ANGULAR 2.0

Key features overview by Evgeny Dmitriev

FUNDAMENTALS

New Template syntax

Annotations

Observables

Zones

Routing

TEMPLATES

TEMPLATES 1.X

```
input type="text" class="form-control"
      uib-datepicker-popup="{{expression}}"
      ng-model="expression"
      is-open="expression"
      min-date="expression"
      max-date="expression"
      datepicker-options="expression"
      date-disabled="expression"
      ng-required="expression"
      close-text="text"
```

```
input type="text" class="form-control"
      uib-datepicker-popup="{{format}}"
      ng-model="dt"
      is-open="status.opened"
      min-date="minDate"
      max-date="maxDate"
      datepicker-options="dateOptions"
      date-disabled="disabled(date, mode)"
      ng-required="true"
      close-text="Close"
```

TEMPLATES 2.X

{{}} plain old interpolation

[] properties binding

() event binding

local variable binding

PROPERTIES BINDING

```
<img src="{{imageUrl}}">
```

Attributes are defined by HTML.

Properties are defined by DOM

```
<img [src]="imageUrl">
```

```
<input type="text" [disabled]="isDisabled">
```

```
<span [class.primary]="isPrimary"></span>
```

EVENTS BINDING

```
<button (click)="callMethod($event)"></button>
```

Any triggered event including child events

```
<button (keydown.space)="callMethod($event)"></button>
```

Upcoming change

LOCAL VARIABLES

```
<div ng-repeat="item in collection">
    {{item.name}}
</div>
```

```
<div *ng-for="#item of collection">
    {{item.name}}
</div>
```

```
<google-youtube #player=""></google-youtube>
<button (click)="player.play()">Play</button>
```

STILL A JS FRAMEWORK

In angular 1.x we have
.module(), .config(), run()
.provider(), .service(), .factory(), .value(), .constant(),
.controller()
.filter()

ANNOTATIONS

It might be es7 in the future
Currently supported in typescript and traceur
In fact standard spec supports decorators
We kinda have them in angular 1

UNDER THE HOOD

```
@Injectable()
class PageController{
    constructor(http: Http){
        this.http = http;
    }
}
```

```
function PageController($http) {
    this.http = http;
}
PageController = __decorate([
    angular2_1.Injectable(),
    __metadata('design:paramtypes', [Http])
], PageController);
return PageController;
```

ANGULAR ANNOTATIONS

@Component
@Pipe
@Inject
@Attribute, @ContentChildren, @HostBinding, @Optional, @Query, @ViewChild, @HostListener, @Output, @Input, @ViewChildren, @Host

COMPONENT ANNOTATION

```
@Component({
    selector: 'app',
    template: 'Hello {{name}}!'
})
class Greet {
    name: string;
    constructor() {
        this.name = 'World';
    }
}
```

OBSERVABLES

https://github.com/ReactiveX/RxJS

Http, custom events and propertis bindings, observable queries, async pipe

OBSERVABLE HTTP

```
http.get('api/person')
    .subscribe(person => this.person = person);
```

OBSERVABLE EVENTS

```
@Component({selector: 'fire-button'})
class TodoCmp {
    @Output() fire = new EventEmitter();
    onFireButton() {
        this.fire.next();
    }
}
```

```
<fire-button (fire)="onFire()"></fire-button>
```

OBSERVABLE PROPERTIES

```
@Component({
    selector: 'fire-button',
    template: 'fire text: {{fireText}}'
})
class FireCmp{
    @Input() fireText: string;
}
```

```
<fire-button [fire-text]="textProp"></fire-button>
```

OBSERVABLE QUERY

```
<tabs>
    <tab *ng-for="#tab of tabs"></tab>
</tabs>
```

```
@Component({
    selector: 'tabs',
    templateUrl: 'someTemplate',
    directives: [Tab]
})
class Tabs{
    @ViewChildren(Tab) children: QueryList<tab>;
    construct() {
        this.children.onChange((items) => this.total = items.length);
    }
}
</tab>
```

ROUTING

```
@Component({selector: 'app', templateUrl: "app.html"})
@RouteConfig([
    new Route({path: '/', component: DashboardPage, name: 'Dashboard'
    new Route({path: '/about', component: AboutCmp, name: 'About'}),
    new Route({path: '/detail/:id', component: DetailCmp, name: 'Deta
])
export class InboxApp {}
```

```
@Component({selector: 'detail', templateUrl: "detail.html"})
class DetailCmp {
    id: sting;

    constructor(params: RouteParams) {
        this.id = params.get('id');
    }
}
```