Component Output/Events

- Events can be emitted from components. These events can be either custom or they could be DOM events
- The syntax is `(eventname)="fn()"` where `eventname` is the name of the event and `fn` is the handler function
- The handler function is called when the event is fired
- For example, if you want to handle a click event you can do: `(click)="handler()"`. In this case the `hander` is called whenever the click event is fired off
- You can use Angular's `EventEmitter` to fire off custom events

Project Files

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

Final Result

The goal of this section is to show you how to create a component that contains a button that when is clicked, calls a handler defined by the component's class. The final html will look like the following:

```
<p>Value: {{ value }}</p>
<button (click)=addOne()>Add +</button>
```

That idea is very simple: every time we click on the button we want to increment the value by one. In addition to that, we want to be able to hook into a custom event and run the `addOne` method whenever the event is fired:

```
<p>Value: {{ value }}</p>
<span adder-auto (myevent)=addOne()>adding ...</span>
```

Getting Started

Let's get started by defining our `Adder` component:

```
@Component({
  selector: 'adder',
  template:`
  <p>Value: {{ value }}</p>
  <button (click)=addOne()>Add +</button>
  `
})
class Adder {
  private value: number;
  constructor() {
    this.value = 0;
  }
  addOne() {
  this.value += 1;
  console.log(this.value);
  }
}
```

Now, we are just going to register `Adder` with our root component:

```
@Component({
  selector: 'app',
  directives: [Adder],
  template: '<adder></adder>'
})
class App {}
```

after you bootstrap the app and run it you should be able to see a button that when clicked increments the value by one.

Using EventEmitter

Now, let's see how we can use the `EventEmitter` to increment the value by one every time a custom event is fired every second. In order to achieve that, we are going to create an

attribute directive called `AdderAuto`. Start by importing the `Directive` metadata class:

```
import {Directive} from 'angular2/core';
```

and then define the selector for the directive:

```
@Directive({
  selector: '[adder-auto]'
})
```

- `selector: '[adder-auto]'` means that angular will target any element that has the `adder-auto` attribute and will create an instance of the class. Now we need to define the class for our directive:

```
class AdderAuto {
  // custom event definition
}
```

In this class we need to define a custom event output hook. We are going to call it `myevent`. The same way that you can hook into `(click)`, we want to be able to use `(myevent)`. To achieve that, we need to create an instance variable and decorate it with the `Output` decorator:

```
// -> importing `EventEmitter` and `Output` decorator.
import {EventEmitter, Output} from 'angular2/core';
class AdderAuto {
  @Output() myevent: EventEmitter<string>;
  constructor() {
    this.myevent = new EventEmitter();
  }
}
```

- If you notice, `myevent` is of type `EventEmitter` that emit events of type string

- In the constructor we are creating an instance of `EventEmitter`. So now we can use `myevent` to emit events
- We can use `setInterval` to emit event from our custom event every second

```
class AdderAuto {
  @Output() myevent: EventEmitter<string>;
  constructor() {
    this.myevent = new EventEmitter();
    setInterval(()=> {this.myevent.emit('myevename')}, 1000);
  }
}
```

Now we can register `AdderAuto` with the `Adder` component and run the `addOne` method every second:

```
@Component({
  selector: 'adder',
  ...
  directives: [AdderAuto] // <- register `AdderAuto`
})
```

and then we can update the template:

```
<p>Value: {{ value }}</p>
<button (click)="addOne()">Add +</button>
<!-- using the event. -->
<h2>Using Emitter</h2>
<span adder-auto (myevent)="addOne($event)"> EVENT: </span>
```

- first we are adding the attribute directive `adder-auto` on the span
- second, we are using the `myevent` hook and attaching `addOne` handler to it. This means that whenever the `myevent` event is triggered, run the `addOne` handler.

The `Adder` component now looks like the following with the updated template:

```
@Component({
  selector: 'adder',
  template:`
  <p>Value: {{ value }}</p>
  <button (click)="addOne()">Add +</button>
  <h2>Using Emitter</h2>
  <span adder-auto (myevent)="addOne($event)"> EVENT: </span>
  `,
  directives: [AdderAuto]
})
```

Now if you run the code, you should be able to see the number incrementing by one every second.