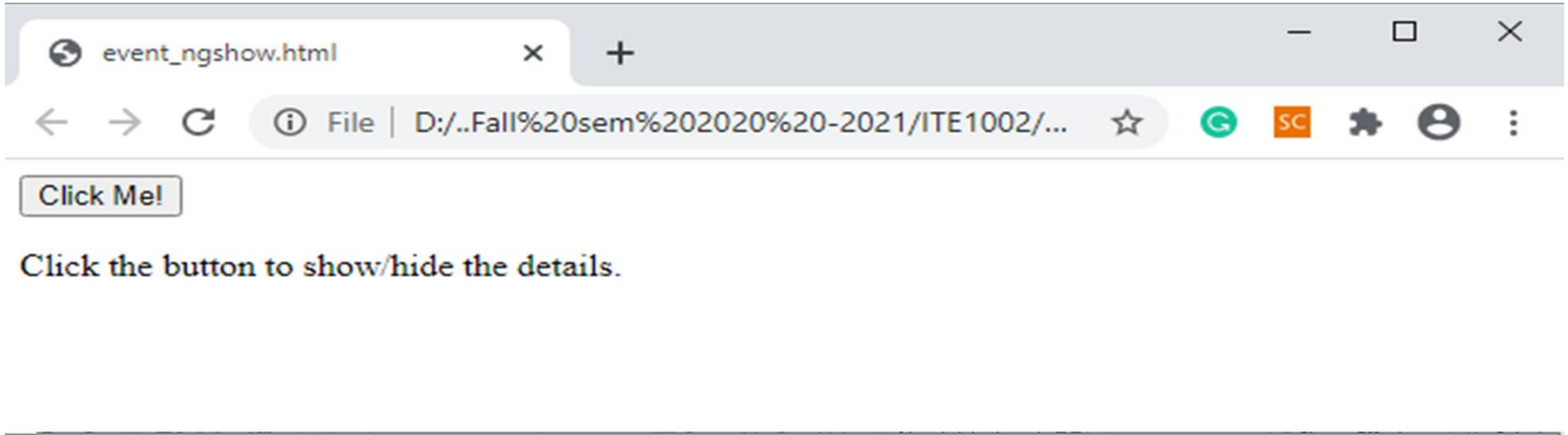
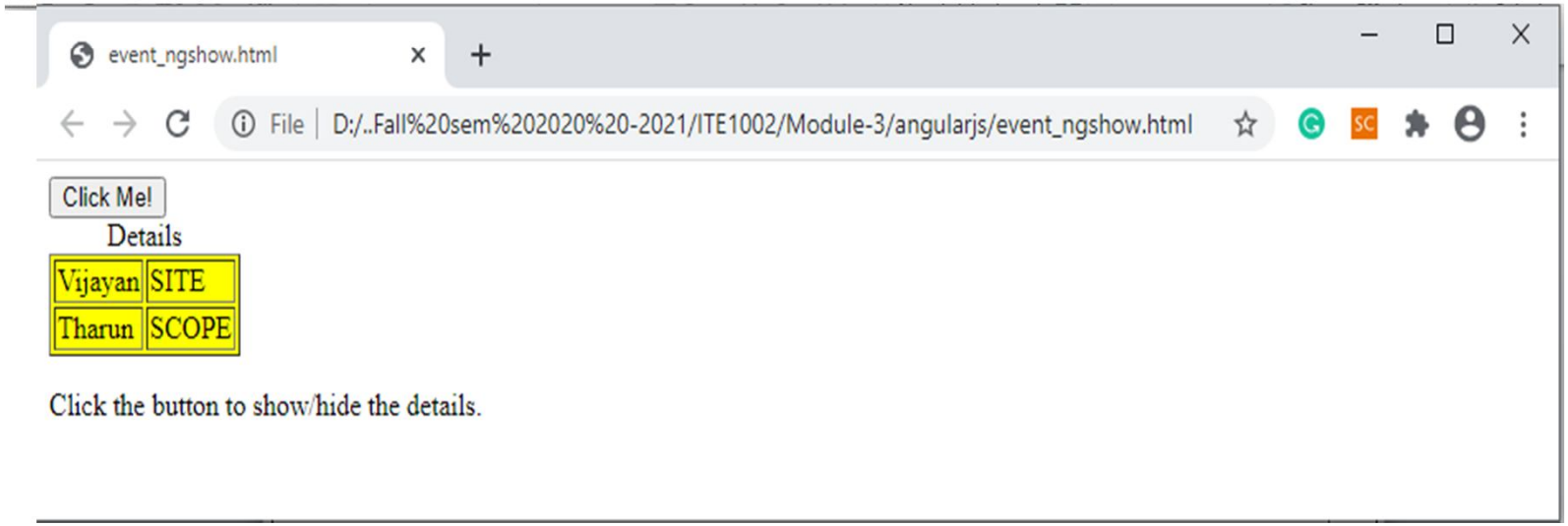
```
<!DOCTYPE html><html><script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js">
</script> <body><div ng-app="myApp" ng-controller="myCtrl">
<button ng-click="myFunc()">Click Me!</button>
<div ng-show="showMe">
<table border=1 bgcolor="yellow"><caption>Details</caption>
<tr><td>Vijayan</td><td>SITE</td></tr>
<tr><td>Tharun</td><td>SCOPE</td></tr></table>
</div> </div> <script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
   $scope.showMe = false;
   $scope.myFunc = function() {
      $scope.showMe = !$scope.showMe;
   } });</script> <p>Click the button to show/hide the details.</p>
</body></html>
```

Call function at click for show / hide


event_ngshow.html
File | D:/..Fall%20sem%202020%20-2021/ITE1002/...
Click Me!
Click the button to show/hide the details.


event_ngshow.html
File | D:/..Fall%20sem%202020%20-2021/ITE1002/Module-3/angularjs/event_ngshow.html
Click Me!
Details
Vijayan SITE
Tharun SCOPE
Click the button to show/hide the details.

# Angular JS Forms

- Forms in AngularJS provides **data-binding and validation of input controls.**
- Input controls are the HTML input elements:
  - input elements
  - select elements
  - button elements
  - Radiobutton
  - textarea elements

- Input controls provides data-binding by using the **ng-model** directive.

# ng-show and ng-hide

- The **ng-show** directive shows or hides an HTML element.

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

- The ng-show directive shows (or hides) an HTML element based on the **value** of ng-show.
- You can use any expression that evaluates to true or false:

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

- The **ng-hide** directive hides or shows an HTML element.

- **Text**

```
<input type="text" ng-model="firstname">
  <script>
  var app = angular.module('myApp', []);
  app.controller('formCtrl', function($scope) {
   $scope.firstname = "John";
  });
  </script>
```

- **Checkbox**

```
<form>
 Check to show a header:
 <input type="checkbox" ng-model="myVar">
</form>

<h1 ng-show="myVar">My Header</h1>
```

- **Radio**

```
Pick a topic:
  <input type="radio" ng-model="myVar" value="dogs">Dogs
  <input type="radio" ng-model="myVar" value="tuts">Tutorials
  <input type="radio" ng-model="myVar" value="cars">Cars

<div ng-switch="myVar">
  <div ng-switch-when="dogs">
    <h1>Dogs</h1>
    <p>Welcome to a world of dogs.</p>
  </div>
  <div ng-switch-when="tuts">
    <h1>Tutorials</h1>
    <p>Learn from examples.</p>
  </div>
  <div ng-switch-when="cars">
    <h1>Cars</h1>
    <p>Read about cars.</p>
  </div>
</div>
```

# AngularJS Select Boxes

- Creating a Select Box Using **ng-options**
- If you want to create a dropdown list, based on an object or an array in AngularJS, you should use the **ng-options** directive:
- ng-options vs ng-repeat
- You can also use the ng-repeat directive to make the same dropdown list

# AngularJS HTML DOM

- AngularJS has directives for binding application data to the attributes of HTML DOM elements.
- The ng-disabled Directive
- The **ng-disabled** directive binds AngularJS application data to the disabled attribute of HTML elements.

# AngularJS HTML DOM

- The **ng-disabled** directive binds the application data **mySwitch** to the HTML button's **disabled** attribute.
- The **ng-model** directive binds the value of the HTML checkbox element to the value of **mySwitch**.
- If the value of **mySwitch** evaluates to **true**, the button will be disabled:

- **Select**

```
Select a topic:
 <select ng-model="myVar">
  <option value="">
  <option value="dogs">Dogs
  <option value="tuts">Tutorials
  <option value="cars">Cars
 </select>
</form>

<div ng-switch="myVar">
 <div ng-switch-when="dogs">
  <h1>Dogs</h1>
  <p>Welcome to a world of dogs.</p>
 </div>
 <div ng-switch-when="tuts">
  <h1>Tutorials</h1>
  <p>Learn from examples.</p>
 </div>
 <div ng-switch-when="cars">
  <h1>Cars</h1>
  <p>Read about cars.</p>
 </div>
</div>
```

# Form Validation

- AngularJS offers client-side form validation.
- AngularJS monitors the state of the form and input fields (input, textarea, select), and lets you notify the user about the current state.
- AngularJS also holds information about whether they have been **touched, or modified, or not.**
- You can use standard HTML5 attributes to validate input, or you can make your own validation functions.

# Input Fields State

- AngularJS is constantly updating the state of both the form and the input fields.
- **Input fields** have the following states:
  - **$untouched** → The field has not been touched yet
  - **$touched** → The field has been touched
  - **$pristine** → The field has not been modified yet
  - **$dirty** → The field has been modified
  - **$invalid** → The field content is not valid
  - **$valid** → The field content is valid
- They are all properties of the input field, and are either **true or false**.

# Form State

- Forms have the following states:
  - **$pristine** → No fields have been modified yet
  - **$dirty** → One or more have been modified
  - **$invalid** → The form content is not valid
  - **$valid** → The form content is valid
  - **$submitted** → The form is submitted
- They are all properties of the form, and are either true or false.

# CSS Classes

- AngularJS adds CSS classes to forms and input fields depending on their states.
- The following classes are added to, or removed from, input fields:
  - **ng-untouched->** The field has not been touched yet
  - **ng-touched →**The field has been touched
  - **ng-pristine→** The field has not been modified yet
  - **ng-dirty→** The field has been modified
  - **ng-valid→** The field content is valid
  - **ng-invalid→** The field content is not valid
  - **ng-valid-key→** One *key* for each validation.
    - Example: ng-valid-required, useful when there are more than one thing that must be validated
  - **ng-invalid-key→** Example: ng-invalid-required

# CSS Classes

- The following classes are added to, or removed from, forms:
  - **ng-pristine** No fields has not been modified yet
  - **ng-dirty** One or more fields has been modified
  - **ng-valid** The form content is valid
  - **ng-invalid** The form content is not valid
  - **ng-valid-*key*** One *key* for each validation.
    - Example: ng-valid-required, useful when there are more than one thing that must be validated
  - **ng-invalid-*key*** Example: ng-invalid-required

The classes are removed if the value they represent is false.

# Validation directives

- **ng-required** directive to do the same thing. Using this directive you have the flexibility to set the input field should have a value or not.

  <input type="text" ng-required="true" />
- **ng-minlength** is used to validate the minimum length of the input value.

<input type="text" ng-minlength=10 />
- **ng-maxlength** is used to validate the maximum length of the input value.

<input type="text" ng-maxlength=20 />

# Validation directives

- **ng-pattern** directive is used to ensure that an input matches a **regex** pattern, the following syntax is used.

```
<input type="text" ng-pattern="[a-zA-Z]" />
```

- **Email→**We can set the input type to email to ensure that the input field is a valid email id.

```
<input type="email" name="email" ng-model="user.email" />
```

- **Number→**We can set the input type to number to ensure that the input field is a number.

```
<input type="number" name="age" ng-model="user.age" />
```

- **URL→**We can set the input type to **url** to ensure that the input field is a url.

```
<input type="url" name="homepage" ng-model="user.url" />
```

# Angular Form Validation Using $error property

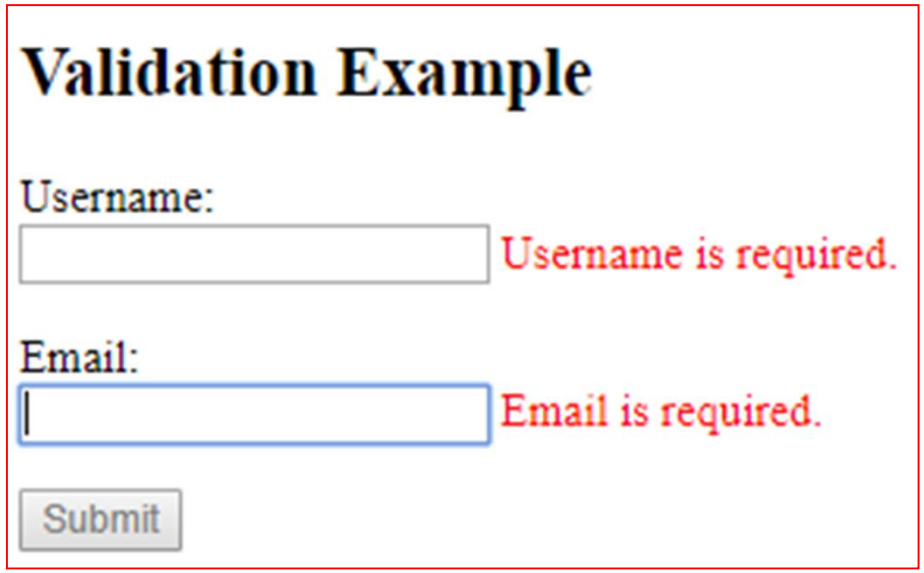
- **$error** , is an object hash, containing references to controls or forms with failing validators, where its keys are validation tokens (error names) and values are arrays of controls or forms that have a failing validator for given error name. The following list shows the Built-in validation tokens:
- email , max, maxlength, min,minlength,number
- pattern, required, url
- date, datetimelocal,time
- week,month

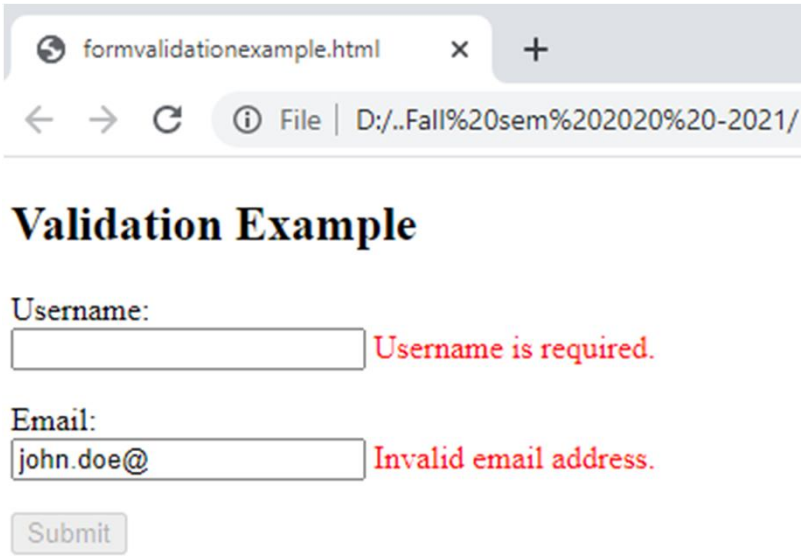
# Form Validation

```
<!DOCTYPE html>
<html>
<script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min
.js"></script>
<body ng-app="">
<p>Try writing in the input field:</p>
<form name="myForm">
<input name="myInput" ng-model="myInput" required>
</form>
<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
</body></html>
```

```
<!DOCTYPE html><html><script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
<body><h2>Validation Example</h2>
<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></body></html>
```


Validation Example
Username:
John Doe
Email:
john.doe@gmail.com
Submit


Validation Example
Username:
Username is required.
Email:
Email is required.
Submit


formvalidationexample.html
File | D:/..Fall%20sem%202020%20-2021/
Validation Example
Username:
Username is required.
Email:
john.doe@
Invalid email address.
Submit