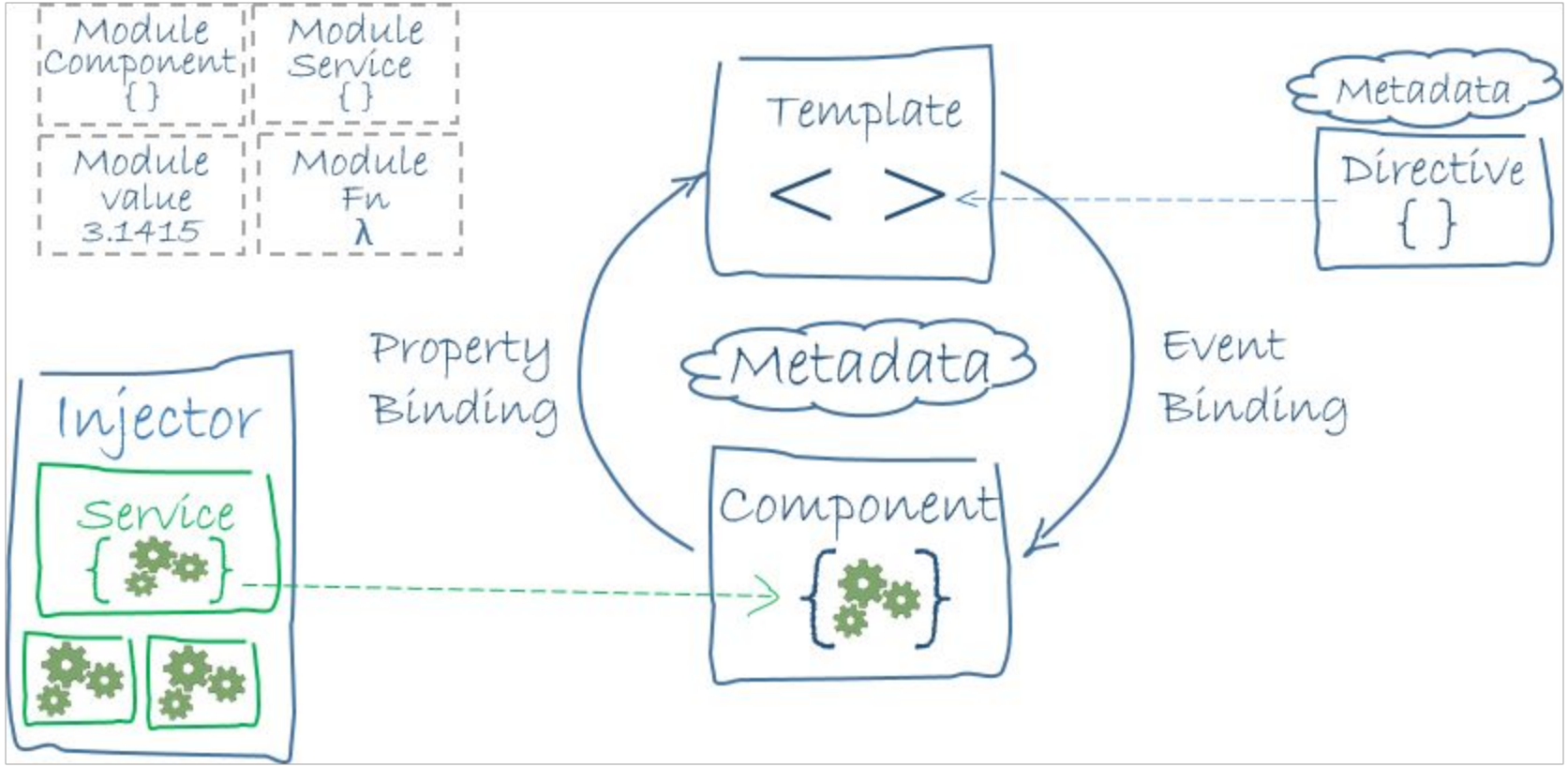
Piotr Skurski

AtScript (@Script)

- Bazuje na języku TypeScript
- Składnia ECMAScript 6
- Adnotacje
- Sprawdzanie typu podczas działania programu (runtime)
- Kompilowany do JavaScript-u

Architektura

- Moduły (Modules)
- Komponenty (Components)
- Szablony (Templates)
- Metadane (Metadata)
- Wiązanie danych (Data binding)
- Dyrektywy (Directives)
- Serwisy (Services)
- Wstrzykiwanie zależności (Dependency injection)


Module
Component
{}
Module
Service
{}
Module
value
3.1415
Module
Fn
λ
Template
< >
Metadata
Directive
{}
Property
Binding
Metadata
Event
Binding
Injector
Service
Component

Moduły

- @NgModule - adnotacja (dekorator) klasy
- AppModule - każda aplikacja ma przynajmniej jeden główny moduł

```
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 { }
```

Komponenty

- Klasa
- Kontroluje element strony zwany “widokiem”
- Tutaj definiujemy logikę
- Kontroluje widok poprzez implementacje zdefiniowanych metod

```
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; }
}
```

Szablony

- Definiujemy jak renderowany będzie widok
- HTML + dyrektywy (strukturalne i atrybutów)

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

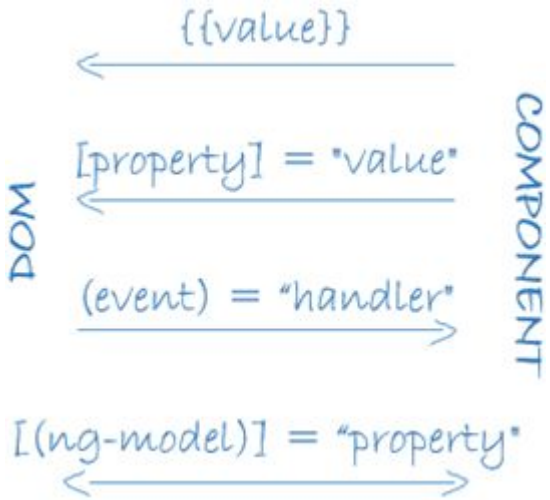
Metadane

- Informują Angulara jak przetwarzać daną klasę
- Są zawarte w adnotacjach (dekoratorach)

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

Wiązanie danych (Data binding)

- Mechanizm koordynacji danych między widokiem a komponentem
- 4 sposoby




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

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

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

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

Dyrektywy

- Klasa z adnotacją @Directive
- Szablony Angulara są dynamiczne (własny kompilator HTML)
- Strukturalne dyrektywy

```
<li *ngFor="let hero of heroes"></li>
<hero-detail *ngIf="selectedHero"></hero-detail>
```

- Dyrektywy atrybutu - element HTML

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

Serwisy

- Klasyczne rozumienie Serwisu (Angular nie wprowadza własnego)
- Klasa dostarczająca określoną funkcjonalność
- Komponenty korzystają z serwisów

```
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;
  }
}
```

Wstrzykiwanie zależności (dependency injection)

- Dostarczenie obiektowi zależności jakich potrzebuje
- Większość zależności to serwisy
- Injector - odpowiedzialny za dostarczenie odpowiedniego serwisu
- Injector - utrzymuje kontener z serwisami
- Jeśli w kontenerze nie ma serwisu, Injector tworzy nowy

```
constructor(private service: HeroService) { }
```

```
providers: [BackendService, HeroService, Logger],
```

Linki

- https://angular.io/docs/ts/latest/guide/