



```
function(b, c) {  
  var d = a(c.form.querySelectorAll('input[type=checkbox][name="' + b.el.name + '"']));  
  if (0 === d.index(b.el)) {  
    var e = d.filter(":checked").length;  
    return e >= b.arg || g.minChecked.replace("{count}", b.arg)  
  }  
},  
maxSelected: function(a) {  
  return null !== a.val ? a.val  
},  
minSelected: function(a) {  
  return null !== a.val && a.val  
},  
radio: function(b) {  
  var c = a(this.form.querySelector  
  return 1 === c  
},  
custom: function(a, b) {  
  var c = b.options.custom[a.arg],  
  d = new RegExp(c.pattern);  
  return d.test(a.val) || c.errorMessage  
},  
remote: function(a) {  
  a.remote = a.arg  
}
```

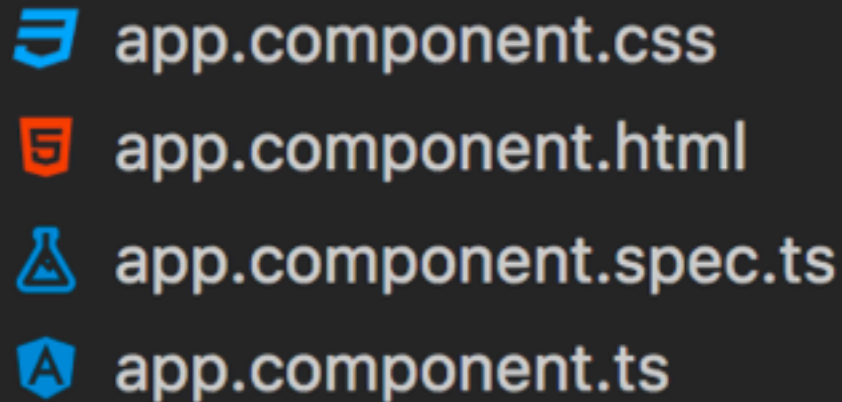
Formación Desarrollo Aplicaciones Web con Angular 5

7.- Creación de componentes



Angular | Creación de componentes

- Componente

Podemos definir un componente como un elemento que encapsula los datos, el marcado HTML y la lógica de las vistas de una aplicación.



A dark-themed code block containing four lines of text, each preceded by an Angular icon. The icons are: a blue 'E' for CSS, an orange 'H' for HTML, a blue flask for spec.ts, and a blue 'A' for the main .ts file.

-  `app.component.css`
-  `app.component.html`
-  `app.component.spec.ts`
-  `app.component.ts`

Angular | Creación de componentes

- Archivos de un componente

Por convención, se caracterizan por un nombre seguido de punto y el sufijo component.

<code>nombre.component.css</code>	Estilos CSS propios (prioridad sobre globales)
<code>nombre.component.html</code>	Plantilla con código html del componente
<code>nombre.component.spec.ts</code>	Test unitario del componente
<code>nombre.component.ts</code>	TypeScript con la clase del componente.

Angular | Creación de componentes

- Archivo de la clase del componente TypeScript

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})

export class AppComponent {
  title = 'app';
}
```

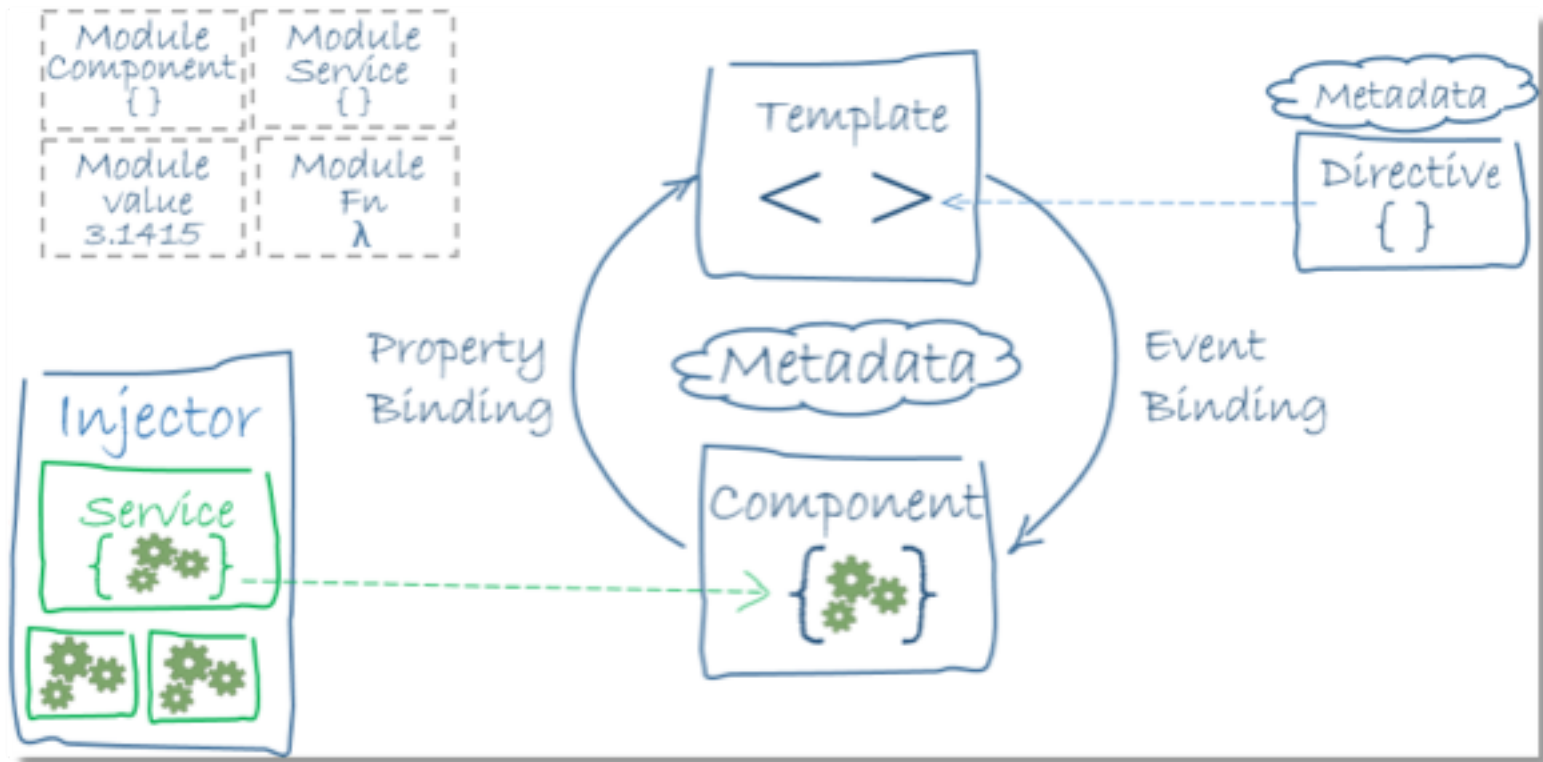
➔ Importaciones

➔ Función decoradora a la que se le pasa un objeto de Metadatos

➔ Clase del componente con el código TypeScript

Angular | Creación de componentes

- Arquitectura del componente



Angular | Creación de componentes

- Ejemplo '¡Hola Mundo!'

```
app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8
9  export class AppComponent {
10 |   mensaje: any = 'Hola Mundo!';
11 }
```

```
app.component.html x
1 | <h1>{{ mensaje }}</h1>
```

```
app.component.css x
1  h1 {
2    color: crimson;
3  }
```



Angular | Creación de componentes

- Módulos Angular

Podemos definir un módulo en Angular (no confundir con los módulos ECMAScript 6) como un contenedor que agrupa componentes relacionados por su funcionalidad.

Una aplicación Angular tendrá siempre al menos un módulo, el raíz, generado por Angular CLI con el nombre app.

Angular | Creación de componentes

- Módulos Angular

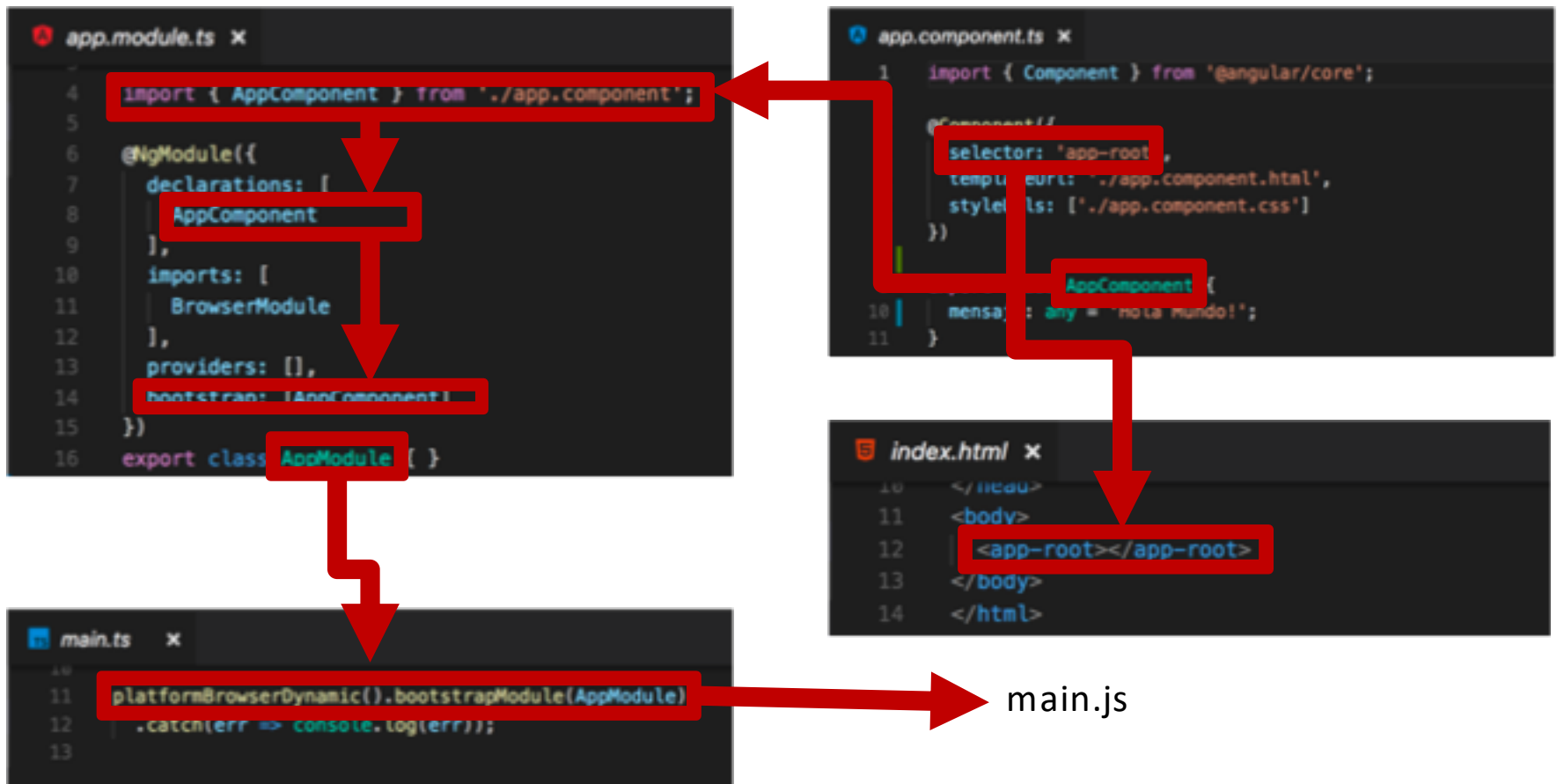
```

app.module.ts x
1  import { BrowserModule } from '@angular/platform-browser';
2  import { NgModule } from '@angular/core';
3
4  import { AppComponent } from './app.component';
5
6  @NgModule({
7    declarations: [
8      AppComponent
9    ],
10   imports: [
11     BrowserModule
12   ],
13   providers: [],
14   bootstrap: [AppComponent]
15 })
16 export class AppModule { }
  
```

- ➔ Importación librerías
- ➔ Importación elementos aplicación
- ➔ Función decoradora con elementos aplicación
- ➔ Clase del módulo

Angular | Creación de componentes

- Módulo y componente raíz



Angular | Creación de componentes

- Generación de componentes

De manera manual o desde con angular cli desde el directorio raíz del proyecto:

```
ng generate component nombredelcomponente
```

Se implementa en la plantilla de otro componente a través de la etiqueta definida en su selector:

```
<nombredelcomponente></nombredelcomponente>
```

Angular | Creación de componentes

- Simplificación de componentes

Para componentes muy sencillos se puede prescindir de la plantilla y/o el archivo CSS, incorporando el código a los metadatos:

```
@Component({  
  selector: 'app-nombre',  
  template: `  
    <!-- código HTML -->  
  `,  
  styles: [' /* reglas CSS */ ']  
})
```