



# Angular

Vojtech Mašek

Head of engineering

@flow<sup>up</sup>



vmasek



vmasek



VojtechMasek



vojtechmasek



# What is Angular?

- Platform
- Combines framework and tooling
- Integrates best practices
- Empowers web, mobile and desktop

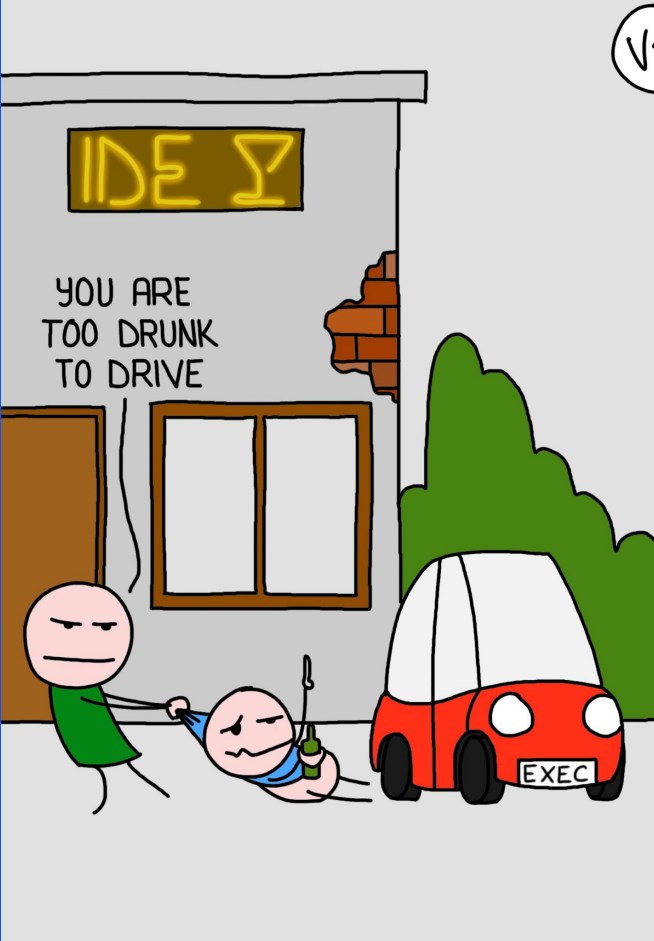


# TypeScript

- **Type** safety
- Superset of JavaScript (JS)
- **Compiles** to JS
- Latest JS features (+*more*)

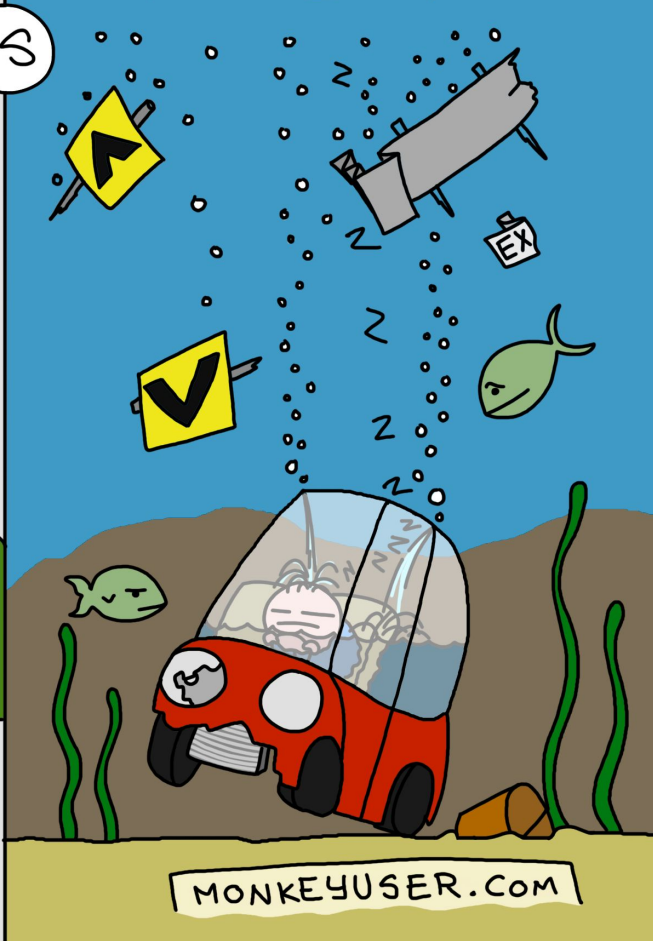


# COMPILE TIME ERROR



VS

# RUNTIME ERROR





# What is included in Angular

flow<sup>up</sup>

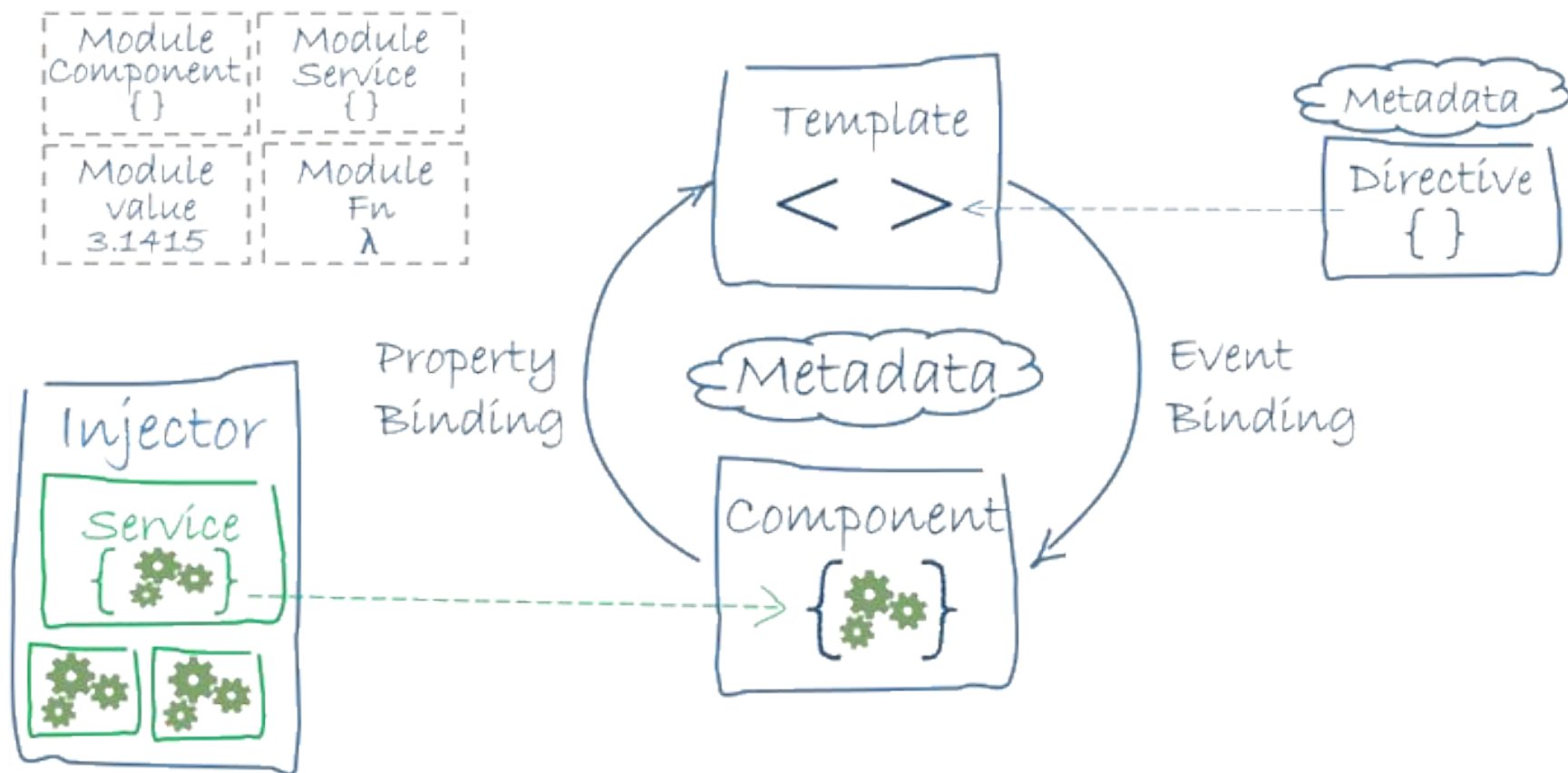
- Form Builder
- Routing
- HTTP Client
- Animations
- Templating
- Shadow DOM
- Tests suite
- Change Detection
- Observables (RxJS)
- CLI + build tooling
- Server side rendering
- Dependency Injection
- Schematics
- PWA (+ service worker)
- ... more

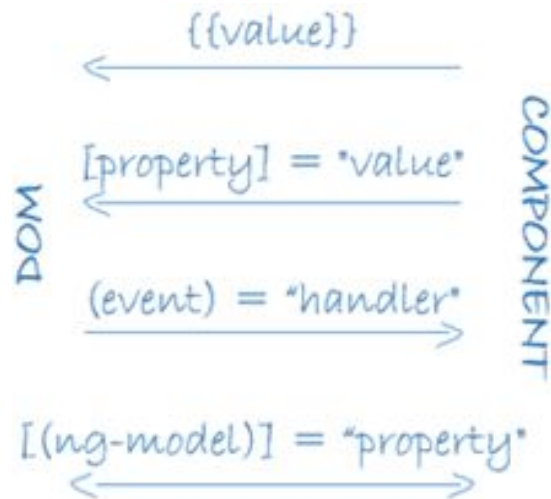


# External

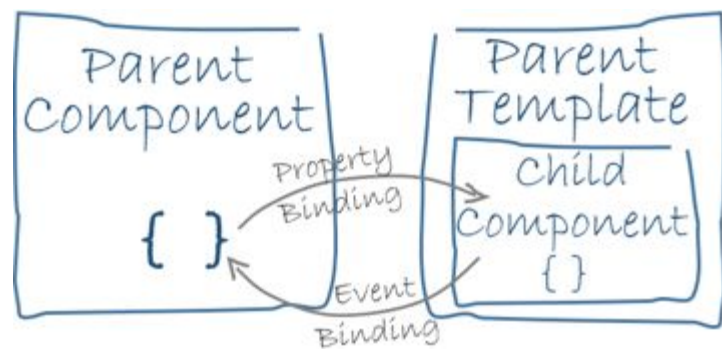
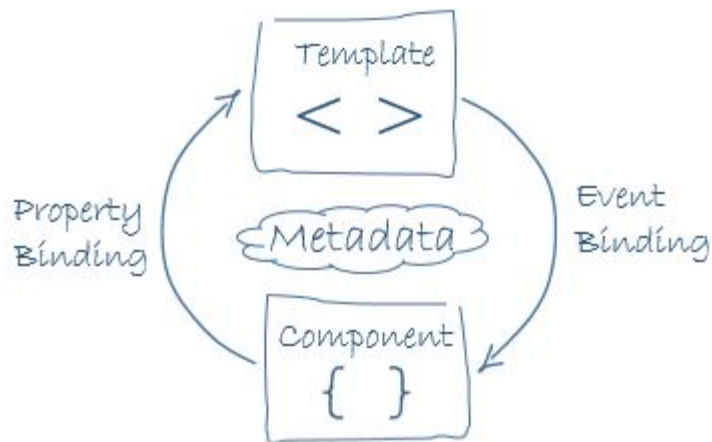
(but still maintained with angular release cycle)

- Material framework
- Angular Fire (reactive Firebase SDK)
- NgRX (Redux/Flux + RxJS based state management)











# Simple component

```
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     <h1>{{title}}</h1>
7     <h2>My favorite hero is: {{myHero}}</h2>
8   `
9 })
10 export class AppComponent {
11   title = 'Tour of Heroes';
12   myHero = 'Windstorm';
13 }
14
```



# Pipes

Are the way to write display-value transformations that you can declare in your HTML.



# Built-in pipes

- Async
- Currency
- Date
- I18n Plural
- I18n Select
- Json
- KeyValue
- Decimal
- Percent
- Slice
- LowerCase
- TitleCase
- UpperCase



# Date pipe

```
1 <!-- 'Jun 15, 2015' →  
2 <p>Today is {{ today | date }}</p>  
3  
4 <!-- 'Monday, June 15, 2015' →  
5 <p>The date is {{ today | date:'fullDate' }}</p>  
6  
7 <!-- '9:43 AM' →  
8 <p>The time is {{ today | date:'shortTime' }}</p>
```



# Custom pipe

```
1 /** Formats seconds to duration in format HH:mm:ss. */
2 @Pipe({ name: 'myDuration' })
3 export class DurationPipe implements PipeTransform {
4   transform(value?: number): string | undefined {
5     if (!value) return;
6
7     const hours = Math.floor(duration / SECONDS_IN_HOUR);
8     const durationModHours = duration % SECONDS_IN_HOUR;
9     const minutes = Math.floor(durationModHours / SECONDS_IN_MINUTE);
10    const seconds = durationModHours % SECONDS_IN_MINUTE;
11
12    if (hours === 0) {
13      return `${minutes}:${this.pad(seconds)}`;
14    }
15
16    return `${hours}:${this.pad(minutes)}:${this.pad(seconds)}`;
17  }
18
19  pad = (num: number): string => num.toString().padStart(2, '0');
20 }
```

```
<!-- 0:01 -->
{{ 1 | myDuration }}

<!-- 0:42 -->
{{ 42 | myDuration }}

<!-- 4:20 -->
{{ 260 | myDuration }}

<!-- 19:17 -->
{{ 69420 | myDuration }}
```



# Directives

## Attribute

- appearance or behavior of a DOM element

## Structural

- responsible for HTML layout
- shape the DOM's structure
  - ◆ typically by adding, removing, or manipulating elements



# \*ngFor structural directive

```
1 <div *ngFor="let hero of heroes; let i=index; let odd=odd; trackBy: trackById"
2     [class.odd]="odd"
3 >
4   Index: {{i}} → Name: {{hero.name}}
5 </div>
```

Index: 0 → Name: Foo

Index: 1 → Name: Bar

Index: 2 → Name: Wololo

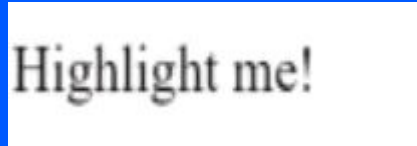
Index: 3 → Name: Name





# Custom attribute directive

```
1 /** Usage <p appHighlight>Highlight me!</p> */
2 @Directive({ selector: '[appHighlight]' })
3 export class HighlightDirective {
4   constructor(private el: ElementRef) { }
5
6   @HostListener('mouseenter') onMouseEnter() {
7     this.highlight('yellow');
8   }
9
10  @HostListener('mouseleave') onMouseLeave() {
11    this.highlight(null);
12  }
13
14  private highlight(color: string) {
15    this.el.nativeElement.style.backgroundColor = color;
16  }
17 }
```



Highlight me!



# Custom structural directive

```
1 <label>
2   Select a year:
3   <select>
4     <ng-container *appRange="[2015, 2018]; let num">
5       <option [value]="num">{{num}}</option>
6     </ng-container>
7   </select>
8 </label>
```

Select a year:

- 2015
- 2016
- 2017
- 2018



# Simple component

```
1 @Component({
2   selector: 'app-hero-parent',
3   template: `
4     <h2>{{master}} controls {{heroes.length}} heroes</h2>
5     <app-hero-child
6       *ngFor="let hero of heroes"
7       [hero]="hero"
8       [master]="master">
9     </app-hero-child>
10  `
11 })
12 export class HeroParentComponent {
13   heroes = HEROES;
14   master = 'Master';
15 }
16
```

```
1 @Component({
2   selector: 'app-hero-child',
3   template: `
4     <h3>{{hero.name}} says:</h3>
5     <p>
6       I, {{hero.name}},
7       am at your service, {{masterName}}.
8     </p>
9   `
10 })
11 export class HeroChildComponent {
12   @Input() hero: Hero;
13   @Input('master') masterName: string;
14 }
```

## Master controls 2 heroes

### Mr. IQ says:

I, Mr. IQ, am at your service, Master.

### Mr. IQ says:

I, Mr. IQ, am at your service, Master.



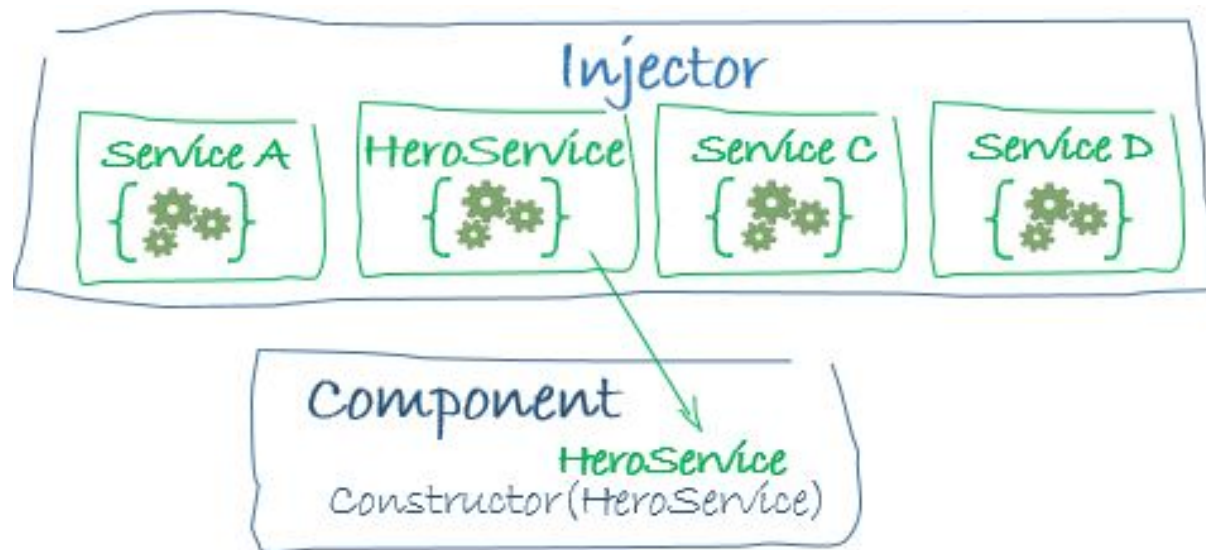
# Component binding

```
1 @Component({
2   selector: 'app-little-tour',
3   template: `
4     <input #newHero
5       (keyup.enter)="addHero(newHero.value)"
6     >
7
8     <button (click)="addHero(newHero.value)">Add</button>
9
10    <ul>
11      <li *ngFor="let hero of heroes">{{hero}}</li>
12    </ul>
13  `
14 })
15 export class LittleTourComponent {
16   heroes = ['Windstorm', 'Bombasto', 'Magenta', 'Tornado'];
17
18   addHero(heroName: string) {
19     if (heroName) {
20       this.heroes.push(heroName);
21     }
22   }
23 }
```

## Little Tour of Heroes

- Windstorm
- Bombasto
- Magenta
- Tornado





# Dependency Injection

```
@Component({
  selector: 'app-hero-list',
  template: `
    <div *ngFor="let hero of heroes | async">
      {{hero.id}} - {{hero.name}}
    </div>
  `
})
export class HeroListComponent {
  heroes: Observable<Hero[]>;

  constructor(heroService: HeroService) {
    this.heroes = heroService.getHeroes();
  }
}
```

```
1 export interface Hero {
2   id: number;
3   name: string;
4   isSecret: false;
5 }
6
7 @Injectable({providedIn: 'root'})
8 export class HeroService {
9   constructor(
10     private logger: Logger,
11     private http: HttpClient
12   ) {}
13
14   getHeroes(): Observable<Hero[]> {
15     this.logger.log('Getting heroes ... ');
16     return this.http.get('/heroes');
17   }
18 }
```



# Start

```
1 npm install @angular/cli -g
2 ng new my-app && cd my-app
3 ng serve --open
```



# Advanced concepts

- Advanced state management
- Service workers and caching
- Optimizations
  - ◆ Pure pipes
  - ◆ Change detection





# Release process

- Semantic versioning
- Mayor version ~6 months
  - ◆ Patch release ~every other week
- Long Time Support
  - ◆ Mayor versions have 2 years LTS
  - ◆ Breaking changes does not occur with one version migration



# Updating to new version

```
ng update @angular/cli @angular/core
```



# Links

- [angular.io/start](https://angular.io/start)
- [angular.io/guide/quickstart](https://angular.io/guide/quickstart)
- [angular.io/tutorial](https://angular.io/tutorial)
- [material.angular.io](https://material.angular.io)



## Generate basic stuff

```
# create component
ng g c my-new-component

# create module
ng g m my-module

# create service in my module
ng g s my-module/my-service
```



## Material components

```
# add library
ng add @angular/material

# create navigation UI
ng g @angular/material:material-nav

# create dashboard UI
ng g @angular/material:material-dashboard
```



# Generate advanced stuff

```
ng new my-project --style=scss --routing=true

npm install @ngrx/{store,effects,entity,store-devtools,schematics} --save

ng generate @ngrx/schematics:store State --root --module app.module.ts

ng generate @ngrx/schematics:effect App --root --module app.module.ts

ng generate @ngrx/schematics:reducer Counter --reducers reducers/index.ts --group

ng g component counter
```



# Q&A

Vojtech Mašek

Head of engineering

@flow<sup>up</sup>



vmasek



vmasek



VojtechMasek



vojtechmasek