Seneca
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

BTI425

Web Programming for Apps and Services

Notes	Weekly
Resources	Graded work
Policies	Standards
Professors & addendum	Code examples

Angular component interaction

In React, you learned about *contained components*, in the `react-tania-updated` code example.

Specifically, you learned how to pass a value from a parent component (`App.js`) to a child component (`Table.js`). In the child component, the passed-in value was available in the `props` property.

In this note, you will learn how it's done in Angular. Open the `component-interaction` code example as you work your way through this content.

Containment

The app was generated in the standard way. Then, four components were created:

1. parent
2. child-one
3. child-two
4. child-three

In the `app-component.html` template code, the "parent" element was added:

```
<div>
  <h2>Angular component interaction</h2>
  <p>Shows containment and passing data</p>
</div>
<hr>
<app-parent></app-parent>

<router-outlet></router-outlet>
```

Then, in the `parent-component.html` template code, the elements for each “child” were added. To make it look nice, they were contained in a Bootstrap row-and-panel structure:

Pass data from parent to child

In the code example, it was decided that the top-level app component would initialize some data properties, and pass the data on to the “parent”, which in turn will pass data on to each “child”.

Setup

Here’s the setup:

First declare some data properties in the top-level app component class:

```
courseName: string;
enrolledStudentCount: number;
// Professor is a class that has "name" and "age" properties
professors: Professor[];
```

Next, in its constructor, set their values:

```
this.courseName = "BTI425 - Web Programming for Apps and Services";
```

```
this.enrolledStudentCount = 90;
this.professors = [
  { name: "Peter McIntyre", age: 45 },
  { name: "Tanvir Alam", age: 35 }
];
```

Let's assume that we want to pass on the `courseName` value. Open the app component's template for editing. Edit the markup for the `app-parent` element.

```
<app-parent [courseNameInParent]="courseName"></app-parent>
```

The attribute name - in brackets - is the name of the *property* in the target or destination component class.

The attribute value is the name of the property in this source ("app") component class.

Now, open the destination component class (`parent-component.ts`) for editing. Add the "Input" symbol to the top-most import statement:

```
import { Component, OnInit, Input } from '@angular/core';
```

Declare a data property to hold the passed-in value. Prefix it with an `@Input()` decorator:

```
@Input()
courseNameInParent: string;
```

Finally, open the template code, and render the result:

```
<p>{{ courseNameInParent }}</p>
```

In summary, to pass data from one component to another:

Step	Task	Where done
1	Declare and initialize a data property	Parent class
2	Add `[targetPropertyName]="sourceValue"` to the markup	Parent template
3	Declare a decorated data property	Child class
4	Render `targetPropertyName` in the markup	Child template

Do more

Above, we declared and initialized other data properties in the top-level app component. Now it's time to pass them on.

Edit the `app.component.html` markup, to pass on the other two properties.

In the `parent.component.html` markup, pass on the enrolled students number to component #1, and pass on the professor info to component #2.

If all goes well, this shows the result. The yellow-background content is the passed-in data.

Learn more

There's much more to this topic, but you now have enough knowledge to use it.

Summary of interaction patterns

Some of this is nicely covered in the Angular docs section titled Binding syntax: an overview.

For the following, assume that the component has a property named `info` and a method named `getInfo()` etc. For example:

```
info: string = "Peter";
photo: string = "https://example.com/people/123";

getInfo(): string {
  return "Peter";
```

```
}

doSomething() {
  // Some task (no return value)
}
```

Here's a brief summary of data display, usage, and interaction patterns.

Display data in an element

Interpolation:

```
<h3>Customer name is {{ info }}</h3>
<!-- or... -->
<h3>Customer name is {{ getInfo() }}</h3>
```

Use data in an element

Property binding:

```
<img [src]="photo" alt="Customer photo">
```

Pass data to a component

```
<app-child [infoInChild]="info"></app-child>
```

Handle an event in an element

Template statement:

```
<div (mouseover)="doSomething()">
<!-- or... -->
<div (click)="doSomething()">
<!-- or... -->
<div (click)="doSomething($event)">
```

Two-way binding in a form

Two-way binding:

```
<input [(ngModel)]="info" required placeholder="Customer name">
```