



Session 5

Form Validation and AngularJS Animations



Session Overview

- Explain form validation with AngularJS
- Identify the different Form states and Input states
- Describe how to work with AngularJS Animations
- Identify the use of CSS classes for AngularJS Animations

Forms Validation

1-7

What are Forms?



- Enable validation of data in input controls
- Provide data-binding for elements

Input Controls

- HTML input elements

- Input elements
- Select elements
- Button elements
- Textarea elements

Forms Validation

2-7

Angular Forms

Offer client-side form validation

Monitor state of the form and input fields

Notify the user about current state

Hold information about whether they have been touched, or modified, or not

Use standard HTML5 attributes to validate input, or enable to make custom validation functions

Forms Validation

3-7

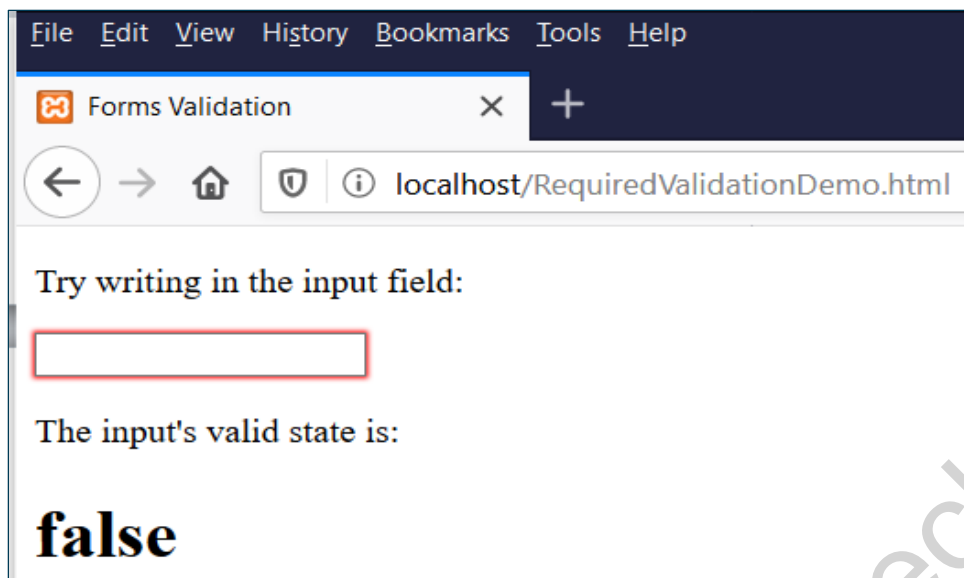
- **required**
 - Is an HTML5 attribute used to specify that input field must be filled out.

```
<!DOCTYPE html>
<html>
<title>Forms Validation</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js">
</script>
<body ng-app="">
<p>Try writing in the input field:</p>
<form name="myForm">
<input type="text" name="myInput" ng-model="myInput" required>
</form>
<p>The input's valid state is:</p>
<h1>{{myForm.myInput.$valid}}</h1>
</body>
</html>
```

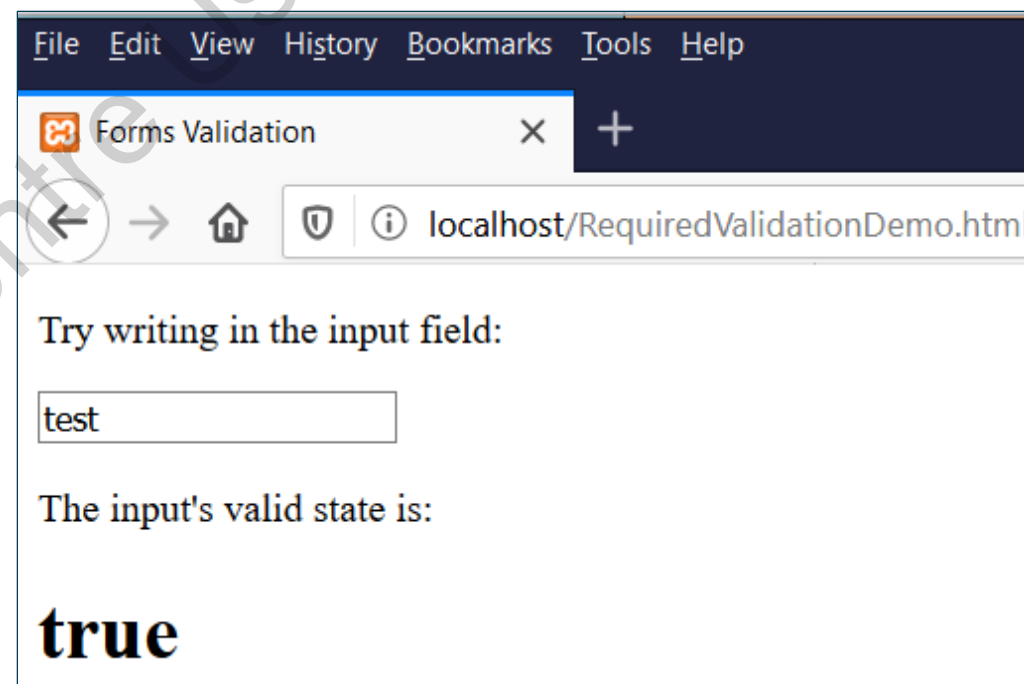
Required - Example Code

Forms Validation

4-7



Required - Example Initial Output



Required - Example after Adding Input

Forms Validation

5-7

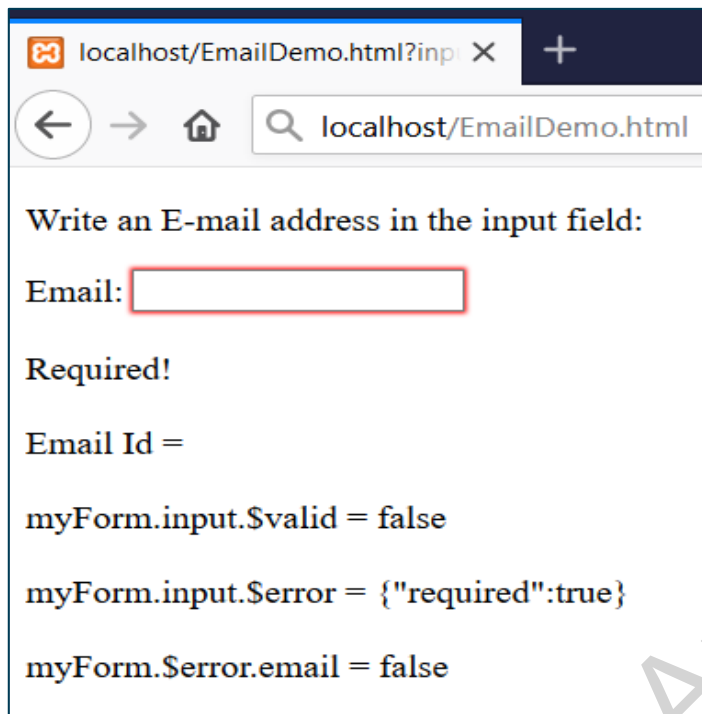
- **Email**

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js">
</script>
<body ng-app="">
<p>Write an E-mail address in the input field:</p>
<form name="myForm" >
  Email: <input type="email" name="input" ng-model="emailid" ngMinlength=5 required>
<br/>
<br/>
<span class="error" ng-show="myForm.input.$error.required">
  Required!</span><br/>
<span class="error" ng-show="myForm.input.$error.email">
  Not valid email!</span>
<br>
  Email Id = {{emailid}}<br/><br/>
  myForm.input.$valid = {{myForm.input.$valid}}<br/><br/>
  myForm.input.$error = {{myForm.input.$error}}<br/><br/>
  myForm.$error.email = {{!!myForm.$error.email}}<br/><br/>
</form>
</body>
</html>
```

Forms Validation

6-7

- Email



localhost/EmailDemo.html?inp X +

← → 🏠 🔍 localhost/EmailDemo.html

Write an E-mail address in the input field:

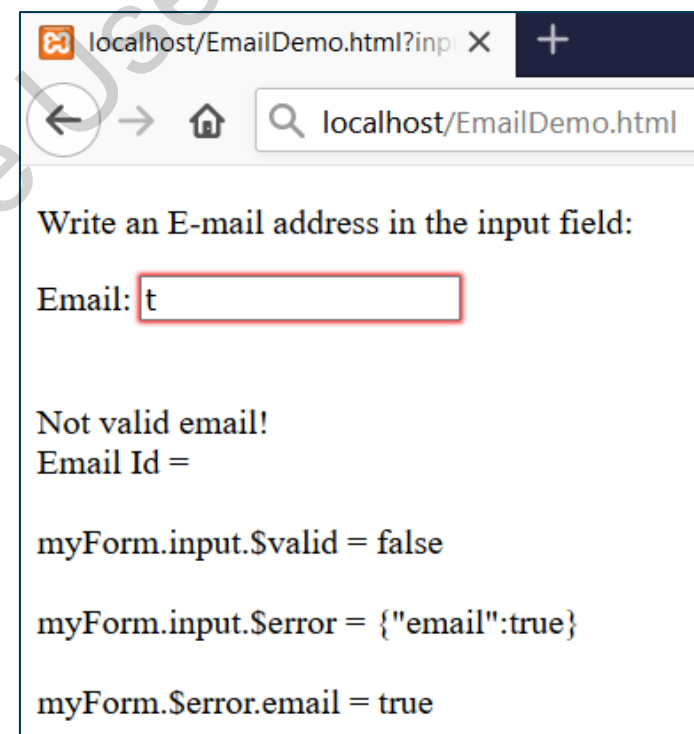
Email:

Required!

Email Id =

```
myForm.input.$valid = false
myForm.input.$error = {"required":true}
myForm.$error.email = false
```

Email- Example Initial Screen



localhost/EmailDemo.html?inp X +

← → 🏠 🔍 localhost/EmailDemo.html

Write an E-mail address in the input field:

Email:

Not valid email!

Email Id =

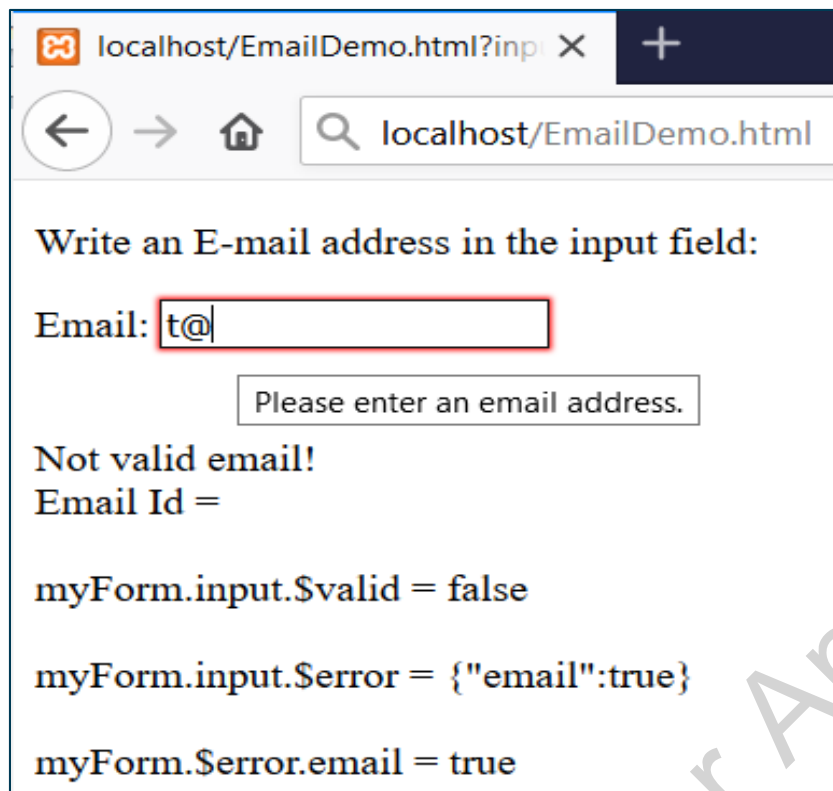
```
myForm.input.$valid = false
myForm.input.$error = {"email":true}
myForm.$error.email = true
```

Required - Example after Adding Input

Forms Validation

7-7

- **Email**



localhost/EmailDemo.html?inp X +

← → 🏠 🔍 localhost/EmailDemo.html

Write an E-mail address in the input field:

Email:

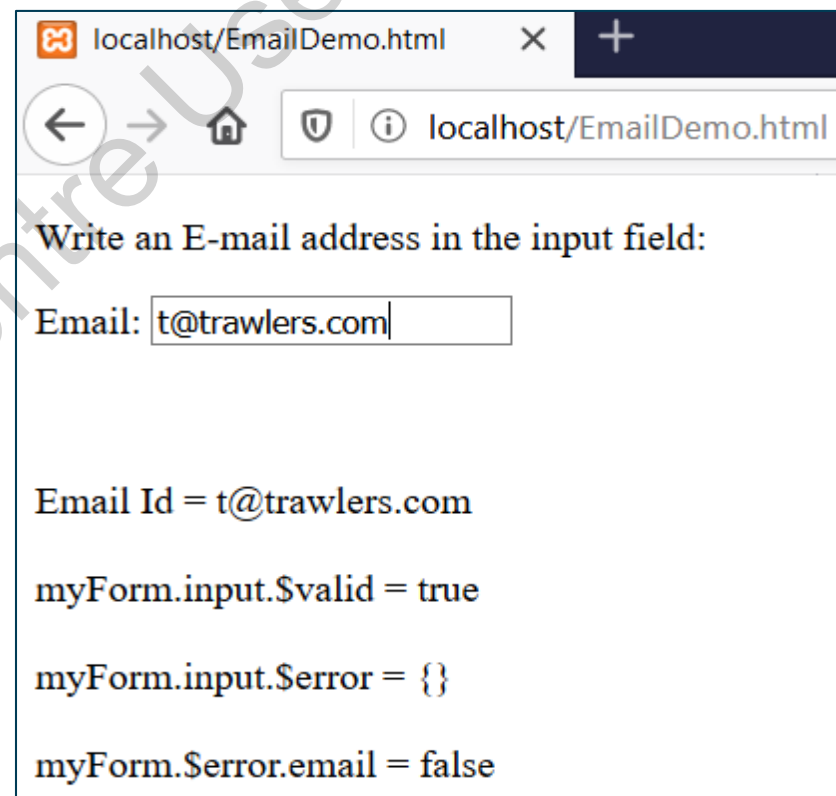
Please enter an email address.

Not valid email!

Email Id =

```
myForm.input.$valid = false
myForm.input.$error = {"email":true}
myForm.$error.email = true
```

Email - Example after @ input



localhost/EmailDemo.html X +

← → 🏠 ⓘ localhost/EmailDemo.html

Write an E-mail address in the input field:

Email:

Email Id = t@trawlers.com

```
myForm.input.$valid = true
myForm.input.$error = {}
myForm.$error.email = false
```

E-mail - Example After Validated Input

Form State and Input State

- AngularJS is constantly updating state of both form and input fields

Input fields have following states:

\$untouched	The field has not lost focus yet
\$touched	The field has been touched and focus is lost
\$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

Properties of input fields and are either **true** or **false**

Flow of Form States

Flow 1: <i>pristine and invalid</i>	When the form is first rendered and the user has not interacted with the form yet.
Flow 2: <i>dirty and invalid</i>	User has interacted with the form, but validity has not been satisfied yet.
Flow 3: <i>dirty and valid</i>	User has finished filling the form and the entire validation rule has been satisfied.

CSS Classes

CSS Class	Description
ng-valid	Is set if the input field is valid without errors
ng-invalid	Is set if the input does not pass validations
ng-pristine	Is set if a user has not interacted with the control yet
ng-dirty	Is set if the value of the form field has been changed
ng-touched	Is set if a user tabbed out from the input control
ng-untouched	Is set if a user has not tabbed out from the input control
ng-submitted	Is set if the form has been submitted

States and CSS Classes - Example

1-3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-">
<title>Forms Validation CSS class</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js"></script>
<style>
input.ng-pristine { background-color:yellow;}
input.ng-touched.ng-invalid { background-color:red;}
input.ng-touched.ng-valid {background-color:green;}
</style>
</head>
<body ng-app>
<form name="studentForm" novalidate class="student-form">
<label for="firstName">First Name: </label><br />
<input type="text" name="firstName" ng-model="firstName" ng-required ="true" />
<br /><br />
<label for="lastName">Last Name</label><br />
<input type="text" name="lastName" ng-model="lastName" ng-required="true" />
```

States and CSS Classes - Example

2-3

```
<br /><br />
<label for="dob">E-mail</label><br />
<input type="email" id="email" ng-model="email" name="email" ng-required =
"true"/>
<br /><br />
<input type="submit" value="Submit" /><br />
<p>Initially the input fields are yellow They become green when we give
valid data in it. They turn red if data is invalid, say an empty string
</p>
</form>
</body>
</html>
```

States and CSS Classes - Example

3-3

- CSS Classes



Forms Validation CSS cla x

localhost/angular/formsvalidati

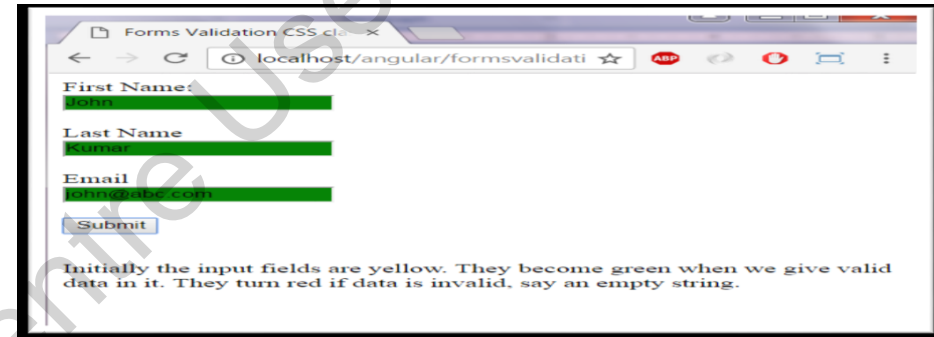
First Name:

Last Name

Email

Initially the input fields are yellow. They become green when we give valid data in it. They turn red if data is invalid, say an empty string.

States and CSS Classes - Initial Screen



Forms Validation CSS cla x

localhost/angular/formsvalidati

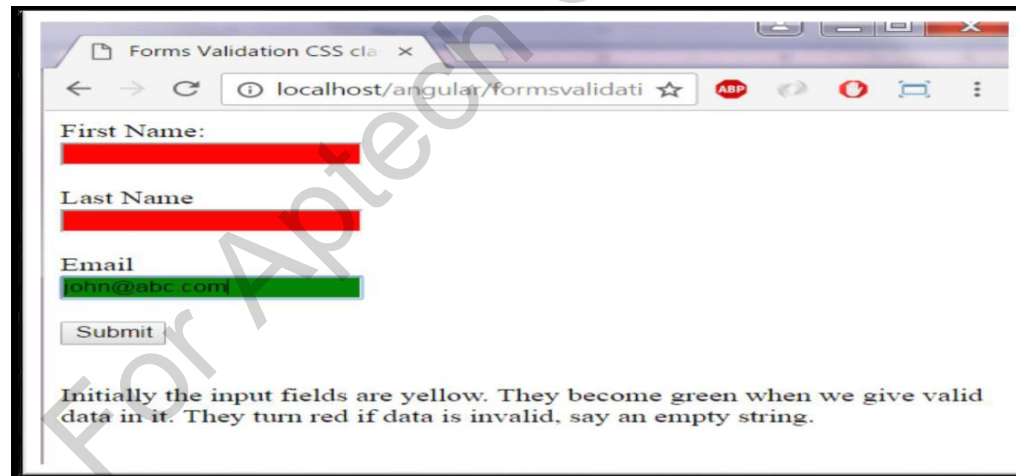
First Name:

Last Name

Email

Initially the input fields are yellow. They become green when we give valid data in it. They turn red if data is invalid, say an empty string.

States and CSS Classes - with Valid Inputs



Forms Validation CSS cla x

localhost/angular/formsvalidati

First Name:

Last Name

Email

Initially the input fields are yellow. They become green when we give valid data in it. They turn red if data is invalid, say an empty string.

States and CSS Classes - with Invalid Inputs

AngularJS Animations

1-3

- An animation is the transformation of an HTML element which gives an illusion of motion.
- Animating elements in our app or pages adds to the fun and increases user experience.
- Animations enhance user interface by making it smooth and more attractive.



AngularJS Animations

2-3

What do You Need?

- To make your applications ready for animations, you must include the AngularJS Animate library:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular-animate.js">
</script>
```

- Then, you must refer to the `ngAnimate` module in your application:

```
<body ng-app="ngAnimate">
```

AngularJS Animations

3-3

- If your application has a name, add **ngAnimate** as a dependency.

Adding
Dependency

```
<body ng-app="myApp">

<h1>Hide the DIV: <input type="checkbox" ng-
model="myCheck"></h1>

<div ng-hide="myCheck"></div>

<script>

var app = angular.module('myApp',
['ngAnimate']);

</script>
```

AngularJS Animations - Example

1-3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>AngularJS Animations</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js">
</script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular-
animate.js">
</script>
<style>
    div {
    transition: all linear 1s;
    background-color: cyan;
    height: 100px;
    width: 100%;
    position: relative;
    top: 0;
    left: 0;
    }
```

AngularJS Animations - Example

2-3

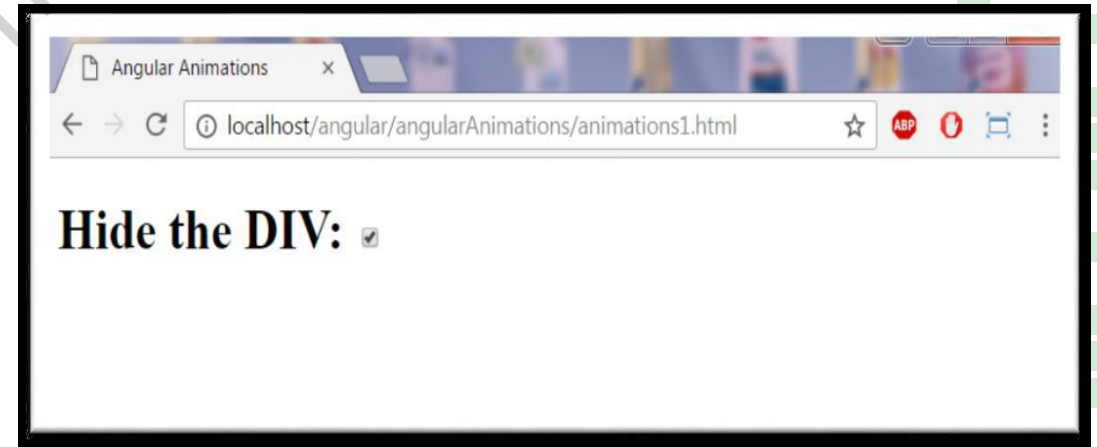
```
.ng-hide {  
  height: 0;  
  width: 0;  
  background-color: transparent;  
  top: -200px;  
  left: 200px;  
}  
  
</style>  
</head>  
<body ng-app="ngAnimate" >  
<h1>Hide the DIV: <input type="checkbox" ng-model="myCheck"></h1>  
<div ng-hide="myCheck"></div>  
</body>  
</html>
```

AngularJS Animations - Example

3-3



ngAnimate Example - Initial Screen



ngAnimate Example - After checking and completion of animation

Purpose of ngAnimate

- Adds and removes classes
- It does not animate your HTML elements by itself

Directives in AngularJS which are animate-aware and can add/remove classes:

- ng-show
- ng-hide
- ng-class
- ng-view
- ng-include
- ng-repeat
- ng-if
- ng-switch

Example: **ng-hide** directive will add these class values:

- ng-animate
- ng-hide-animate
- ng-hide-add (if the element will be hidden)
- ng-hide-remove (if the element will be showed)
- ng-hide-add-active (if the element will be hidden)
- ng-hide-remove-active (if the element will be showed)

CSS Animations

1-3

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>AngularJS Animations</title>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular.min.js">
</script>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.7.9/angular-animate.js">
</script>
<style>
    div { transition: all linear 1s;
        background-color: cyan;
        height: 100px;
    }
    .ng-hide {
        height: 0px;
    }
</style>
</head>
```

CSS Animations

2-3

```
<body ng-app="myApp">
  <h1>Animate the DIV: <input type="checkbox" ng-model="myCheck"></h1>
  <div ng-hide="myCheck"></div>
  <script>

var app = angular.module('myApp', ['ngAnimate']);
  </script>
</body>
</html>
```


CSS Animations

3-3



CSS Animations - Initial Screen



CSS Animations - Final Screen



Summary

- Forms are the major way users communicate with applications we develop.
- First-line-of-defense validation is typically performed in the Web browser.
- Form validations reduce the load on our Web servers, conserve bandwidth, and provide better user experience.
- AngularJS monitors the state of the form and input fields and lets us notify the user about the current state.
- Animating the elements in our application adds to the fun and increases the user experience.
- The angular-animate.js library must be added in addition to the core angular.js library, to implement animations in AngularJS.
- The ngAnimate module adds and removes classes.