Component Inputs

- You can pass data to a component.
- You can either use the `inputs` array on a component or annotate an instance variable with the `Input` decorator
- Once you specify the inputs to your component, they become available in the `ngOnInit` method
- You can implement the `ngOnInit` and access the input instance variables
- You can use the `[propname]="data"` to set the `propname` to whatever `data` evaluates to
- Note that if you set `[propname]="'data'"`, `propname` will be set to the literal `data` string

Project files

The project files for this section are in angular2-intro/project-files/angular-examples/component-input.

Getting Started

In order to demonstrate component inputs, we are going to create a `user` component and pass `name`, `lastName`, and `userId` to it. So our final html tag would look something like the following:

```
1 <user name="Tom" lastName="Johnson" uesrId="1"></user>
```

And the template for the component will be:

```
1 <h1>Hello, {{ name }} {{ lastName }}, id: {{ userId }}</h1>
```

which would output: `Hello, Tom Johnson id: 1`.

To get started, let's define the `User` component:

```
1  @Component({
2    selector: 'user',
3    template: '<h1>Hello, {{ name }} {{ lastName }} id: {{ userId }}</h1>',
4    inputs: ['name', 'lastName', 'userId'] // <- specifying the inputs to the `U
5  })
6  class User {}
```

- On line 4 we are defining the inputs as an array of strings

Then, we are going to use the `User` component inside our app's template:

```
1  @Component({
2    selector: 'app',
3    template: `<user name="Tom" lastName="Johnson" uesrId="1"></user>`
4  })
5  class Root {}
```

because we are using the `User` component in the app, we need to register it with the app by adding `User` class to the list of `directives` of the app component:

```
1  @Component({
2    selector: 'app',
3    template: `<user name="Tom" lastName="Johnson" userId="1"></user>`,
4    directives: [User] // <- register the component
5  })
6  class Root {}
```

and at the end we need to bootstrap the app:

```
1  bootstrap(Root, [])
```

Now, notice that instead of adding the inputs to the `inputs` array, we could have decorated the instance variables with the `@Input` decorator:

```
import {Input} from 'angular2/core'; // <- importing the Input decorator
@Component({
  selector: 'user',
  template: '<h1>Hello, {{ name }} {{ lastName }} id: {{ userId }}</h1>'
  // <- removing the inputs array.
})
class User {
  @Input() private name: string;
  @Input() private lastName: string;
  @Input() private userId: number;
}
```

Binding Data to Properties

Now, let's see how we can bind to a property from another component. For this example, we are going to continue with our `User` component and create a new component called `Permission`. Then we are going to use the the `Permission` component inside the `User` component and set the `uid` of `Permission` by the `userId` of the `User`.

The `Permission` component is defined as follows:

```
@Component({
  selector: 'permission',
  template: '<h2> Restriction is: {{ restriction }}'
})
class Permission {
  @Input() private uid: string;
  private restriction: string;
  constructor() {
    this.restriction = 'none';
  }
  ngOnInit() {
    this.restriction = this.uid === '1' ? 'admin' : 'normal';
  }
}
```

- On line 6 we are defining `uid` to be an input instance variable. It's value is set from outside.
- In the constructor we are setting a default value for the restriction.
- Then in the `ngOnInit` hook, we are evaluating the value of `restriction` based on the given id provided by other components, in this case the `User` component
- In this silly example, if the passed id is `1`, we will set the `restriction` to `admin`, otherwise we set it to `normal`.

then we are going to register the `Permission` component with the `User` component so that we can use it in the `User` template:

```
1  @Component({
2    selector: 'user',
3    ///...
4    directives: [Permission] // <-
5  })
6  class User {}
```

then we can update the `User` template to include the `Permission`:

```
1  @Component({
2    selector: 'user',
3    template: `
4    <h1>Hello, {{ name }} {{ lastName }}, id: {{ userId}}</h1>
5    <div>
6      <permission [uid]="userId"></permission>
7    </div>
8    `,
9    inputs: ['name', 'lastName', 'userId'],
10   directives: [Permission]
11 })
12 class User {}
```

- Note that on line 6 we are setting the `uid` of `Permission` by `userId` available from the `User` component.

If you run the app you should see the following printed to the page:

Input to components

Binding to DOM Properties

- `[style.color]="done ? 'green' : 'red' "`
- `[class.name]="done ? 'done' : 'pending'"`

TODO