Angular 2 Forms

Deborah Kurata

Consultant | Developer | Mentor

deborahk@insteptech.com

Deborah Kurata

- Independent Consultant | Developer | Mentor
  - Web (Angular), .NET
- Pluralsight Author
  - AngularJS Line of Business Applications
  - Angular Front to Back with Web API
  - Object-Oriented Programming Fundamentals in C#
- Microsoft MVP

Session Materials & Sample Code

https://github.com/DeborahK/AngularU2015-Angular2Forms

Angular 2 Form Techniques

Template Driven	Imperative (Model) Driven	Data Driven
Similar to 1.x	Built w/Code	Automatic

Template-Driven
Similar to 1.x

Controller

Template Driven

TypeScript class

Bindable Controller Properties

Controller Methods

```
class CheckOutCtrl {
  model = new CheckoutModel();
  countries = ['US', 'Canada'];

  onSubmit() {
    console.log("Submitting:");
    console.log(this.model);
  }
}
```

Model

Template Driven

TypeScript class

Strongly Typed Properties

```
class CheckoutModel {
  firstName: string;
  middleName: string;
  lastName: string;
  country: string = "Canada";

  creditCard: string;
  amount: number;
  email: string;
  comments: string;
}
```

View

Template Driven

Event Binding

Local Variable

Two-way Binding

Property Binding

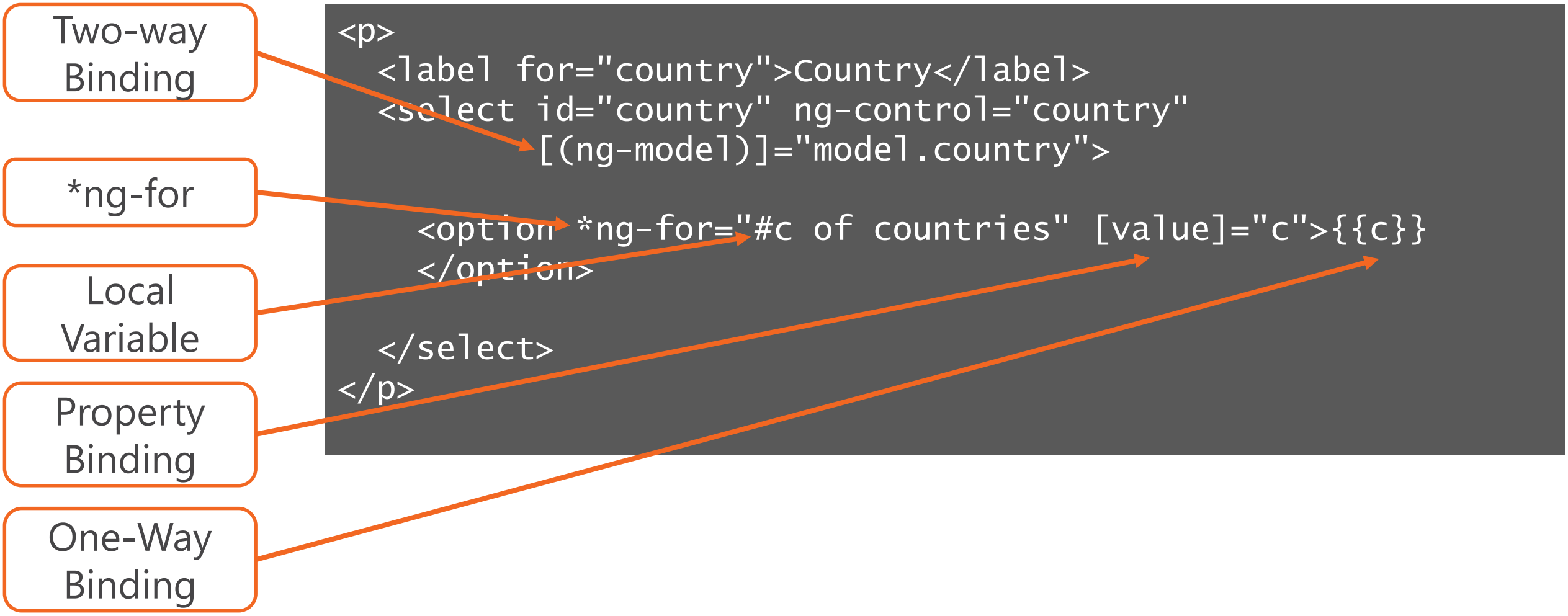
```
<h1>Checkout Form</h1>

<form (ng-submit)="onSubmit()" #f="form">
 <p>
  <label for="firstName">First Name</label>
  <input type="text" id="firstName" ng-control="firstName"
         [(ng-model)]="model.firstName" required>
  <show-error control="firstName"
              [errors]="['required']"></show-error>
 </p>
. . .

  <button type="submit" [disabled]="!f.form.valid">
       Submit
  </button>
</form>
```

ng-repeat -> *ng-for

Template Driven



DEMO:
TEMPLATE-BASED FORMS

Imperative (or Model) Driven
Build Form with Code

Controller

Model Driven

TypeScript class

Bindable Controller Properties

Building the Form/Model

Controller Methods

```
class CheckOutCtrl {
  formModel;
  countries = ['US', 'Canada'];

  constructor(fb: FormBuilder) {
    this.formModel = fb.group({
     "firstName": ["", Validators.required],
     "country": ["Canada", Validators.required],
     "creditCard": ["", Validators.compose([Validators.required,
                                            creditCardValidator])]

     });
  }
  onSubmit() {
    console.log("Submitting:");
    console.log(this.form.value);
  }
}
```

View

Model Driven

Event Binding

Property Binding

Control Mapping

Property Binding

```
<h1>Checkout Form</h1>

 <form (ng-submit)="onSubmit()"
       [ng-form-model]="formModel">
  <p>
   <label for="firstName">First Name</label>
   <input type="text" id="firstName" ng-control="firstName">
   <show-error control="firstName"
               [errors]="['required']"></show-error>
  </p>
. . .

   <button type="submit" [disabled]="!formModel.valid">
        Submit
   </button>
</form>
```

Key Differences

Template-Driven

- Controller exposes **data** model

```
class CheckOutCtrl {
  model = new CheckoutModel();
  countries = ['US', 'Canada'];
...
}
```

Model-Driven

- Controller exposes **form** model

```
class CheckOutCtrl {
  formModel;
  countries = ['US', 'Canada'];
...
}
```

Key Differences

Template-Driven

- Controller exposes data model
- Binding & Validation in **View**

```
<input
 type="text"
 id="firstName"
 ng-control="firstName"
 [(ng-model)]="model.firstName"
 required>
```

Model-Driven

- Controller exposes form model
- Binding & Validation in **Controller**

```
class CheckOutCtrl {
 formModel;
 countries = ['US', 'Canada'];

 constructor(fb: FormBuilder) {
  this.formModel = fb.group({
  "firstName": ["", Validators.required],
  "LastName": ["", Validators.required],
  . . .
      });
  }
...
}
```

Key Differences

Template-Driven

- Controller exposes data model
- Binding & Validation in View
- View contains **data bindings**

```
<input
  type="text"
  id="firstName"
  ng-control="firstName"
  [(ng-model)]="model.firstName"
  required>
```

Model-Driven

- Controller exposes form model
- Binding & Validation in Controller
- View contains **control mappings**

```
<input
  type="text"
  id="firstName"
  ng-control="firstName">
```

Benefits to Model-Driven

- Behavior is in the code, not the template
  - Binding, validation
- Easier to reason against
- More readily unit tested
- Allows for future scenarios
  - Data-driven forms

Take Away

Template-Driven: Similar to Angular 1.x

Model-Driven: Build Forms in Code

Data-Driven or Other Future Approaches: Possible

Thank You!

- @deborahkurata
- deborahk@insteptech.com
- http://blogs.msmvps.com/deborahk
- **https://github.com/DeborahK/AngularU2015-Angular2Forms**