch1-ex23-form-validate.html

true ← ค่าจะกล้ายเป็น true

FirstName Tavon

LastName Seesepila

Submit

อธิบายเพิ่มเติม

เริ่มต้นเราทำการสร้าง module ชื่อ `myApp` ขึ้นมา จากนั้นทำการสร้าง controller ชื่อว่า `User` ให้มีการทำงานภาย ในดังนี้

กำหนดตัวแปร `user` เป็นแบบ `object` ด้วยคำสั่ง `$scope.user = {};` จากนั้นกำหนดตัวแปร `$scope.wasSubmitted = false` ไว้ก่อน เพื่อให้รู้ว่า ยังไม่มีการ submit ข้อมูลเข้ามาในฟอร์ม

จากนั้นก็สร้าง `function` ชื่อว่า `submit` ภาย ในการทำงานของ `function` ก็คือสั่งเปลี่ยนค่าตัวแปร `wasSubmitted` ให้เป็น `true`

ส่วนของ `html code` นั้นก็มีการสั่งผูก `body` เข้ากับ `module myApp` ด้วยคำสั่ง `ng-app="myApp"` จากนั้นก็ทำการผูก `controller` ไปที่ `User` และสร้าง `form input` ต่างๆ ให้ผูกเข้ากับค่าของ `model` สิ่งที่เพิ่มเติม คือคำว่า `required` ความหมายก็คือ ให้ช่องนั้นๆ ต้องมีการรับข้อมูลเข้ามา จึงจะให้ทำการ `submit` `form` ได้ นั่นเองครับ

Display Form Validation

ในการนี้ที่เราต้องการแสดงข้อความแจ้งเตือน เพื่ออธิบายเพิ่มเติม ว่าทำไมจึงไม่ผ่านการตรวจสอบ ก็ สามารถทำได้ครับ โดยใช้ `ng-show` เข้ามาช่วย แล้วใส่เงื่อนไขเข้าไปอีกเล็กน้อย คือใช้ `$invalid` กับ `$dirty` เข้ามาช่วย โดยความหมายเป็นดังนี้

`$invalid` = จะตรวจสอบเฉพาะตอนที่มีการป้อนข้อมูลเข้าไปแล้ว

`$dirty` = ให้ตรวจสอบทันที ตั้งแต่ `form` เริ่มโหลด

ดังนั้นเพื่อให้เห็นการใช้งาน `form validation` ผู้อ่านจำเป็นต้องดูในไฟล์ครับ

ตัวอย่างโค้ด

ch1-ex24-form-validation.html

```
<html>
<head>
  <!-- กำหนดแบบอักษรเป็น utf-8 -->
  <meta charset="utf-8" />

  <script src="angular-1.4.3/angular.js"></script>
```

```
<!-- กำหนด Style Sheet -->
<style>

/* ช่องที่ไม่ผ่านการตรวจ ให้พื้นหลังสีแดง */
input.ng-invalid.ng-dirty {
    background-color: red;
}

/* ช่องที่ผ่านการตรวจ ให้พื้นหลังสีเขียว */
input.ng-valid.ng-dirty {
    background-color: green;
}
</style>
</head>

<!-- กำหนดเป็น Angular Page -->
<body ng-app>

<!-- สร้าง form ชื่อว่า form -->
<form name="form">
    <div>
        <label>FirstName</label>

        <!-- กำหนดให้เชื่อมกับ user.firstName -->
        <input type="text" name="firstName" ng-model="user.firstName" required>

        <!-- ล็อกให้แสดงถ้าเมื่อ ไม่ผ่านการตรวจสอบ เท่านั้น -->
        <span ng-show="form.firstName.$invalid && form.firstName.$dirty">
            กรอกช่องนี้ด้วย
        </span>
    </div>
</form>
</body>
</html>
```

ผลการทำงาน



เมื่อทำการป้อนข้อมูล

A screenshot of a web browser window. The address bar shows 'localhost/angularjs/ch1-ex24-form-validation.html'. Below the address bar is a form with a text input field. The input field contains the text 'tavon' and has a green background color.

เมื่อลงลิ่งที่ป้อนออกไป

A screenshot of a web browser window. The address bar shows 'localhost/angularjs/ch1-ex24-form-validation.html'. Below the address bar is a form with a text input field. The input field is empty and has a red background color. To the right of the input field, the text 'กรอกช่องนี้ด้วย' (Please enter this field) is displayed.

Filtering and Sorting

การกรองข้อมูลจากตัวแปรที่มีค่าเป็นแบบ array ในบางครั้งเราอาจจะกรองจาก js ที่มีข้อมูลอยู่ แต่บางครั้งเราก็อ่านมาจาก json data ที่ server ทำการ response มาให้ ในตัวอย่างนี้จะพำทำกำรใช้งาน filter เพื่อค้นหาและกรองข้อมูล พร้อมทำการจัดเรียง โดยจะใช้การอ่านตัวแปรจาก js data ที่มีอยู่แล้วครับ

ตัวอย่างโค๊ด

ch1-ex25-filter-and-sorting.html

```
<html>
  <head>
    <script src="angular-1.4.3/angular.js"></script>
    <script>
      // สร้าง module ชื่อ myApp
      var app = angular.module('myApp', []);

      // สร้าง controller ชื่อ User
      app.controller('User', function($scope) {
        // สร้างตัวแปร friends เป็น array เก็บ object
        $scope.friends = [
          {name: 'Tavon'},
          {name: 'Hope'},
          {name: 'Thongchai'},
          {name: 'Somchai'},
          {name: 'Bualoi'}
        ];
      });
    </script>
```

```
</head>

<!-- ผูกเข้ากับ module myApp -->
<body ng-app="myApp">

    <!-- ผูกเข้ากับ controller ชื่อ User -->
    <div ng-controller="User">
        <form>

            <!-- สร้างช่อง input ผูกเข้ากับ query -->
            <input type="text" ng-model="query" autofocus />
        </form>

        <!-- ทำการ loop และดึงค่าจาก friends และกรองตาม query และเรียงด้วย name -->
        <ul ng-repeat="friend in friends | filter: query | orderBy: 'name'">

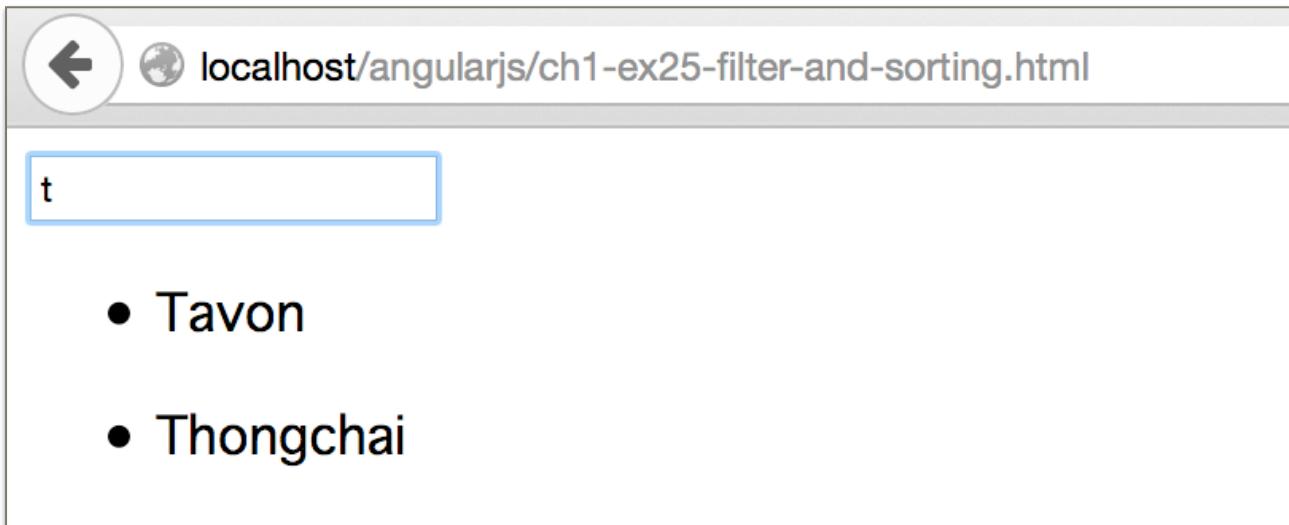
            <!-- 显示朋友的名字 -->
            <li>{{ friend.name }}</li>
        </ul>
    </div>
</body>
</html>
```

ผลการทำงาน

The screenshot shows a web browser window with the following details:

- URL Bar:** localhost/angularjs/ch1-ex25-filter-and-sorting.html
- Search Input:** An input field containing a single vertical bar character '|'. This is highlighted with a blue border.
- List Content:** An ordered list (
) containing five items:
 - Bualoi
 - Hope
 - Somchai
 - Tavon
 - Thongchai

ผลการทำงานเมื่อมีการป้อนข้อความลงใน



อธิบายเพิ่มเติม

เริ่มต้นเราทำการสร้าง module ชื่อว่า `myApp` จากนั้นก็ทำการสร้าง controller ชื่อว่า `User` และทำการสร้างตัวแปร `friends` เป็น array ที่เก็บข้อมูลแบบ object ไว้จำนวนหนึ่ง

ในส่วนของ html code เราทำการผูก body เข้ากับ module ชื่อว่า `myApp` และผูก div เข้ากับ controller ชื่อว่า `User` จากนั้นสร้างช่องกรอกข้อมูล ให้ผูกเข้ากับ model ชื่อ `query` และกำหนดให้เป็น `autofocus` ไว้ ต่อมาเราจะทำการวนรอบด้วย loop แบบ `ng-repeat` เพื่ออ่านตัวแปร `friends` มาเก็บไว้ใน `friend` และให้มันสามารถกรองได้ โดยใช้คำสั่ง `filter: query` จากนั้นก็สั่งให้เรียงตาม `name` ด้วยคำสั่ง `orderBy: 'name'`