ARCHITECTURE

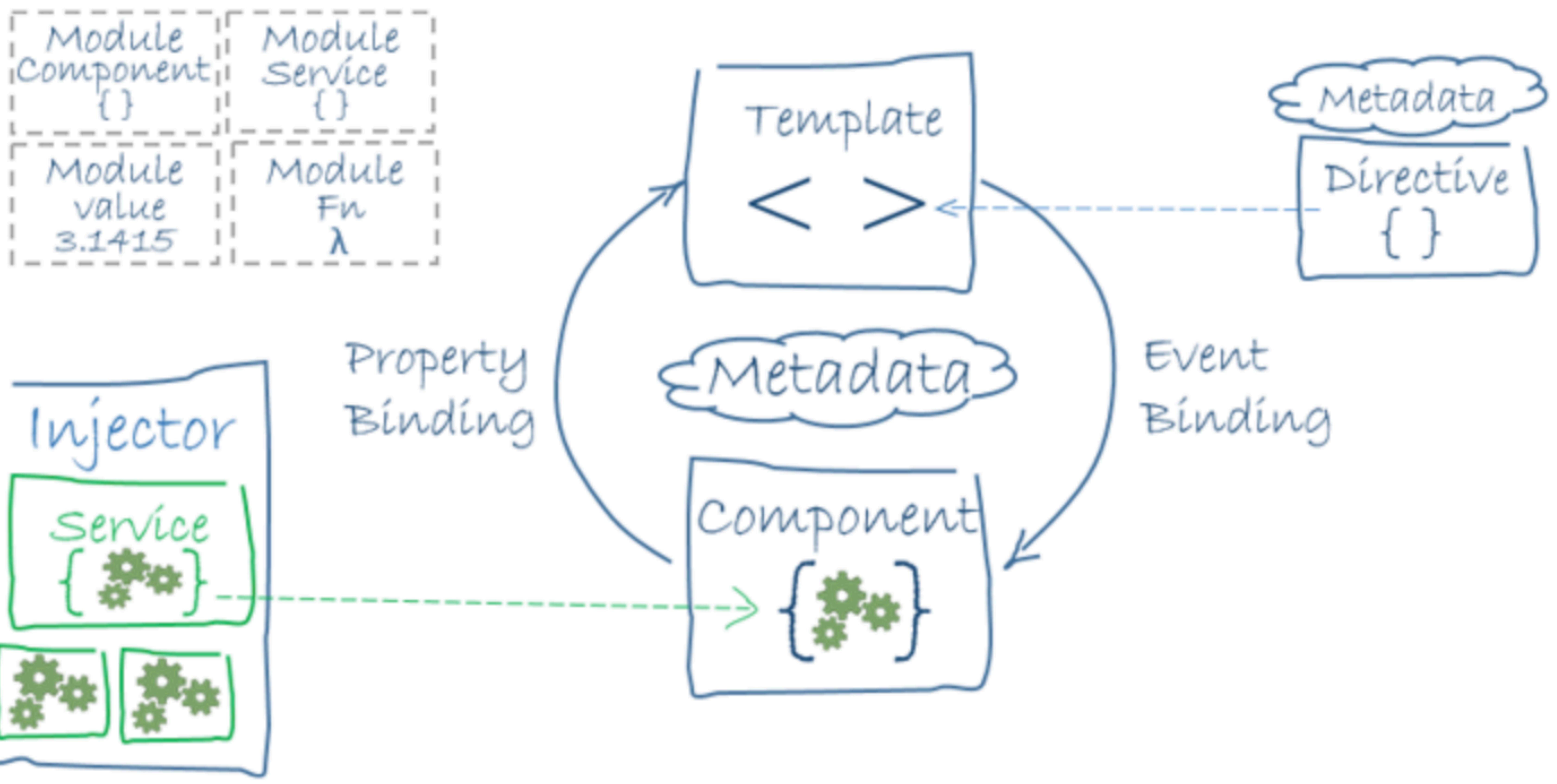
📖 ANGULAR

Angular is a platform and framework for building client applications in HTML and TypeScript. Angular is written in TypeScript.

📖 BASIC BLOCKS

- NgModules
- Components
- Views
- Services
- Router

📖 BASIC BLOCKS



📖 MODULES

Angular apps are modular and Angular has its own modularity system called NgModules. NgModules are containers for a cohesive block of code dedicated to an application domain, a workflow, or a closely related set of capabilities.

Every Angular app has at least one NgModule class, the root module, which is conventionally named AppModule and resides in a file named app.module.ts. You launch your app by bootstrapping the root NgModule.

📖 MODULE METADATA

- declarations: The components, directives, and pipes that belong to this NgModule.
- exports: The subset of declarations that should be visible and usable in the component templates of other NgModules.
- imports: Other modules whose exported classes are needed by component templates declared in this NgModule.
- providers: Creators of services that this NgModule contributes to the global collection of services; they become accessible in all parts of the app.
- bootstrap: The main application view, called the root component, which hosts all other app views.

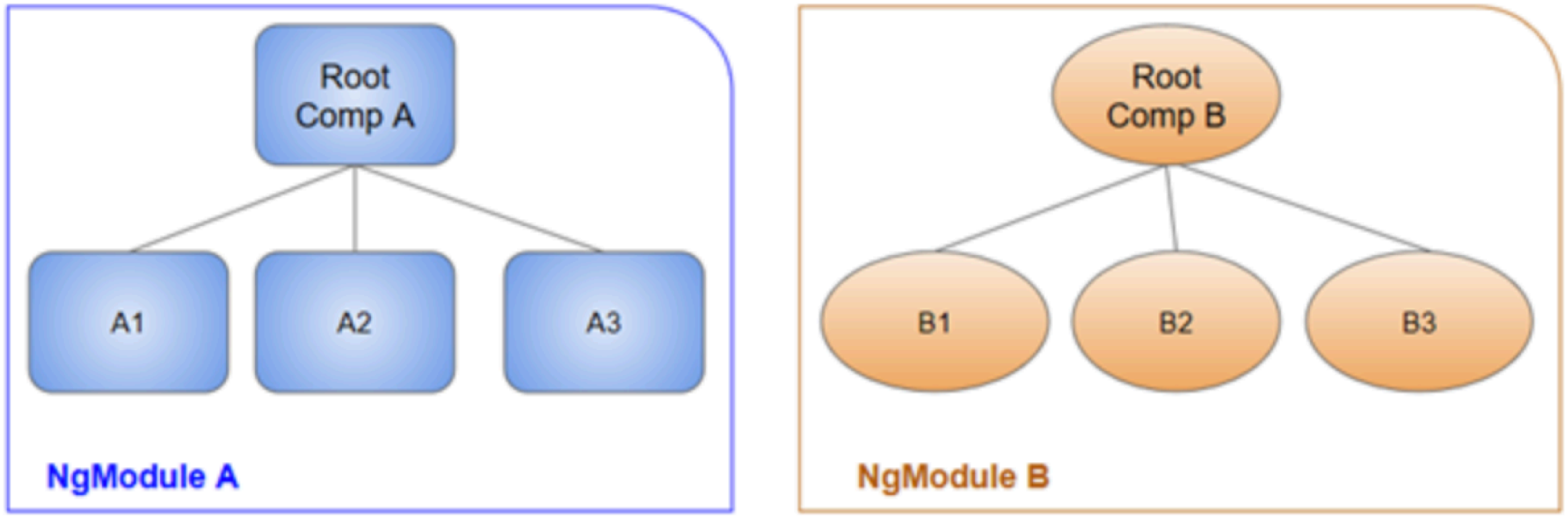
TBC

📖 MODULE METADATA

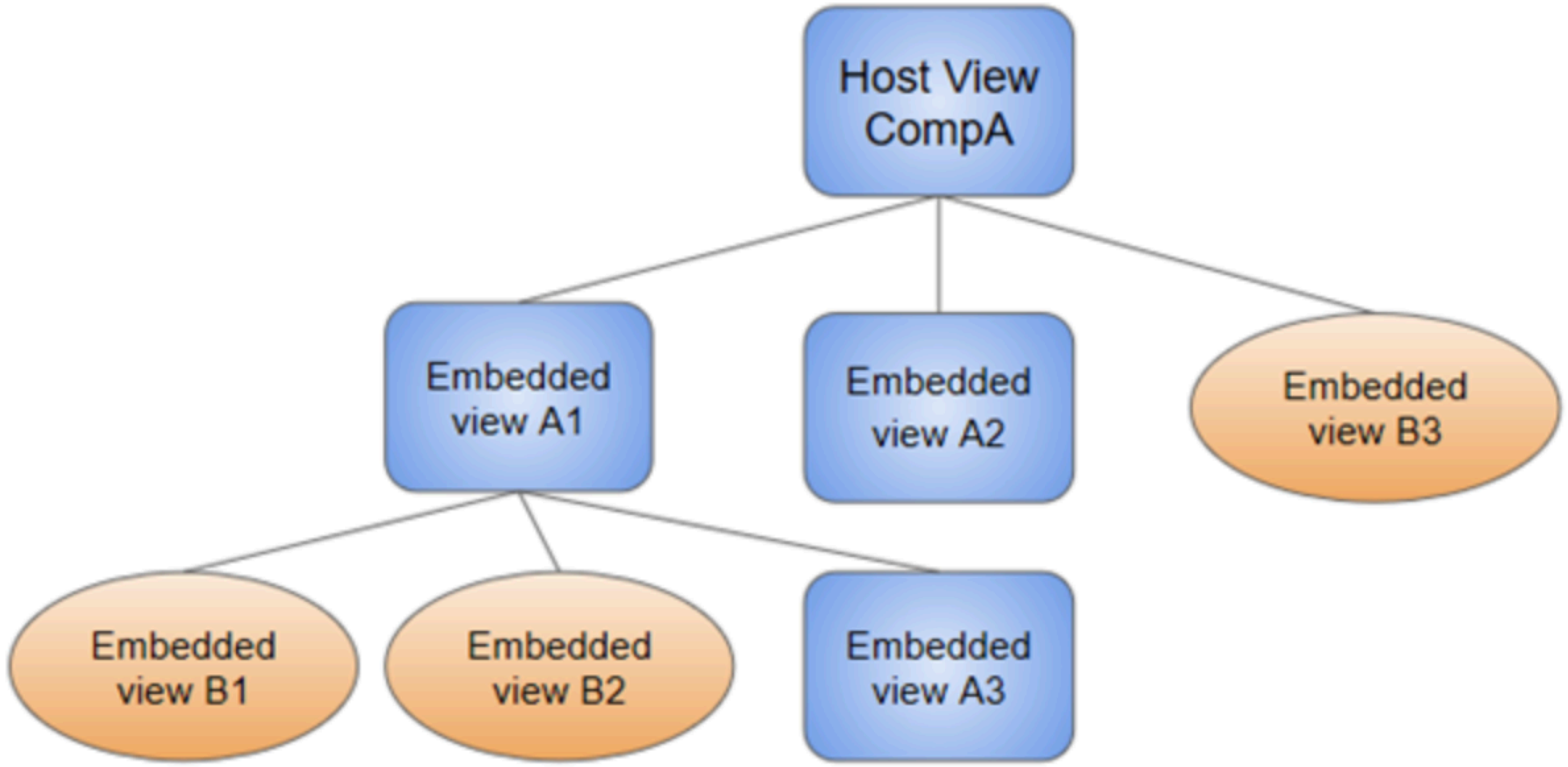
```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

TBC

📖 MODULE AND COMPONENTS



📖 COMPONENTS AND VIEWS



📖 COMPONENTS

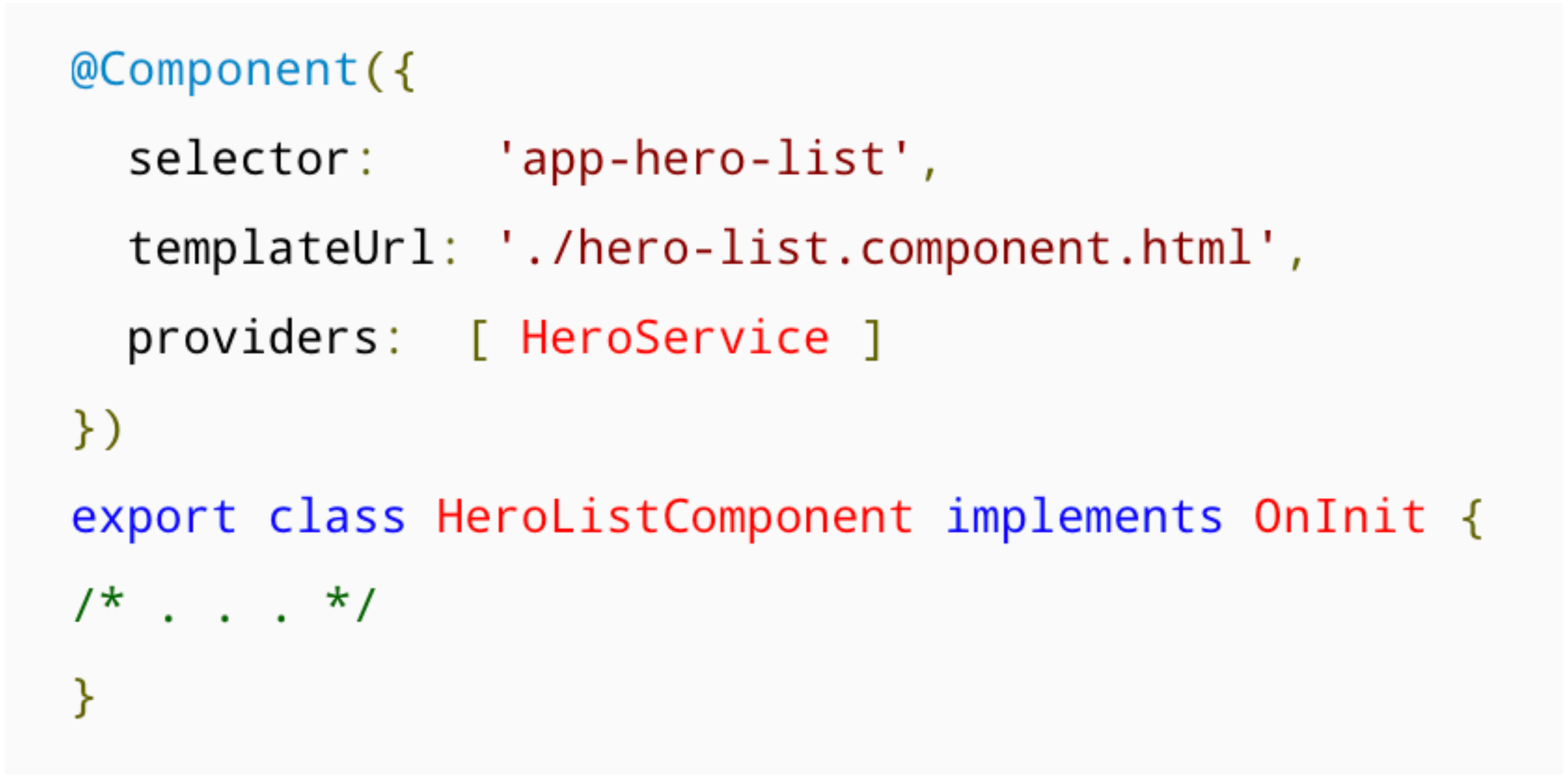
A component controls a patch of screen called a view.

You define a component's application logic—what it does to support the view—inside a class. The class interacts with the view through an API of properties and methods.

📖 COMPONENTS

```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

📖 COMPONENTS METADATA



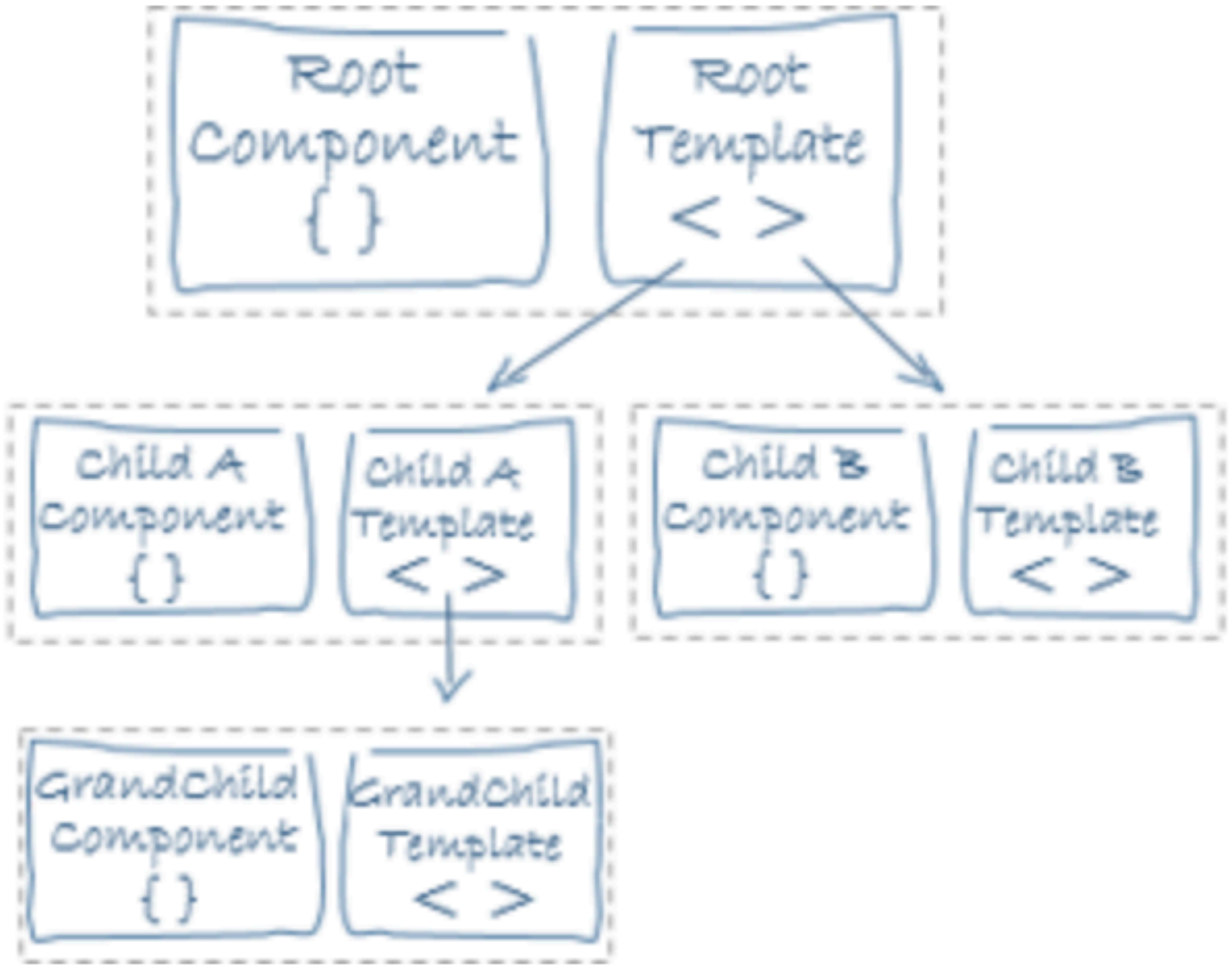
```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

TBC

📖 TEMPLATES AND VIEWS

You define a component's view with its companion template. A template is a form of HTML that tells Angular how to render the component.

📖 TEMPLATES AND VIEWS



TBC

📖 TEMPLATES AND VIEWS

```html
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<app-hero-detail *ngIf="selectedHero" [hero]="selectedHero"></app-hero-detail>
```

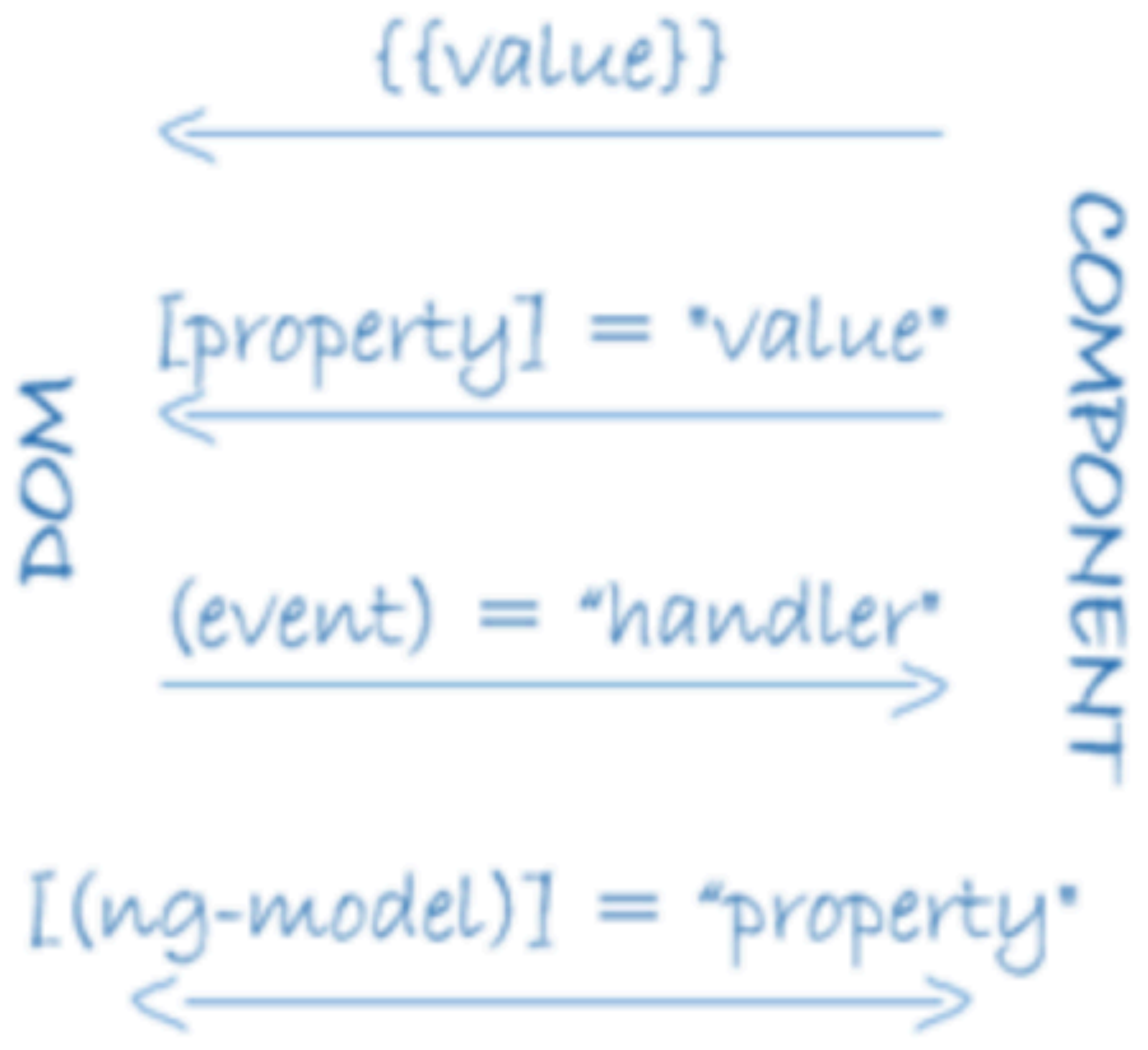
TBC

📖 DATA BINDING

Angular supports two-way data binding, a mechanism for coordinating the parts of a template with the parts of a component. Add binding markup to the template HTML to tell Angular how to connect both sides.

TBC

📖 DATA BINDING



TBC

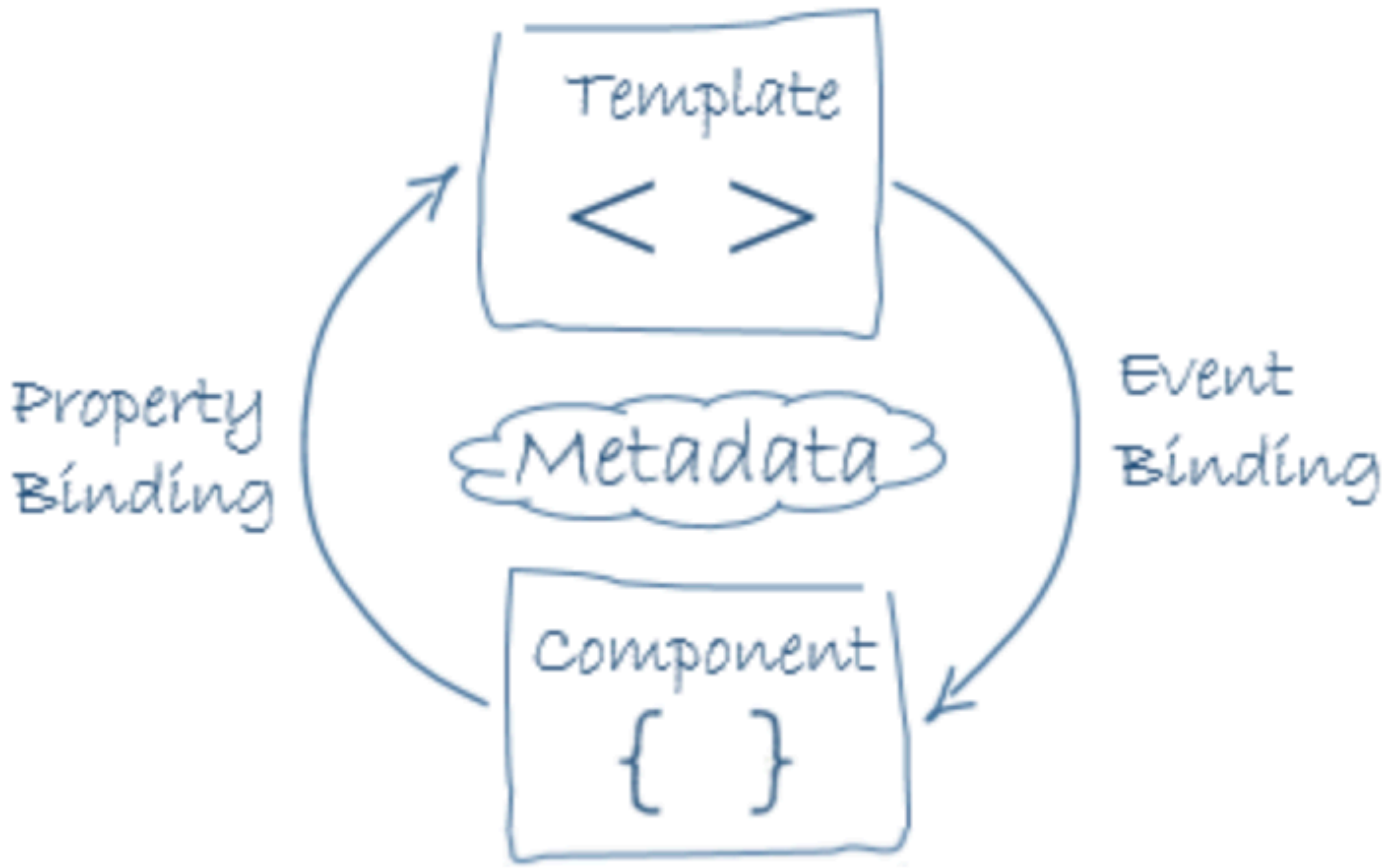
📖 DATA BINDING

```
<li>{{hero.name}}</li>

<app-hero-detail [hero]="selectedHero"></app-hero-detail>

<li (click)="selectHero(hero)"></li>
```

```
<input [(ngModel)]="hero.name">
```

TBC

📖 DATA BINDING



TBC

📖 PIPES

```
<!-- Default format: output 'Jun 15, 2015'-->
 <p>Today is {{today | date}}</p>

<!-- fullDate format: output 'Monday, June 15, 2015'-->
<p>The date is {{today | date:'fullDate'}}</p>

 <!-- shortTime format: output '9:43 AM'-->
 <p>The time is {{today | date:'shortTime'}}</p>
```

📖 SERVICES AND DI

Service is a broad category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

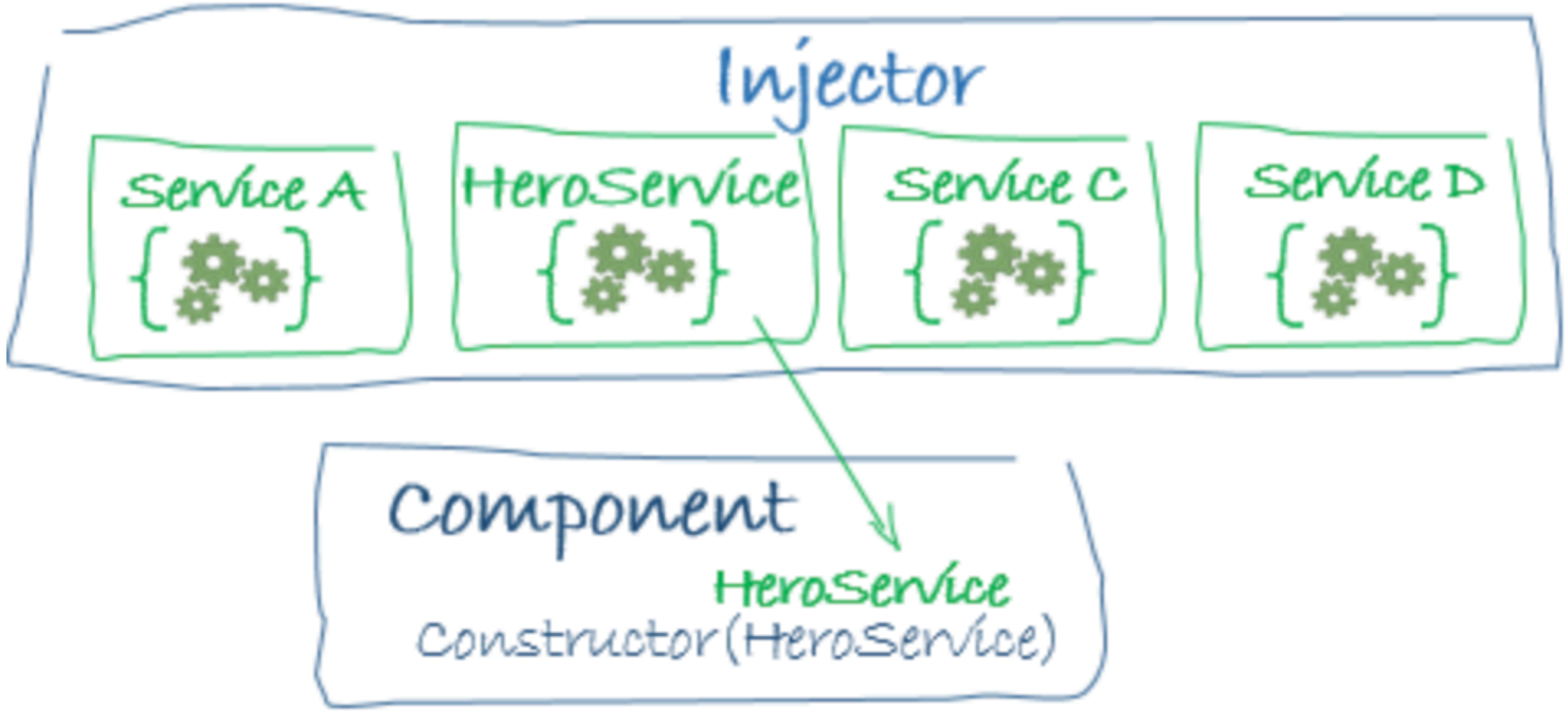
📖 SERVICES AND DI

```
export class Logger {
  log(msg: any)   { console.log(msg); }
  error(msg: any) { console.error(msg); }
  warn(msg: any)  { console.warn(msg); }
}
```

📖 SERVICES AND DI

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

📖 DEPENDENCY INJECTION



HERE GOES!

TBC