Angular 6 – Use Angular Elements to build Custom Elements example

Custom Element is a web standard with new HTML element which allows us to define a tag whose content is created and controlled by JavaScript code in a framework-agnostic way. Custom Element bootstraps itself. That means it starts automatically when added to the DOM, and is automatically destroyed when removed from the DOM. It looks and behaves like any HTML element without need to know anything about Angular, component’s data structures or implementation. This feature is currently supported by Chrome, Opera, and Safari, and available in other browsers through polyfills. In the tutorial, we will show you way to build a Custom Element from Angular Component using Angular Elements.

**Contents**[[hide](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example)]

* [Angular Elements](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Angular_Elements)
* [Step by Step](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Step_by_Step)
  + - [Setup Environment](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Setup_Environment)
      * [Install latest Angular CLI](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Install_latest_Angular_CLI)
      * [Install Angular Elements Package](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Install_Angular_Elements_Package)
    - [Build Custom Element](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Build_Custom_Element)
      * [Add Angular Component](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Add_Angular_Component)
      * [Convert to Custom Element](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Convert_to_Custom_Element)
    - [Use Custom Element](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Use_Custom_Element)
      * [Inside Angular Project](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Inside_Angular_Project)
      * [Outside Angular Project](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Outside_Angular_Project)
* [Source Code](https://grokonez.com/frontend/angular/angular-6-elements-use-angular-elements-example-to-build-custom-elements-example#Source_Code)

**Angular Elements**

@angular/elements package includes a createCustomElement() function that converts a component into a class that can be registered with the browser as a Custom Element.



|  |  |
| --- | --- |
| 1  2  3  4 | ngDoBootstrap() {    const element = createCustomElement(JsaCounterComponent, { injector: this.injector });    customElements.define('jsa-counter', element);  } |

After registering configured class with the browser’s custom-element registry, we can use the new element just like a built-in HTML element:



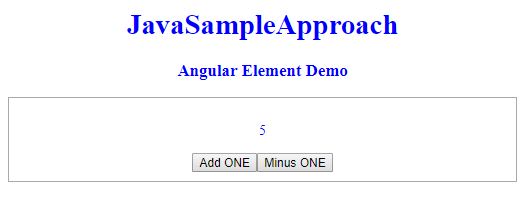
|  |  |
| --- | --- |
| 1 | <jsa-counter counter='3'></jsa-counter> |

Now our component-defined view with change detection and data binding, mapping Angular functionality automatically connect to the corresponding native HTML equivalents:

– Inside Angular Project:



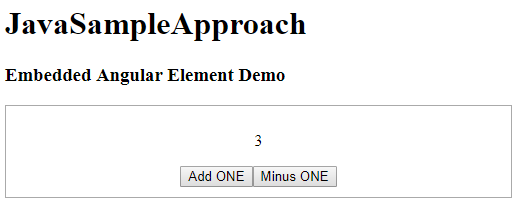
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <body>    <div style="color: blue; text-align: center">      <h1>JavaSampleApproach</h1>      <h3>Angular Element Demo</h3>        <jsa-counter counter='5'></jsa-counter>    </div>  </body> |



– Outside Angular Project:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <body>      <h1>JavaSampleApproach</h1>      <h3>Embedded Angular Element Demo</h3>        <jsa-counter counter='3'></jsa-counter>      <script type="text/javascript" src="./jsa-counter.js"></script>  </body> |



**Step by Step**

**Setup Environment**

***Install latest Angular CLI***

Angular Elements is a new feature of Angular 6. So to use it, we have to get latest Angular CLI to work with this new release.

Run this command: npm install -g @angular/cli@latest.

***Install Angular Elements Package***

First, create new Angular Project:



|  |  |
| --- | --- |
| 1  2 | ng new AngularElement  cd AngularElement |

Installing @angular/elements is just like a regular npm package, but now, the thing is easier to add elements to our project with new ‘add’ command:



|  |  |
| --- | --- |
| 1 | ng add @angular/elements |

**Build Custom Element**

***Add Angular Component***

*jsa-counter.component.html*



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <div class="jsa-element">    <p>{{counter}}</p>      <button (click)="increase()">Add ONE</button>    <button (click)="descrease()">Minus ONE</button>  </div> |

*jsa-counter.component.css*



|  |  |
| --- | --- |
| 1  2  3  4  5 | .jsa-element {      border: 1px solid #aaa;      padding: 10px;      text-align: center;  } |

Component CSS styles will be encapsulated into the Component’s view and won’t affect the rest of the application.  
We can set the view encapsulation mode in the component metadata.

*jsa-counter.component.ts*



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32 | import { Component, OnInit, ViewEncapsulation, Input } from '@angular/core';    @Component({    selector: 'app-jsa-counter',    templateUrl: './jsa-counter.component.html',    styleUrls: ['./jsa-counter.component.css'],    encapsulation: ViewEncapsulation.Native  })  export class JsaCounterComponent implements OnInit {      @Input()    set counter(counter: number) {      this.\_counter = counter;    }    get counter(): number {      return this.\_counter;    }    \_counter = 0;      constructor() { }      ngOnInit() {    }      increase() {      this.counter++;    }      descrease() {      this.counter--;    }  } |

***Convert to Custom Element***

To do this, inside **app.module.ts**, follow the steps below:  
– Add your component to entryComponents of @NgModule decorator. This will exclude the Component from compilation and avoid startup warnings or errors.  
– Implement the ngDoBootstrap() method to manually bootstrap that app. We don’t register a bootstrap component with this module.  
– Call createCustomElement() to convert Angular Component (together with its dependencies) to a Custom Element.  
– Use customElements.define() JavaScript function to register the configured constructor and its associated custom-element tag with the browser’s CustomElementRegistry.

*app.module.ts*



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | import { BrowserModule } from '@angular/platform-browser';  import { NgModule, Injector } from '@angular/core';  import { createCustomElement } from '@angular/elements';    import { AppComponent } from './app.component';  import { JsaCounterComponent } from './jsa-counter/jsa-counter.component';    @NgModule({    declarations: [      AppComponent,      JsaCounterComponent    ],    entryComponents: [      JsaCounterComponent    ],    imports: [      BrowserModule    ],    providers: [],    bootstrap: []  })  export class AppModule {    constructor(private injector: Injector) { }      ngDoBootstrap() {      const element = createCustomElement(JsaCounterComponent, { injector: this.injector });      customElements.define('jsa-counter', element);    }  } |

**Use Custom Element**

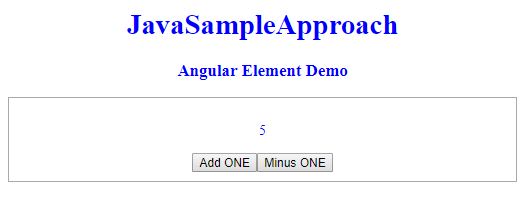
***Inside Angular Project***

Add the Custom Element to **src/index.html**:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | <body>    <div style="color: blue; text-align: center">      <h1>JavaSampleApproach</h1>      <h3>Angular Element Demo</h3>        <jsa-counter counter='5'></jsa-counter>    </div>  </body> |

Run command ng serve or npm start.  
Open browser with url http://localhost:4200/:



***Outside Angular Project***

These things should be done:  
– build production including ***runtime.js***, ***polyfills.js***, ***scripts.js*** and ***main.js***  
– concatenate these files into a single javascript file ***jsa-counter.js***

To do them, follow the steps below:  
– Install *fs-extra* and *concat* packages by running command: npm install fs-extra concat.

– Under project folder, create ***elements-build.js*** file:



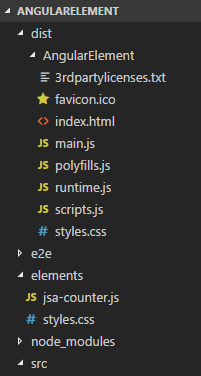
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | const fs = require('fs-extra');  const concat = require('concat');    (async function build() {      const files = [          './dist/AngularElement/runtime.js',          './dist/AngularElement/polyfills.js',          './dist/AngularElement/scripts.js',          './dist/AngularElement/main.js',      ]        await fs.ensureDir('elements');        await concat(files, 'elements/jsa-counter.js');        await fs.copyFile('./dist/AngularElement/styles.css', 'elements/styles.css');    })() |

– In **package.json**, add a build command to NPM scripts:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | {    ...    "scripts": {      ...      "build:elements": "ng build --prod --output-hashing none && node elements-build.js"    },  } |

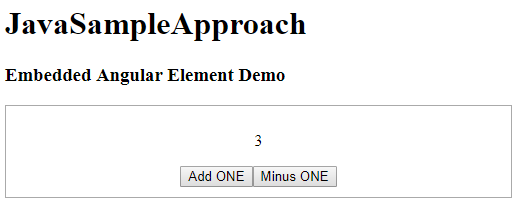
– Run command: npm run build:elements, we can see that the project tree is updated:



– With ***jsa-counter.js*** file, we can use jsa-counter element in any HTML page:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <body>      <h1>JavaSampleApproach</h1>      <h3>Embedded Angular Element Demo</h3>        <jsa-counter counter='3'></jsa-counter>      <script type="text/javascript" src="./jsa-counter.js"></script>  </body> |



**Source Code**

– [AngularElement](https://grokonez.com/wp-content/uploads/2018/05/AngularElement.zip)  
– [custom-element-test](https://grokonez.com/wp-content/uploads/2018/05/custom-element-test.zip) << just use browser to open **index.html** for testing.   
  
By [grokonez](https://grokonez.com/author/javasampleapproachco) | May 25, 2018.