AngularJS v2.0 18/08/2015

<BASE>

C'est un Framework JavaScript
Il peut être ajouté à une page HTML avec une balise <script>.
Angular.JS étend les attributs HTML avec les directives, et se lie à des données HTML avec des expressions.

Ajout d'Angular.JS:
```
<script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angular.min.js"></script>
```

<DIRECTIVES>

Angular.JS étend l'HTML avec les NG-directives :

- ✓ **ng-app :**
  Définit une application angularJS.
  The ng-app directive defines the root element of an AngularJS application.
  The ng-app directive will auto-bootstrap (automatically initialize) the application when a web page is loaded.
- ✓ **ng-model:**
  Lie la valeur de contrôles HTML (input, select, textarea) aux données de l'application.
  The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.
  The ng-model directive can also:
  Provide type validation for application data (number, email, required).
  Provide status for application data (invalid, dirty, touched, error).
  Provide CSS classes for HTML elements.
  Bind HTML elements to HTML forms.
- ✓ **ng-init** :
  Permet d'initialiser une ou des variables au lancement.
  The ng-init directive defines initial values for an AngularJS application.
  Normally, you will not use ng-init. You will use a controller or module instead.
- ✓ **ng-repeat** :
  Simule une boucle foreach (écrit une liste de valeurs par exemple)
  The **ng-repeat** directive **clones HTML elements** once for each item in a collection (in an array).
- ✓ **ng-bind** :
  Equivaut à l'utilisation de l'expression {{}}
  Data binding in AngularJS synchronizes AngularJS expressions with AngularJS data.
- ✓ **Ng-controller :**
  AngularJS controllers control the data of AngularJS applications.
  AngularJS controllers are regular JavaScript Objects.
  The ng-controller directive defines the application controller.

{{ firstName }} is synchronized with ng-model="firstName".
You can use **data-ng-**, instead of **ng-**, if you want to make your page HTML valid.

//EX :
```
<p>The name is
<span ng-bind="firstName"></span></p>
```
//Remplacé par :
```
<p>The name is
<span data-ng-bind="firstName"></span></p>
```

Liaison de données :
```
<div ng-app="" ng-init="quantity=1;price=5">
Quantity: <input type="number" ng-model="quantity">
Costs:   <input type="number" ng-model="price">
Total in dollar: {{ quantity * price }}
</div>
```

Répétition d'éléments HTML :
```
<div ng-app="" ng-init="names=['Jani','Hege','Kai']">
 <ul>
  <li ng-repeat="x in names">
   {{ x }}
  </li>
 </ul>
</div>
```

Répétition d'objets:
```
<div ng-app="" ng-init="names=[
{name:'Jani',country:'Norway'},
{name:'Hege',country:'Sweden'},
{name:'Kai',country:'Denmark'}]">
<ul>
 <li ng-repeat="x in names">
  {{ x.name + ', ' + x.country }}
 </li>
</ul>
</div>
```

<EXPRESSIONS>

AngularJS expressions are written inside double braces: {{ expression }}.
AngularJS will "output" data exactly where the expression is written.

AngularJS Expressions vs. JavaScript Expressions

Comme les expressions JavaScript, les expressions AngularJS :

- Peuvent contenir des littéraux, des opérateurs et des variables.

Contrairement aux expressions JavaScript, les expressions d'AngularJS :

- Peuvent être écrites à l'intérieur HTML.
- Ne supportent pas les conditions, boucles ou exceptions.
- Supportent les filtres.

Example {{ 5 + 5 }} or {{ firstName + " " + lastName }}

```
//EX STRING
<div ng-app="">
<p>Name:<input type="text" ng model="name"></p>
<p>{{name}}</p>
</div>
```

```
//OU EXEMPLE NUMBER
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar:
 {{ quantity * cost }}</p></div>
```

```
//OU EXEMPLE NUMBER 2
<div ng-app="" ng-init="quantity=1;cost=5">
<p>Total in dollar:
 <span ng-bind="quantity * cost"></span></p>
</div>
```

```
//OU EXEMPLE OBJECT
<div ng-app="" ng-init="
person={firstName:'John',lastName:'Doe'}">
<p>The name is
<span ng-bind="person.lastName"></span></p>
</div>
```

```
//OU EXEMPLE ARRAY
<div ng-app="" ng-init="points=[1,15,19,2,40]">
<p>The third result is
<span ng-bind="points[2]"></span></p>
</div>
```

<CONTROLLERS >

AngularJS applications are controlled by controllers.
The ng-controller directive defines the application controller.
A controller is a JavaScript Object, created by a standard JavaScript object constructor.

```
//EX : Exemple d'application
<div ng-app="myApp" ng-controller="myCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.firstName= "John";
    $scope.lastName= "Doe";
});
</script>
```

//EXPLAINS

- The AngularJS application is defined by **ng-app="myApp"**. The application runs inside the .
- The **ng-controller="myCtrl"** attribute is an AngularJS directive. It defines a controller.
- The **myCtrl function** is a JavaScript function.
- AngularJS will invoke the controller with a $scope object.
- In AngularJS, $scope is the application object (the owner of application variables and functions).
- The controller creates two properties (variables) in the scope (firstName and lastName).
- The **ng-model directives bind** the input fields to the controller properties (firstName and lastName).

Controllers Methods :

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{fullName()}}

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    }
});
</script>
```

Controllers in external files :

```
<div ng-app="myApp" ng-controller="personCtrl">

First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}

</div>

<script src="personController.js"></script>
```

```
//FILES personController.js
angular.module('myApp', []).controller('personCtrl',
function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Doe",
    $scope.fullName = function() {
        return $scope.firstName + " " + $scope.lastName;
    }
});
```

Controllers in external files2 :

```
<div ng-app="myApp" ng-controller="namesCtrl">
<ul>
  <li ng-repeat="x in names">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>
</div>
<script src="namesController.js"></script>
```

```
//FILES namesController.js
angular.module('myApp', []).controller('namesCtrl',
function($scope) {
    $scope.names = [
        {name:'Jani',country:'Norway'},
        {name:'Hege',country:'Sweden'},
        {name:'Kai',country:'Denmark'}
    ];
});
```

<FILTERS>

AngularJS filters can be used to transform data:

Filter	Description
currency	Format a number to a currency format.
filter	Select a subset of items from an array.
lowercase	Format a string to lower case.
orderBy	Orders an array by an expression.
uppercase	Format a string to upper case.

Adding Filters to Expressions

```
<div ng-app="myApp" ng-controller="personCtrl">
<p>The name is {{ lastName | uppercase }}</p>
</div>
```

The currency Filter

```
<div ng-app="myApp" ng-controller="costCtrl">
<input type="number" ng-model="quantity">
<input type="number" ng-model="price">
<p>Total = {{ (quantity * price) | currency }}</p>
</div>
```

Adding Filters to Directives

```
<div ng-app="myApp" ng-controller="namesCtrl">
<ul>
  <li ng-repeat="x in names | orderBy:'country'">
    {{ x.name + ', ' + x.country }}
  </li>
</ul>
<div>
```

Filtering Input

```
<div ng-app="myApp" ng-controller="namesCtrl">
<p><input type="text" ng-model="test"></p>
<ul>
  <li ng-repeat="x in names | filter:test | orderBy:'country'">
    {{ (x.name | uppercase) + ', ' + x.country }}
  </li>
</ul>
</div>
```

<AJAX - $HTTP>

$http is an AngularJS service for reading data from remote servers.
AngularJS $http is a core service for reading data from web servers.
$http.get(url) is the function to use for reading server data.

```
//EXEMPLE :
<div ng-app="myApp" ng-controller="customersCtrl">
<ul>
  <li ng-repeat="x in names">
    {{ x.Name + ', ' + x.Country }}
  </li>
</ul>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
    $http.get("http://www.w3schools.com
    /angular/customers.php")
    .success(function(response) {$scope.names =
response.records;});
});
</script>
```

Application explained:

- The AngularJS application is defined by **ng-app**. The application runs inside a .
- The **ng-controller** directive names the **controller object**.
- The **customersCtrl** function is a standard JavaScript **object constructor**.
- AngularJS will invoke customersCtrl with a **$scope** and **$http** object.
- $scope is the **application object** (the owner of application variables and functions).
- $http is an **XMLHttpRequest object** for requesting external data.
- **$http.get()** reads **JSON data** from http://www.w3schools.com/angular/customers.php.
- If **success**, the controller creates a property (**names**) in the scope, with JSON data from the server.

<TABLES>

The ng-repeat directive is perfect for displaying tables.

Displaying Data in a Table :

```
<style>
table, th , td {
  border: 1px solid grey;
  border-collapse: collapse;
  padding: 5px;
}
table tr:nth-child(odd) {
  background-color: #f1f1f1;
}
table tr:nth-child(even) {
  background-color: #ffffff;
}
</style>
<div ng-app="myApp" ng-controller="customersCtrl">
<table>
  <tr ng-repeat="x in names | orderBy : 'Country'">
    <td>{{ $index + 1 }}</td>
    <td>{{ x.Name | uppercase }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {
        $http.get("http://www.w3schools.com/
        angular/customers.php")
        .success(function (response)
        {$scope.names = response.records;});
});
</script>
```

Using $even and $odd :

```
<table>
  <tr ng-repeat="x in names">
    <td ng-if="$odd" style="background-color:#f1f1f1">
    {{ x.Name }}</td>
    <td ng-if="$even">
    {{ x.Name }}</td>
    <td ng-if="$odd" style="background-color:#f1f1f1">
    {{ x.Country }}</td>
    <td ng-if="$even">
    {{ x.Country }}</td>
  </tr>
</table>
```

<SQL>

Fetching Data From a PHP Server Running MySQL

```
<div ng-app="myApp" ng-controller="customersCtrl">

<table>
  <tr ng-repeat="x in names">
    <td>{{ x.Name }}</td>
    <td>{{ x.Country }}</td>
  </tr>
</table>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('customersCtrl', function($scope, $http) {

$http.get("http://www.w3schools.com/angular/customers_mysql.php")
    .success(function (response) {$scope.names = response.records;});
});
</script>
```

Requests for data from a different server (than the requesting page), are called cross-site HTTP requests.

Cross-site requests are common on the web. Many pages load CSS, images, and scripts from different servers.

In modern browsers, cross-site HTTP requests from scripts are restricted to same site for security reasons.

The following line, in our PHP examples, has been added to allow cross-site access.

header("Access-Control-Allow-Origin: *");

//Exemple COMPLET (Using PHP and MySQL. Returning JSON.):

```
<?php
header("Access-Control-Allow-Origin: *");
header("Content-Type: application/json; charset=UTF-8");

$conn = new mysqli("myServer", "myUser", "myPassword",
"Northwind");

$result = $conn->query("SELECT CompanyName, City,
Country FROM Customers");

$outp = "";
while($rs = $result->fetch_array(MYSQLI_ASSOC)) {
    if ($outp != "") {$outp .= ",";}
    $outp .= '{"Name":"'  . $rs["CompanyName"] . '",';
    $outp .= '"City":"'   . $rs["City"]        . '",';
    $outp .= '"Country":"'. $rs["Country"]     . '"}';
}
$outp ='{"records":['.$outp.']}';
$conn->close();

echo($outp);
?>
```

<HTML DOM>

AngularJS has directives for binding application data to the attributes of HTML DOM elements.

The ng-disabled Directive

```
<div ng-app="">
<p>
<button ng-disabled="mySwitch">Click Me!</button>
</p>
<p>
<input type="checkbox" ng-model="mySwitch">Button
</p>
</div>
```

The ng-show Directive

```
<div ng-app="">
<p ng-show="true">I am visible.</p>
<p ng-show="false">I am not visible.</p>
</div>
```

The ng-hide Directive

```
<div ng-app="" ng-init="hour=13">
<p ng-show="hour > 12">I am visible.</p>
</div>
```

<EVENTS>

AngularJS has its own HTML events directives.

The ng-click Directive :

```
<div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="count = count + 1">Click Me!</button>
<p>{{ count }}</p>
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
    $scope.count = 0;
});
</script>
```

Hiding HTML Elements :

```
<div ng-app="myApp" ng-controller="personCtrl">

<button ng-click="toggle()">Toggle</button>

<p ng-hide="myVar">
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Doe"
    $scope.myVar = false;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    };
});
</script>
```

Application explained:

- The first part of the personController is the same as in the chapter about controllers.
- The application has a default property (a variable): $scope.myVar = false;
- The ng-hide directive sets the visibility, of a <p> element with two input fields, according to the value (true or false) of myVar.
- The function toggle() toggles myVar between true and false.
- The value ng-hide="true" makes the element invisible.

Showing HTML Elements :

```
<div ng-app="myApp" ng-controller="personCtrl">

<button ng-click="toggle()">Toggle</button>

<p ng-show="myVar">
First Name: <input type="text" ng-model="firstName"><br>
Last Name: <input type="text" ng-model="lastName"><br>
<br>
Full Name: {{firstName + " " + lastName}}
</p>

</div>

<script>
var app = angular.module('myApp', []);
app.controller('personCtrl', function($scope) {
    $scope.firstName = "John",
    $scope.lastName = "Doe"
    $scope.myVar = true;
    $scope.toggle = function() {
        $scope.myVar = !$scope.myVar;
    }
});
</script>
```

<MODULES>

A Module With One Controller

```
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script>
var app = angular.module("myApp", []);
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName = "Doe";
});
</script>
```

Modules and Controllers in Files

```
<div ng-app="myApp" ng-controller="myCtrl">
{{ firstName + " " + lastName }}
</div>

<script src="myApp.js"></script>
<script src="myCtrl.js"></script>
```

```
//myApp.js
var app = angular.module("myApp", []);
```

```
//myCtrl.js
app.controller("myCtrl", function($scope) {
    $scope.firstName = "John";
    $scope.lastName= "Doe";
});
```

<FORMS>

An AngularJS form is a collection of input controls.

```
<div ng-app="myApp" ng-controller="formCtrl">
  <form novalidate>
    First Name:<br>
    <input type="text" ng-model="user.firstName"><br>
    Last Name:<br>
    <input type="text" ng-model="user.lastName">
    <br><br>
    <button ng-click="reset()">RESET</button>
  </form>
  <p>form = {{user }}</p>
  <p>master = {{master}}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('formCtrl', function($scope) {
    $scope.master = {firstName:"John", lastName:"Doe"};
    $scope.reset = function() {
        $scope.user = angular.copy($scope.master);
    };
    $scope.reset();
});
</script>
```

Example Explained :

The **ng-app** directive defines the AngularJS application.
The **ng-controller** directive defines the application controller.
The **ng-model** directive binds two input elements to the user object in the model.
The **formCtrl** function sets initial values to the **master** object, and defines the reset() method.
The **reset()** method sets the **user** object equal to the master object.
The **ng-click** directive invokes the **reset()** method, only if the button is clicked.
The novalidate attribute is not needed for this application, but normally you will use it in AngularJS forms, to override standard HTML5 validation.

<INPUT VALIDATION>

AngularJS forms and controls can validate input data.

```
<form ng-app="myApp" ng-controller="validateCtrl"
name="myForm" novalidate>

<p>Username:<br>
  <input type="text" name="user" ng-model="user" required>
  <span style="color:red" ng-show="myForm.user.$dirty &&
myForm.user.$invalid">
  <span ng-show="myForm.user.$error.required">Username is
required.</span>
  </span>
</p>

<p>Email:<br>
<input type="email" name="email" ng-model="email"
required>
<span style="color:red" ng-show="myForm.email.$dirty &&
myForm.email.$invalid">
<span ng-show="myForm.email.$error.required">Email is
required.</span>
<span ng-show="myForm.email.$error.email">Invalid email
address.</span>
  </span>
</p>

<p>
<input type="submit"
ng-disabled="myForm.user.$dirty && myForm.user.$invalid ||
myForm.email.$dirty && myForm.email.$invalid">
</p>

</form>

<script>
var app = angular.module('myApp', []);
app.controller('validateCtrl', function($scope) {
    $scope.user = 'John Doe';
    $scope.email = 'john.doe@gmail.com';
});
</script>
```

Example Explained

The AngularJS directive **ng-model** binds the input elements to the model.
The model object has two properties: **user** and **email**.
Because of **ng-show**, the spans with color:red are displayed only when user or email is **$dirty** and **$invalid**.

Property Description

$dirty	The user has interacted with the field.
$valid	The field content is valid.
$invalid	The field content is invalid.
$pristine	User has not interacted with the field yet.

<API>

PI stands for Application Programming Interface.
The AngularJS Global API is a set of global JavaScript functions for performing common tasks like:

- Comparing objects
- Iterating objects
- Converting data

The Global API functions are accessed using the angular object.

Below is a list of some common API functions:

API	Description
angular.lowercase()	Converts a string to lowercase
angular.uppercase()	Converts a string to uppercase
angular.isString()	Returns true if the reference is a string
angular.isNumber()	Returns true if the reference is a number

```
//EXEMPLE : angular.lowercase()
<div ng-app="myApp" ng-controller="myCtrl">
<p>{{ x1 }}</p>
<p>{{ x2 }}</p>
</div>

<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.x1 = "JOHN";
$scope.x2 = angular.lowercase($scope.x1);
});
</script>
```

<WITH BOOTSTRAP CSS>

To include Bootstrap in your AngularJS application, add the following line to the head of your document:

```
<link rel="stylesheet"
href="http://maxcdn.bootstrapcdn.com/bootstrap/3.2.0/css/
bootstrap.min.css">

<body ng-app="myApp" ng-controller="userCtrl">

<div class="container">

<h3>Users</h3>

<table class="table table-striped">
  <thead><tr>
    <th>Edit</th>
    <th>First Name</th>
    <th>Last Name</th>
  </tr></thead>
  <tbody><tr ng-repeat="user in users">
    <td>
```

```
    <button class="btn" ng-click="editUser(user.id)">
    <span class="glyphicon glyphicon-
pencil"></span>  Edit
    </button>
   </td>
   <td>{{ user.fName }}</td>
   <td>{{ user.lName }}</td>
  </tr></tbody>
</table>

<hr>
<button class="btn btn-success" ng-click="editUser('new')">
  <span class="glyphicon glyphicon-user"></span> Create
New User
</button>
<hr>

<h3 ng-show="edit">Create New User:</h3>
<h3 ng-hide="edit">Edit User:</h3>

<form class="form-horizontal">
<div class="form-group">
  <label class="col-sm-2 control-label">First Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="fName" ng-disabled="!edit"
placeholder="First Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Last Name:</label>
  <div class="col-sm-10">
    <input type="text" ng-model="lName" ng-disabled="!edit"
placeholder="Last Name">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Password:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw1"
placeholder="Password">
  </div>
</div>
<div class="form-group">
  <label class="col-sm-2 control-label">Repeat:</label>
  <div class="col-sm-10">
    <input type="password" ng-model="passw2"
placeholder="Repeat Password">
  </div>
</div>
</form>

<hr>
<button class="btn btn-success" ng-disabled="error ||
incomplete">
  <span class="glyphicon glyphicon-save"></span> Save
Changes
</button>
</div>

<script src = "myUsers.js"></script>
```

AngularJS Directive	Description
<body ng-app	Defines an application for the <body> element
<body ng-controller	Defines a controller for the <body> element
<tr ng-repeat	Repeats the <tr> element for each user in users
<button ng-click	Invoke the function editUser() when the <button> element is clicked
<h3 ng-show	Show the <h3>s element if edit = true
<h3 ng-hide	Hide the <h3> element if edit = true
<input ng-model	Bind the <input> element to the application
<button ng-disabled	Disables the <button> element if error or incomplete = true

myUsers.js

```
angular.module('myApp', []).controller('userCtrl',
function($scope) {
$scope.fName = '';
$scope.lName = '';
$scope.passw1 = '';
$scope.passw2 = '';
$scope.users = [
{id:1, fName:'Hege', lName:"Pege" },
{id:2, fName:'Kim',  lName:"Pim" },
{id:3, fName:'Sal',  lName:"Smith" },
{id:4, fName:'Jack', lName:"Jones" },
{id:5, fName:'John', lName:"Doe" },
{id:6, fName:'Peter',lName:"Pan" }
];
$scope.edit = true;
$scope.error = false;
$scope.incomplete = false;

$scope.editUser = function(id) {
  if (id == 'new') {
    $scope.edit = true;
    $scope.incomplete = true;
    $scope.fName = '';
    $scope.lName = '';
    } else {
    $scope.edit = false;
    $scope.fName = $scope.users[id-1].fName;
    $scope.lName = $scope.users[id-1].lName;
  }
};
$scope.$watch('passw1',function() {$scope.test();});
$scope.$watch('passw2',function() {$scope.test();});
$scope.$watch('fName', function() {$scope.test();});
$scope.$watch('lName', function() {$scope.test();});

$scope.test = function() {
  if ($scope.passw1 !== $scope.passw2) {
    $scope.error = true;
    } else {
    $scope.error = false;
  }
  $scope.incomplete = false;
  if ($scope.edit && (!$scope.fName.length ||
  !$scope.lName.length ||
  !$scope.passw1.length || !$scope.passw2.length)) {
     $scope.incomplete = true;
  }};});
```

JavaScript Code Explained :

Scope Properties	Used for
$scope.fName	Model variable (user first name)
$scope.lName	Model variable (user last name)
$scope.passw1	Model variable (user password 1)
$scope.passw2	Model variable (user password 2)
$scope.users	Model variable (array of users)
$scope.edit	Set to true when user clicks on create user.
$scope.error	Set to true if passw1 not equal passw2
$scope.incomplete	Set to true if any field is empty (length = 0)
$scope.editUser	Sets model variables
$scope.watch	Watches model variables
$scope.test	Tests model variables for errors and incompleteness

<INCLUDES>

Including a portion of HTML in HTML is, unfortunately, not **(yet)** supported by HTML.

HTML imports is a W3C suggestion http://www.w3.org for future versions of HTML:
<link rel="import" href="/path/navigation.html">

Client Side Includes

There are many ways to use JavaScript to include HTML in HTML.

The most common way is to use an http request (AJAX) to fetch data from a server, and then write the data to the innerHTML of an HTML element.

AngularJS Side Includes

With AngularJS, you can include HTML content using the ng-include directive:

//Example :
```
<body ng-app="myApp" ng-controller="userCtrl">

<div class="container">
  <div ng-include="'myUsers_List.htm'"></div>
  <div ng-include="'myUsers_Form.htm'"></div>
</div>

<script src= "myUsers.js"></script>

</body>
```

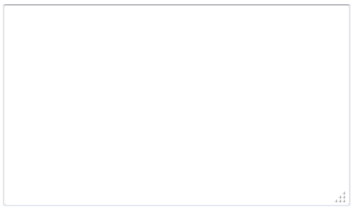
//Tous les autres fichiers :
http://www.w3schools.com/angular/angular_includes.asp

<APPLICATIONS>

- AngularJS **modules** define AngularJS applications.
- AngularJS **controllers** control AngularJS applications.
- The **ng-app** directive defines the application, the **ng-controller** directive defines the controller.

//APPLICATION EXEMPLE

My Note



Save Clear

Number of characters left: 100

```
<html ng-app="myNoteApp">
<div ng-controller="myNoteCtrl">

<h2>My Note</h2>

<p><textarea ng-model="message" cols="40"
rows="10"></textarea></p>

<p>
<button ng-click="save()">Save</button>
<button ng-click="clear()">Clear</button>
</p>

<p>Number of characters left: <span ng-
bind="left()"></span></p>
</div>

<script src="myNoteApp.js"></script>
<script src="myNoteCtrl.js"></script>
```

myNoteApp.js
```
var app = angular.module("myNoteApp", []);
```

myNoteCtrl.js
```
app.controller("myNoteCtrl", function($scope) {
   $scope.message = "";
   $scope.left  = function() {return 100 -
$scope.message.length;};
   $scope.clear = function() {$scope.message = "";};
   $scope.save  = function() {alert("Note Saved");};
});
```

<EXEMPLE TODO LIST>

My Todo List

Add New | Add New

Clean House

Remove marked

```
<!DOCTYPE html>
<html>
<script src=
"http://ajax.googleapis.com/ajax/libs/angularjs/1.3.14/angula
r.min.js"></script>
<body ng-app="myApp" ng-controller="todoCtrl">

<h2>My Todo List</h2>

<form ng-submit="todoAdd()">
   <input type="text" ng-model="todoInput" size="50"
placeholder="Add New">
   <input type="submit" value="Add New">
</form>

<br>

<div ng-repeat="x in todoList">
   <input type="checkbox" ng-model="x.done"> <span ng-
bind="x.todoText"></span>
</div>

<p><button ng-click="remove()">Remove
marked</button></p>

<script>
var app = angular.module('myApp', []);
app.controller('todoCtrl', function($scope) {
   $scope.todoList = [{todoText:'Clean House', done:false}];

   $scope.todoAdd = function() {
      $scope.todoList.push({todoText:$scope.todoInput,
done:false});
      $scope.todoInput = "";
   };

   $scope.remove = function() {
      var oldList = $scope.todoList;
      $scope.todoList = [];
      angular.forEach(oldList, function(x) {
         if (!x.done) $scope.todoList.push(x);
      });
   };
});
</script>

</body>
</html>
```

<REFERENCES>

Directive	Description
ng-app	Defines the root element of an application.
ng-bind	Binds the innerHTML of HTML elements to application data.
ng-click	Defines the behavior when an element is clicked.
ng-controller	Defines the controller object for an application.
ng-disabled	Binds application data to the HTML disabled attribute.
ng-hide	Hides or shows HTML elements.
ng-include	Includes HTML in an application.
ng-init	Defines initial values for an application.
ng-model	Binds the value of HTML controls to application data.
ng-repeat	Defines a template for each data in a collection.
ng-show	Shows or hides HTML elements.

Filter	Description
currency	Format a number to a currency format.
filter	Select a subset of items from an array.
lowercase	Format a string to lower case.
orderBy	Orders an array by an expression.
uppercase	Format a string to upper case.

AngularJS support the following events:

ng-click	ng-dbl-click
ng-mousedown	ng-mouseenter
ng-mouseleave	ng-mousemove
ng-keydown	ng-keyup
ng-keypress	ng-change

AngularJS Validation Properties

$dirty / $invalid / $error

AngularJS Global API

Converting

API	Description
angular.lowercase()	Converts a string to lowercase
angular.uppercase()	Converts a string to uppercase
angular.copy()	Creates a deep copy of an object or an array
angular.forEach()	Executes a function for each element in an object or array

Comparing

API	Description
angular.isArray()	Returns true if the reference is an array
angular.isDate()	Returns true if the reference is a date
angular.isDefined()	Returns true if the reference is defined
angular.isElement()	Returns true if the reference is a DOM element
angular.isFunction()	Returns true if the reference is a function
angular.isNumber()	Returns true if the reference is a number
angular.isObject()	Returns true if the reference is an object
angular.isString()	Returns true if the reference is a string
angular.isUndefined()	Returns true if the reference is undefined
angular.equals()	Returns true if two references are equal

JSON

API	Description
angular.fromJSON()	Deserializes a JSON string
angular.toJSON()	Serializes a JSON string

Basic

API	Description
angular.bootstrap()	Starts AngularJS manually
angular.element()	Wraps an HTML element as an jQuery element
angular.module()	Creates, registers, or retrieves an AngularJS module