# Curso de Angular 6

## Unidad Didáctica 06: Formularios

http://cursosdedesarrollo.com/

Ayuntamiento
de Vitoria-Gasteiz
Vitoria-Gasteizko
Udala

# Índice de contenidos



http://cursosdedesarrollo.com/

# Introducción

Los formularios son uno de los principales problemas a resolver en cualquier framework que tenga interacciones por parte del usuario

http://cursosdedesarrollo.com/

# Introducción

Para poder ver de una manera más bonita los formularios vamos a manejar bootstrap como sistema de maquetación

http://cursosdedesarrollo.com/

# Introducción

Para ello añadiremos en el index.html el siguiente código dentro del <head>

```
<link rel="stylesheet"
      href="https://unpkg.com/bootstrap@3.3.7/dist/css/
bootstrap.min.css">
```

# Introducción

Dentro de los formularios hay varias cuestiones que debemos tener en cuenta como son: data-binding, validaciones y el manejo de errores

http://cursosdedesarrollo.com/

# Introducción

Siempre que queramos usar el módulo de formularios tendremos que incluirlo dentro del módulo de la aplicación en el app.module.ts

```
import { FormsModule }   from '@angular/forms';

@NgModule({
  imports: [
    BrowserModule,
    FormsModule
  ],
```

http://cursosdedesarrollo.com/

# Introducción

La idea de esta unidad será hacer un formulario básico

http://cursosdedesarrollo.com/

# Introducción

## Hero Form

**Name**

Dr IQ

**Alter Ego**

Chuck Overstreet

**Hero Power**

Really Smart

Submit

http://cursosdedesarrollo.com/

# Introducción

Dos de los campos será obligatorios. Verde será válido, rojo inválido

http://cursosdedesarrollo.com/

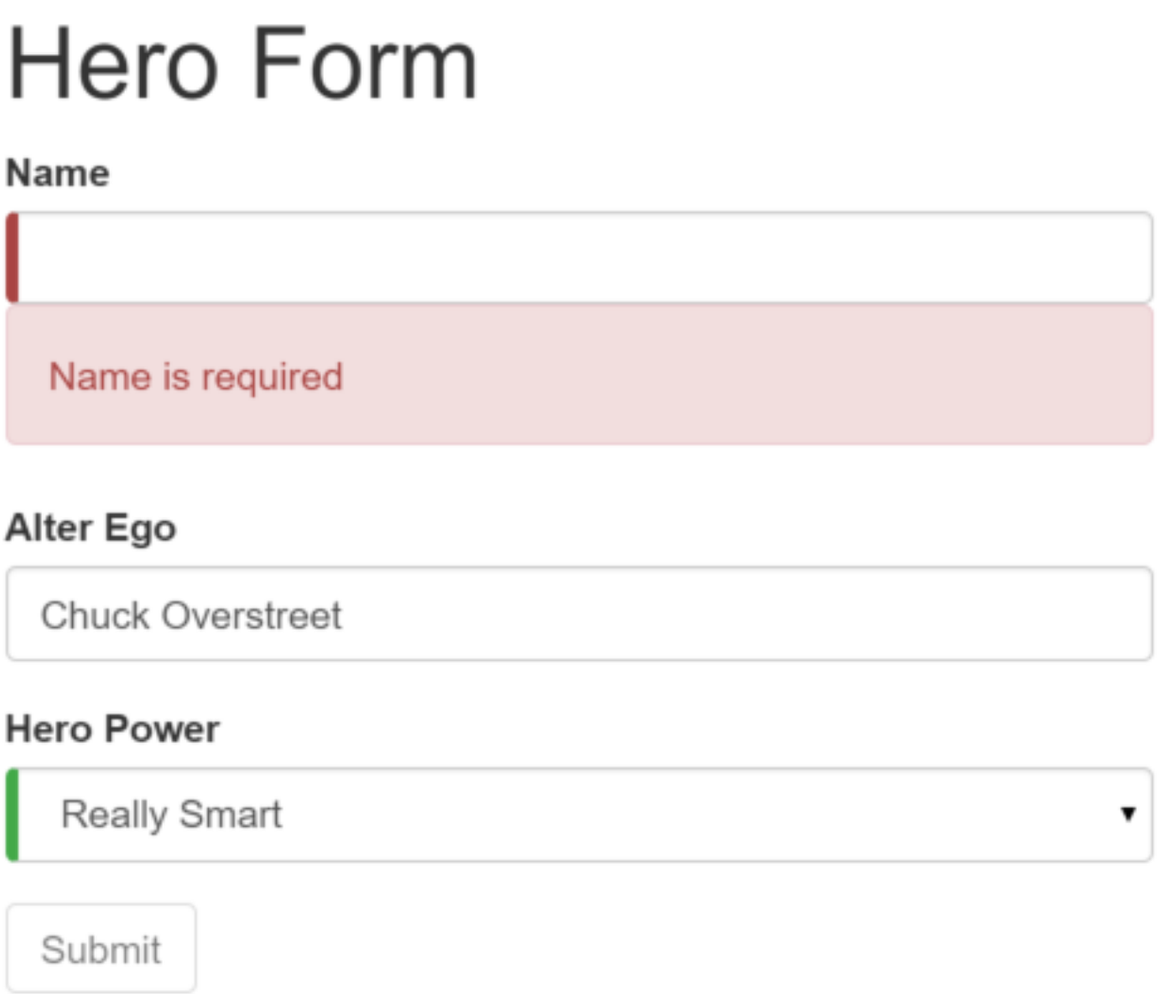
# Introducción

Los pasos que daremos serán:

- Crear el modelo, incluida la clase
- Crear el componente del formulario
    - Asocia una variable modelo con el componente
    - Crea el formulario en la plantilla
    - Asocia los campos con el modelo
    - Muestra los fallos y los mensajes de error
    - Maneja el botón submit y deshabilita lo si no valida

http://cursosdedesarrollo.com/

# Introducción




http://cursosdedesarrollo.com/

# Crear el modelo

Siempre que manejemos un formulario será necesario manejar los datos del formulario y la manera más práctica será crear una clase que disponga de esos datos



# Crear el modelo

Ejecutaremos

ng g class Hero

http://cursosdedesarrollo.com/

# Crear el modelo

Luego editaremos el fichero para que contenga los campos que queremos manejar

http://cursosdedesarrollo.com/

# Crear el modelo

```
export class Hero {

  constructor(

    public id: number,

    public name: string,

    public power: string,

    public alterEgo?: string

  ) {  }

}
```

http://cursosdedesarrollo.com/

# Asociando al Componente

Tendrás que importar la clase desde el componente para poder usarla

```
import { Hero }    from '../hero';
```

http://cursosdedesarrollo.com/

# Asociando al Componente

Luego tendrás que declarar un objeto modelo dentro de la clase del componente donde guardarás los datos el formulario

```
powers = ['Really Smart', 'Super Flexible’,

            'Super Hot', 'Weather Changer'];

model = new Hero(18, 'Dr IQ', this.powers[0], 'Chuck Overstreet');
```



# Asociando al Componente

En este caso el formulario estará rellenado por defecto con unos datos del objeto, como en cualquier formulario de edición



# Asociando al Componente

Si es un formulario de alta (debe estar vacío) se inicializa el objeto con todos los campos a 0 (cadenas vacías etc…)

http://cursosdedesarrollo.com/

# Creando el formulario

Ahora deberemos modificar para que la plantilla contenga el formulario con los datos del objeto modelo



# Creando el formulario

```
<form (ngSubmit)="onSubmit()" #heroForm="ngForm">
  <div class="form-group">
    <label for="name">Name</label>
    <input type="text" class="form-control" id="name"
           required
           [(ngModel)]="model.name" name="name"
           #name="ngModel">
    <div [hidden]="name.valid || name.pristine"
         class="alert alert-danger">
      Name is required
    </div>
  </div>
```

http://cursosdedesarrollo.com/

# Asociando los Modelos

Para ellos utilizaremos la directiva ngModel que nos permitirá asociar cada campo a una propiedad del objeto



# Asociando los Modelos

En el formulario destacamos dos elementos principalmente

el [(ngModel)] que relaciona con el objeto del componente

y el (ngSubmit) que se relaciona con el evento para enviar el formulario

http://cursosdedesarrollo.com/

# Control de Validaciones

Para la validación se utilizan los nombres del campo seguidos de las propiedades de validación: pristine, valid, invalid, dirty, etc…

http://cursosdedesarrollo.com/

# Control de Validaciones

Cabe destacar también [hidden] que ocultará el elemento html si cumple la condición asociado al criterio de validación

http://cursosdedesarrollo.com/

# Control de Validaciones

También se tendrá en cuenta el required para decir que el campo es requerido a la hora de validar

http://cursosdedesarrollo.com/

# Rellenando Selects

```
<div class="form-group">
  <label for="alterEgo">Alter Ego</label>
  <input type="text" class="form-control" id="alterEgo"
         [(ngModel)]="model.alterEgo" name="alterEgo">
</div>
<div class="form-group">
  <label for="power">Hero Power</label>
  <select class="form-control" id="power"
          required
          [(ngModel)]="model.power" name="power"
          #power="ngModel">
    <option *ngFor="let pow of powers" [value]="pow">{{pow}}</
option>
  </select>
  <div [hidden]="power.valid || power.pristine" class="alert alert-
danger">
    Power is required
  </div>
</div>
```

http://cursosdedesarrollo.com/

# Rellenando Selects

La asignación del [value] permite gestionar mejor la generación de la select

http://cursosdedesarrollo.com/

# El evento ngSubmit

```
<button type="submit" class="btn btn-success" [disabled]="!
heroForm.form.valid">Submit</button>
  <button type="button" class="btn btn-
default" (click)="newHero(); heroForm.reset()">New Hero</
button>
</form>
```

http://cursosdedesarrollo.com/

# El evento ngSubmit

El [disabled] permite verificar que realmente el formulario valida para evitar falsos envíos del formulario

http://cursosdedesarrollo.com/

# El evento ngSubmit

```
<div [hidden]="!submitted">
  <h2>You submitted the following:</h2>
  <div class="row">
    <div class="col-xs-3">Name</div>
    <div class="col-xs-9  pull-left">{{ model.name }}</div>
  </div>
  <div class="row">
    <div class="col-xs-3">Alter Ego</div>
    <div class="col-xs-9 pull-left">{{ model.alterEgo }}</div>
  </div>
  <div class="row">
    <div class="col-xs-3">Power</div>
    <div class="col-xs-9 pull-left">{{ model.power }}</div>
  </div>
  <br>
  <button class="btn btn-primary" (click)="submitted=false"
>Edit</button>
</div>
```

http://cursosdedesarrollo.com/

# El evento ngSubmit

Presentamos los datos enviados como manera de verificar que ha ido todo correcto

http://cursosdedesarrollo.com/

# El evento ngSubmit

```
<div class="container">
  <ul>
    <li *ngFor="let heroe of heroes" >
      {{heroe.name}}
    </li>
  </ul>
</div>
```

http://cursosdedesarrollo.com/

# El evento ngSubmit

Presentamos el listado de objetos para verificación

http://cursosdedesarrollo.com/

# El evento ngSubmit

Esta forma de validar los formularios es la más básica, también disponemos de una forma más novedosa que es la ReactiveFormsModule



# Referencias

Documentación de Forms:

https://angular.io/docs/ts/latest/guide/forms.html

Validación de Forms:

https://angular.io/docs/ts/latest/cookbook/form-validation.html

http://cursosdedesarrollo.com/

# Conclusiones

Hemos visto cómo gestionar los formularios sus validaciones y envíos de Angular

http://cursosdedesarrollo.com/

# Datos de Contacto

http://www.cursosdedesarrollo.com
david@cursosdedesarrollo.com

http://cursosdedesarrollo.com/

# Licencia

David Vaquero Santiago



http://cursosdedesarrollo.com/