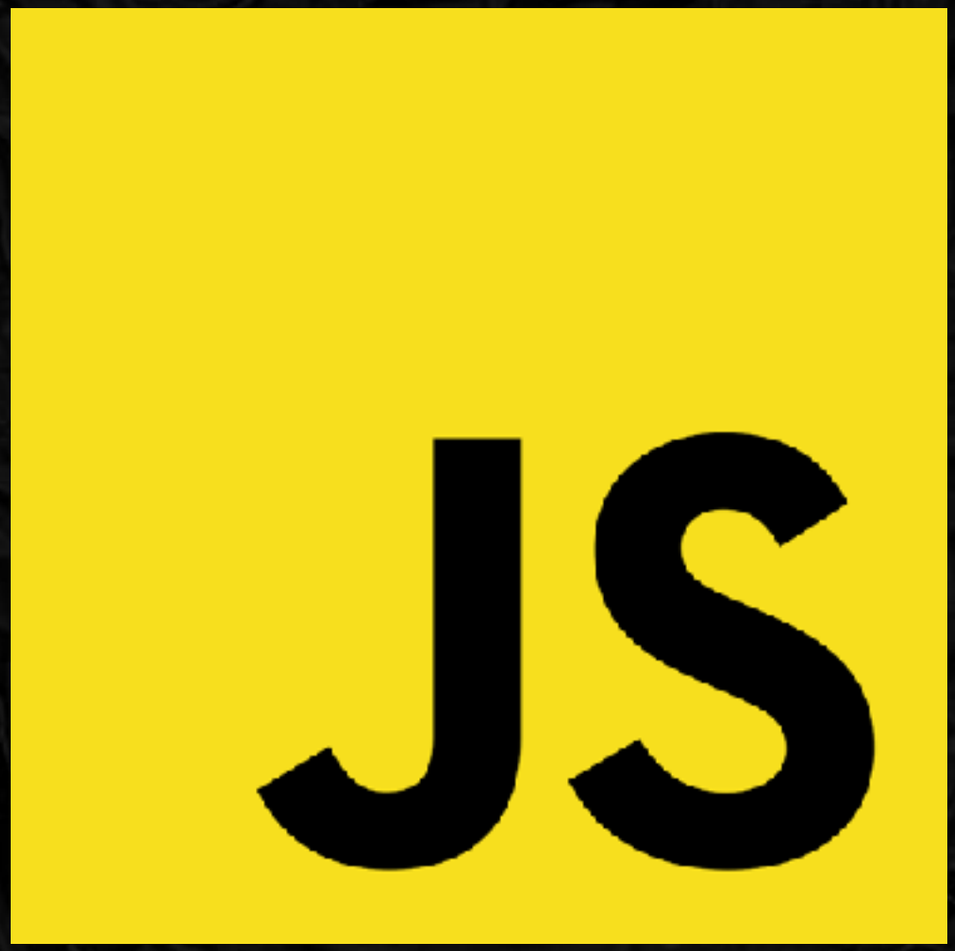


Pre-material Angular2 by Artem Koziar

JavaScript: Dynamic Typing

JavaScript: Environments

React Native, MongoDB, Widgets, Nginx, WinJS, …

JavaScript: ***JS

JS Core

JavaScript: Declarations

```
(function test() {

  if (1) {
    globalVar = 'No no David Blaine';
    var functionVar = 'Old school style';
    // ES6
    let variable = 'is ok';
    const IMMUTABLE_IN_GENERAL = 'some const';
    IMMUTABLE_IN_GENERAL[1] = 'OK O_o';
    IMMUTABLE_IN_GENERAL = 'ne ok';

    // Functions
    function oldStyleFunction() {}
    // ES6
    const f1 = () => {}; // Arrow Function
    let f2 = () => {};
    f2 = 1;
    f2();
  }

  console.log(globalVar, functionVar); // OK

  console.log(variable, IMMUTABLE_IN_GENERAL); // ne OK

  oldStyleFunction(); // OK
  f1(); // ne


})();
```

JavaScript: Types

Six data types that are primitives:

- Boolean
- Null
- Undefined
- Number
- String
- Symbol (new in ECMAScript 6)

and Object

```
(function test() {

  let a = 1;
  let b = 'a';
  let c = true;
  let d = {};
  let e = function () {};
  let f = () => {};
  let g = null;
  let h = undefined;

})();
```

Angular 2

for Kantar

by Artem Koziar

Plan

1. History from Peas King to Angular
2. Angular vs React
3. Angular 2 overview
4. CLI tool for Angular2
5. **TypeScript**
6. **Angular 2 Components**
7. Template syntax
8. Observables
9. Router
10. Q

Front-end Milestones

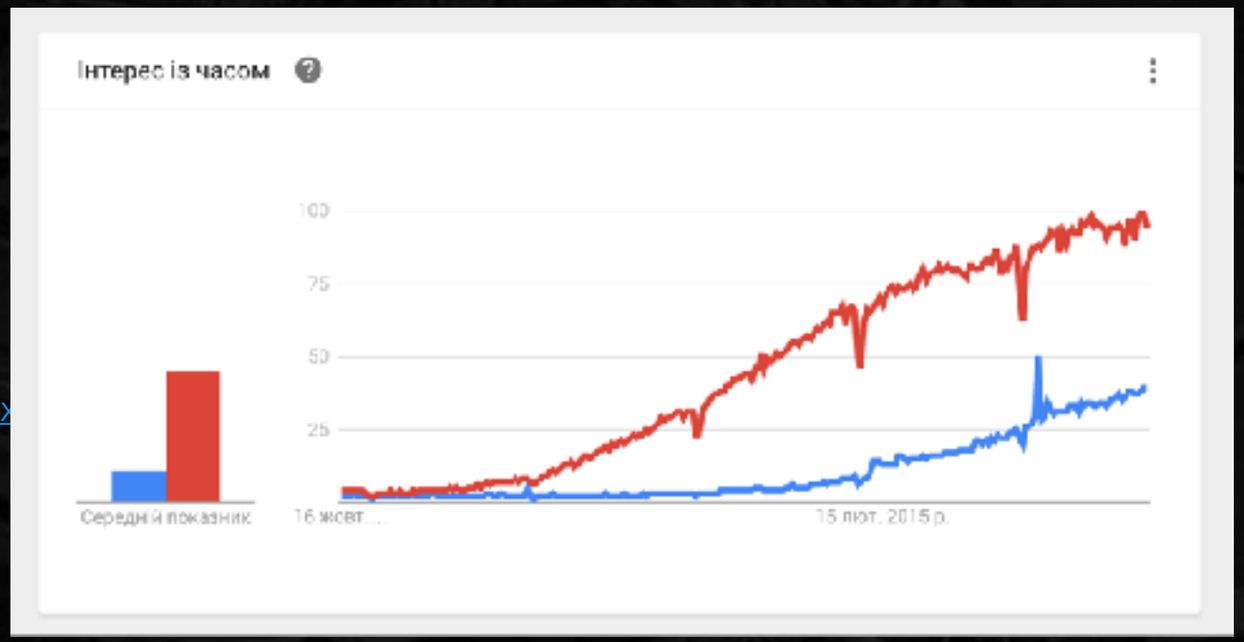
1996 — LiveScript/ECMAScript/JavaScript (Brendan Eich)
1997 — Dynamic HTML
1997 — ES v1.0
1998 — ES v2.0
1999 — XMLHttpRequest
1999 — JS v3.0
2001 — JSON, a JavaScript-based data exchange format
2005 — Ajax, browser-based desktop-class applications
2006 — jQuery, helping with DOM manipulation
2008 — V8, proving JavaScript can be fast
2009 — Node.js, implementing JavaScript on the server
2009 — ES v5.0
2010 — AngularJS
2011 — ES v5.1
2013 — ReactJS
2014 — HTML5
2015 — ES v6.0 (ES2015)
2016 — ES2016
2016 — Angular2

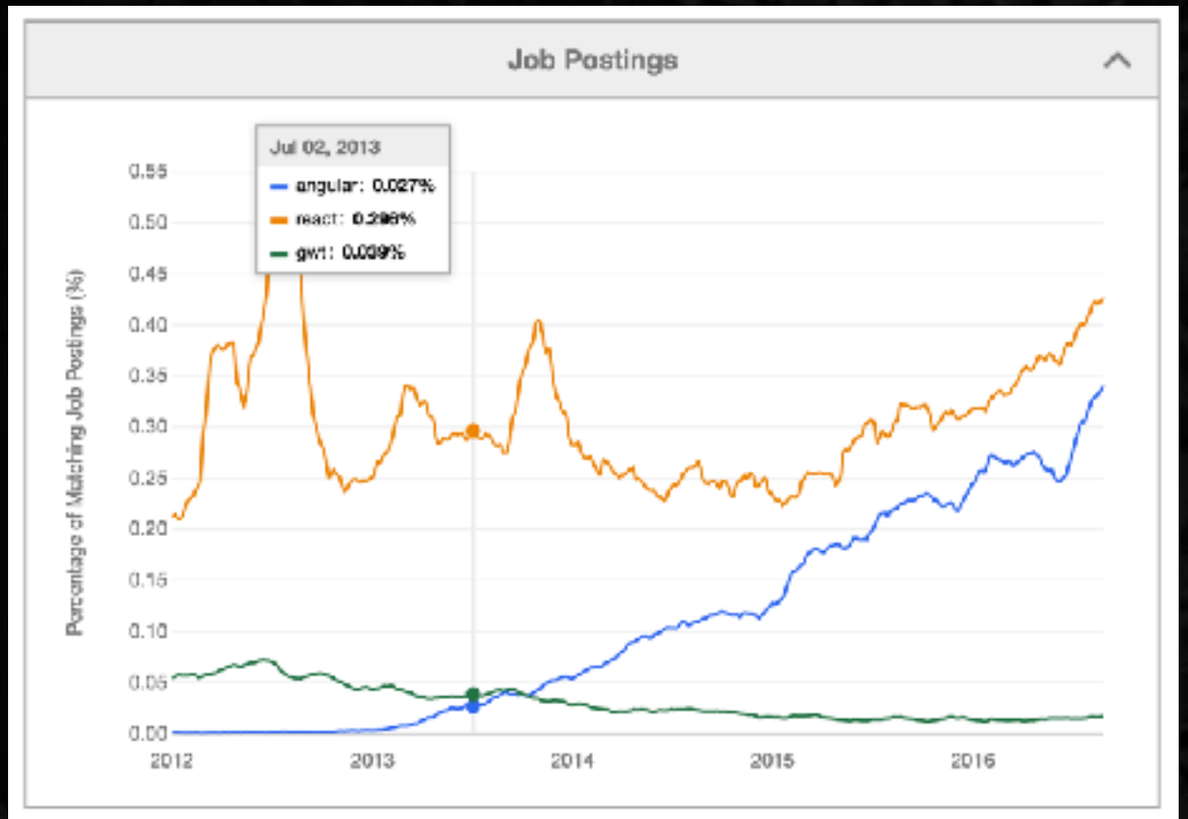
jQuery

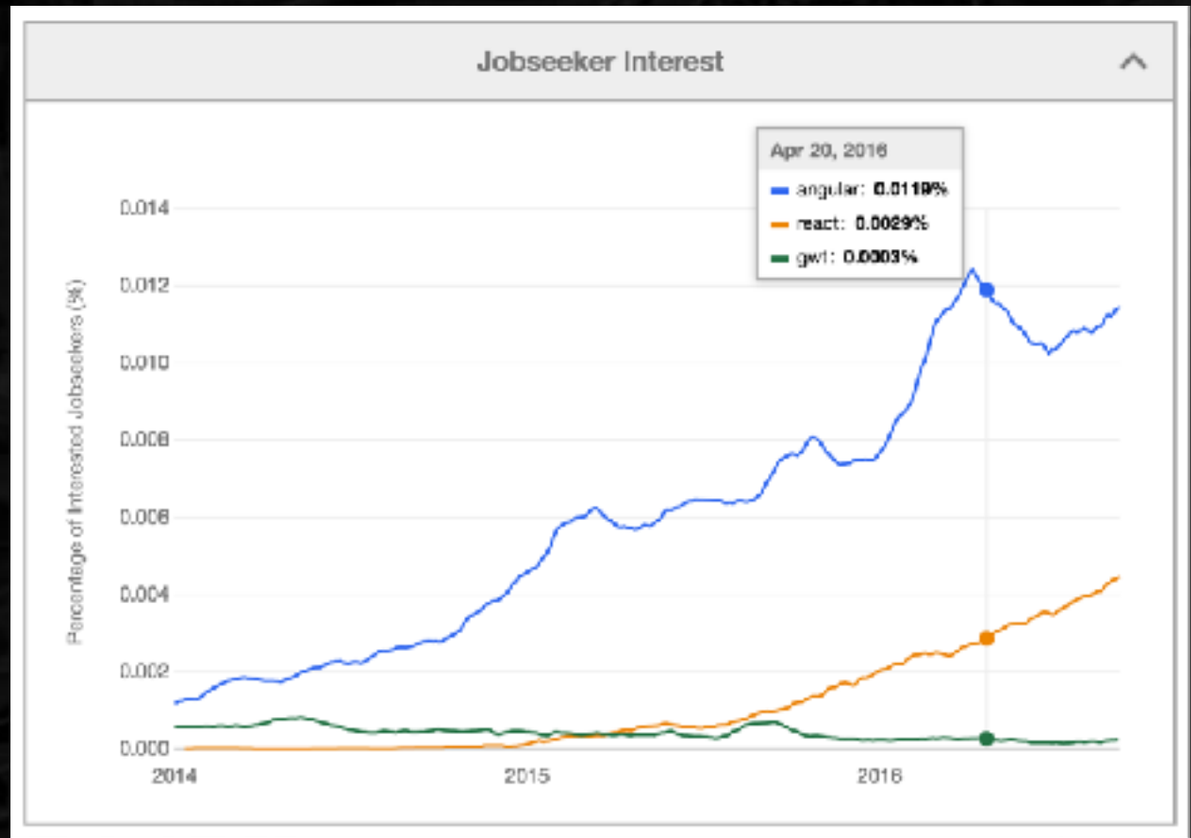
AngularJS vs ReactJS

Angular 1 — 52.6k stars
Angular 2 — 17.1k stars
React — 51.3k stars

https://www.google.com.ua/trends/explore?q=%2Fm%2F012l1vx
http://www.indeed.com/jobtrends/q-angular-q-react-q-gwt.html



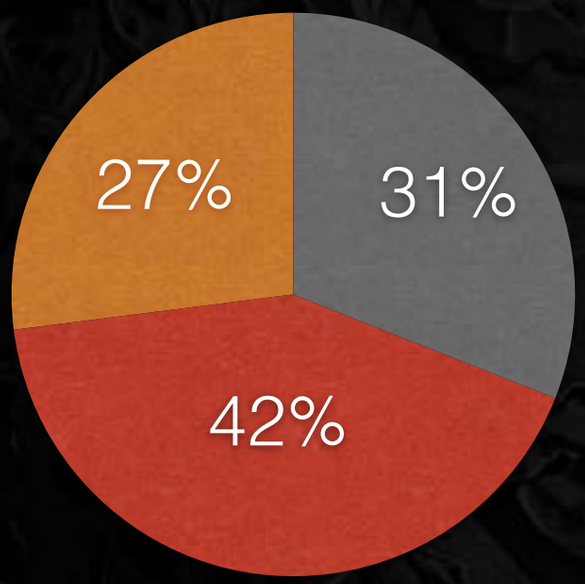




Baba Vanga

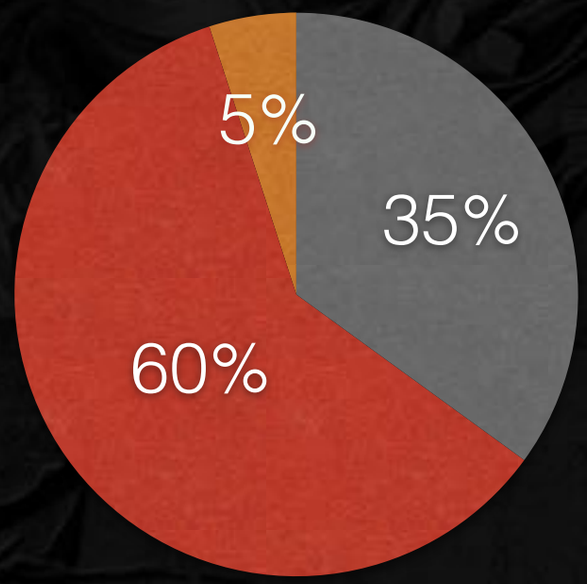
- React
- Angular 2
- Angular 1

2016



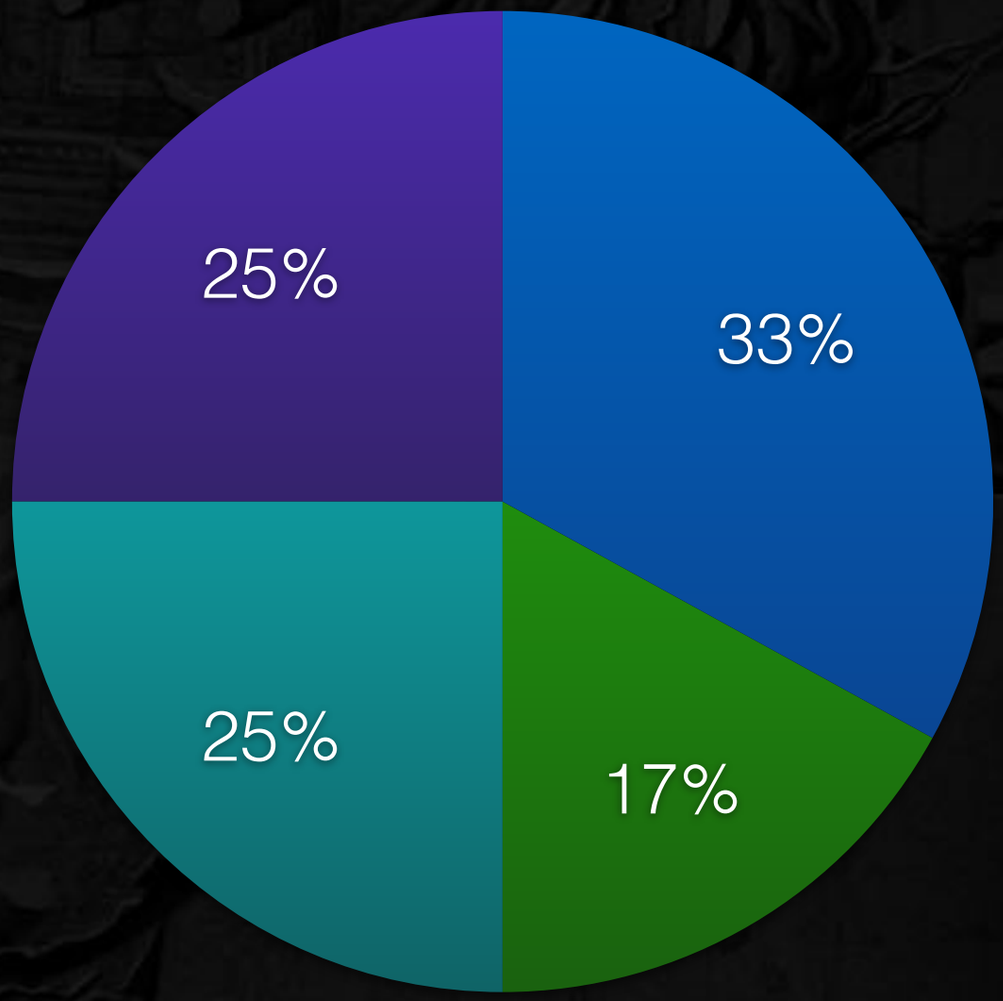
- React
- Angular 2
- Angular 1

2017



What we do

- 1/3 (33%) Planning;
- 1/6 (17%) Writing programs;
- 1/4 (25%) Component testing;
- 1/4 (25%) System testing.



Angular 2

Google

15 Sep 2016 — Final Release

22 Oct 2016 — KRVR Release!

Angular 2: Links

- https://angular.io/
- https://github.com/angular/angular/
- https://angular.io/styleguide
- https://github.com/angular/angular-cli

Angular 2: Overview

1. TypeScript and Decorators
2. Components
3. Observables
4. Dependency injection
5. Routing
6. Change detection strategies
7. (Forms)

CLI tool for Angular2

```
> npm install -g angular-cli
> ng --help

> ng new PROJECT_NAME
> cd PROJECT_NAME
> ng serve

> ng g component my-new-component
> ng g service my-new-service
> ng g directive my-new-directive
> ng g interface my-new-interface
> ng g enum my-new-enum
```

https://github.com/angular/angular-cli

Organize File Structure

```
/src
    app/
        shared/
            todo.service.ts
            todo.service.spec.ts
            todo.ts
        todo-list/
            todo-list.component.ts
            todo-list.component.html
            todo-list.component.css
            todo-list.component.spec.ts
        todo-details/
        …/
        app.component.ts
        app.component.spec.ts
assets/
    imgs/
    fonts/
index.html
main.ts
```

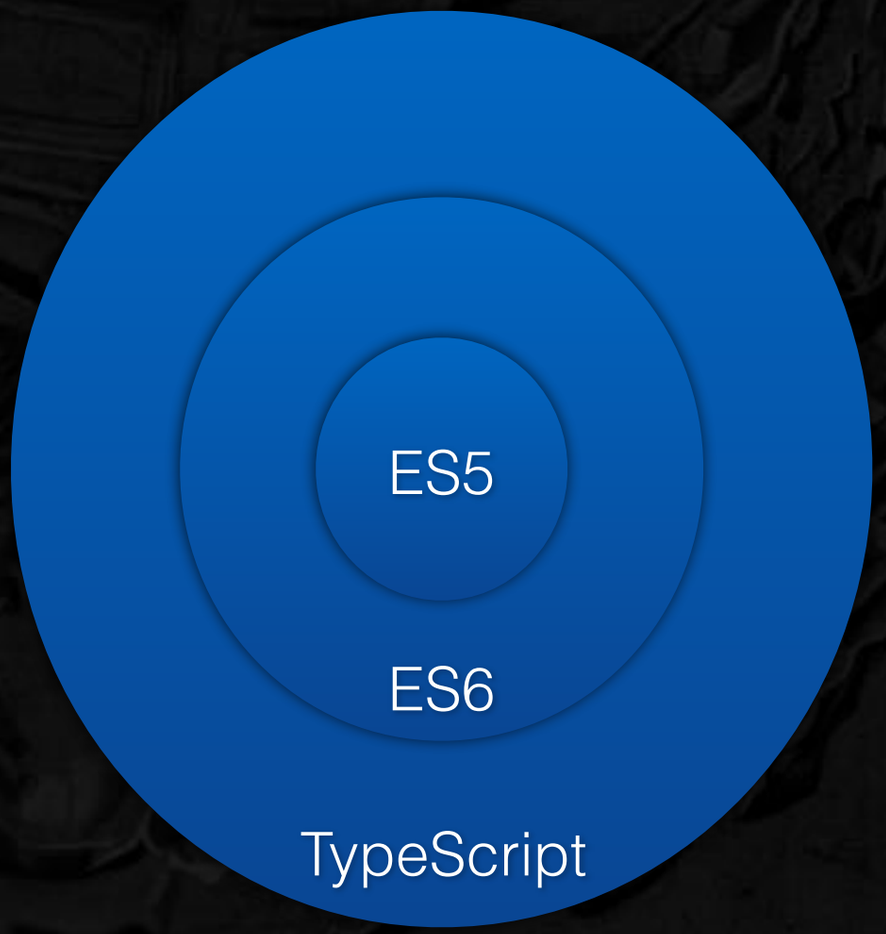
https://angular.io/styleguide

TypeScript

TypeScript

ES6:
— classes
— modules
ES7:
— decorators

TypeScript:
— types
— annotations




```
> npm install -g typescript
```

TypeScript

- JavaScript's types also exist in TypeSctipt
- TypeScript also adds **enum, any** & **void** (like **undefined**)
- **Interface** allows for custom, abstract types
- Function Signatures can be typed Using **Interfaces**
- **Class**es also define types
- *If it walks like a duck, it is a duck, types with the same shapes are compatible*

TypeScript: Simple types

- let a = 123; // Number
- let b: number = 123; // Number

the same other types (boolean, string and objects etc)

- let a1: string[] = []; // Array of Strings (define empty array)
- let a2: string[]; // Array of Strings (undefined)
- let a3: Array<string> // Array of Strings (undefined)
- let a4 = [“a”, “b”]; // Array of Strings

TypeScript: Interface

```ts
interface User {
  name: string;
}

class UserModel {
  constructor(public name: string, private age?: number) {}
}

let u: User = { name: 'foo' };

u = new UserModel(‘bar');


function useUser(user: UserModel) {
  console.log(user.name);
}
```

TypeScript: Parameters

optional parameter

```
interface User {
  name: string;
}

class UserModel {
  constructor(public name: string,
    private age?: number,
    private city? = 'Kyiv') {}
  }

  let u: User = { name: 'foo' };

  u = new UserModel(‘bar');


function useUser(user: UserModel) {
  console.log(user.name);
}
```

optional parameter
with predefined value

TypeScript: Functions

```ts
interface CallbackForUser {
  (userName: string, age: number): number;
}

class UserModel {
  constructor(public name: string, private age?: number, private city? = 'Kyiv') {}

  doSome(cb: CallbackForUser) {
    cb(this.name, this.age);
  }
}

let u: UserModel = new UserModel('bar');

// 1
u.doSome((name: string, age: number) => {
  console.log(`User ${name} is ${age} years old`);
  return age * 100;
});

// 2

let cb: CallbackForUser = (name: string, age: number) => {
  console.log(`User ${name} is ${age} years old`);
  return age * 100;
};
u.doSome(cb);
```

Angular 2 Components

Component

```
<app-hello-world>

  <app-header></app-header>

  <app-user-list>
    <app-user-item></app-user-item>
    <app-user-item></app-user-item>
    <app-user-item></app-user-item>
  </app-user-list>

  <app-add-user-form></app-add-user-form>

  <app-footer></app-footer>

</app-hello-world>
```

Data Binding

Data
[Parent->Child]

App

[items]="someItems"

List

List

[item]="someItem"

Details

Details

Event
(Child->Parent)

App

(event)="someEvent()"

List

List

(event)="someEvent()"

Details

Details

Decorators

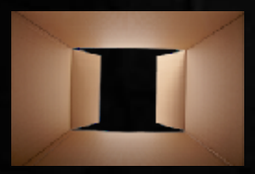
- Decorators — functions that operate on a “target”
- “Target” are classes, methods, properties and parameters
- Decorators invoked with leading **@** like **@Component()**
- Angular 2 decorators always use training brackets, like **@Inject()**
- Decorators do not get follower by **;**

Component

```
// > ng g component hello-world

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

@Component({
  selector: 'app-hello-world',
  template: ‘<p>Hello, {{ title }}</p>’
})
export class HelloWorldComponent {
  title: string;

  constructor() {
    this.title = 'World';
  }

}

// <app-hello-world></app-hello-world>
```

Component: Input/Output

```
// <app-counter [title]=“someTitle” (result)=“onResult()”></app-counter>

import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-counter',
  template: `
    <div>
      <h2>{{ title }}</h2>
      <span>{{ counter }}</span>
      <p>
        <button (click)="inc()" class="inc">inc</button>
      </p>
    </div>

})
export class CounterComponent {

  @Input()
  title: string = '';

  counter: number = 0;

  @Output()
  result: EventEmitter<number> = new EventEmitter();

  inc() {
    this.counter++;
    this.result.emit(this.counter);
  }

}
```



Two-Way Data Binding

Combines the input and output binding into single notation using the **ngModel** directive.

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

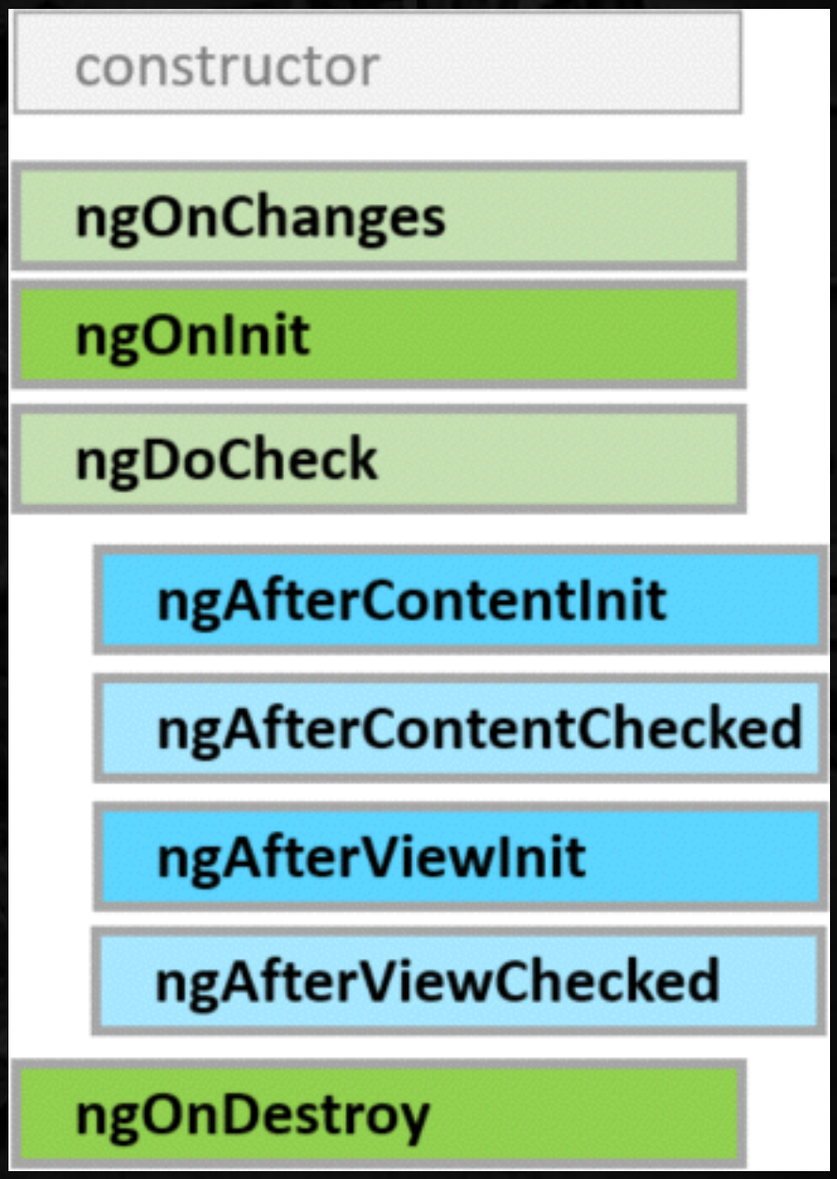
[()] = BANANA IN A BOX

https://angular.io/docs/ts/latest/guide/template-syntax.html

= "someValue"

Component Lifecycle hooks

```
import { Component, Input, OnInit } from '@angular/core';
import { TodoService } from '../shared/todo.service';
@Component({
  selector: 'app-hello-world',
  template: `<p>Hello, {{ title }}!</p>`
})
export class HelloWorldComponent implements OnInit {

  @Input()
  title: string;

  constructor(private todoService: TodoService) {
    // if (!this.title) {
    //   this.title = 'World';
    // }
    console.log('constructor', this.title); // undefined
  }

  ngOnInit() {
    if (!this.title) {
      this.title = 'World';
    }
    console.log('ngOnInit', this.title);
  }

}
```



Template Syntax

```
<ul>
  <li>{{title}}</li>
  <li *ngFor="let menu of menus" [ngClass]="{'active': menu === activeMenu}"><a
routerLink="{{menu.link}}" (click)="onClick(menu)">{{menu.title}}</a></li>
</ul>

<div *ngIf="currentHero">Hello, {{currentHero.firstName}}</div>

<div [class.hidden]="isSpecial">Hide with class</div>
<div [style.display]="isSpecial ? 'block' : 'none'">Show with style</div>
<div [ngClass]="{'first': true, 'second': true, 'third': false}">...</div>
```

```
<select [(ngModel)]="employee.manager" (ngModelChange)="change($event)">
  <option *ngFor="let manager of managers" [ngValue]="manager">{{ manager.name }}</
option>
</select>

<span [ngSwitch]="toeChoice">
  <span *ngSwitchCase="'Eenie'">Eenie</span>
  <span *ngSwitchCase="'Meanie'">Meanie</span>
  <span *ngSwitchCase="'Miney'">Miney</span>
  <span *ngSwitchCase="'Moe'">Moe</span>
  <span *ngSwitchDefault>other</span>
</span>
```

https://angular.io/docs/ts/latest/guide/template-syntax.html

Observables (ES7)

- Observables open up a continuous channel of communication in which multiple values of data can be emitted over time.

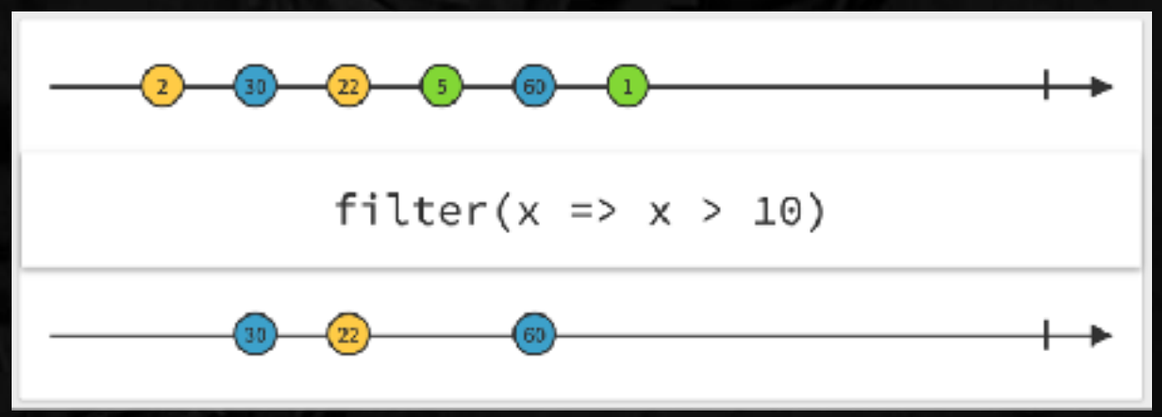
- From this we get a pattern of dealing with data by using array-like operations to parse, modify and maintain data.

- Angular 2 uses observables extensively - you'll see them in the HTTP service and the event system.

http://rxmarbles.com/#filter

Observables: Example

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

import { Http } from '@angular/http';
import './rxjs-operators';
import { Observable } from 'rxjs/Observable';

@Component({
  selector: 'app-test'
})
export class TestSomponent {
  constructor(private http: Http) {}

  getHeroes (): Observable<Hero[]> {
    return this.http.get(this.heroesUrl)
      .map(res => res.json())
      .filter(data => data.age > 18)
      .subscribe((data) => {
        this.data = data;
      });
  }
}
```



Router

```
import { ModuleWithProviders }  from '@angular/
core';
import { Routes, RouterModule } from '@angular/
router';

import { HomeComponent } from './home/
home.component';
import { TodoListComponent } from './todo-list/
todo-list.component';
import { TodoDetailsComponent } from './todo-
details/todo-details.component';

const appRoutes: Routes = [
  {
    path: '',
    component: HomeComponent
  },
  {
    path: 'todo',
    component: TodoListComponent
  },
  {
    path: 'todo/:id',
    component: TodoDetailsComponent
  }
];

export const routing: ModuleWithProviders =
RouterModule.forRoot(appRoutes);
```

```
<a [routerLink]="['/todo/', todo.id]”>{{ todo.title }}</a>
```

```
export interface Route {
    path?: string;
    pathMatch?: string;
    component?: Type<any>;
    redirectTo?: string;
    outlet?: string;
    canActivate?: any[];
    canActivateChild?: any[];
    canDeactivate?: any[];
    canLoad?: any[];
    data?: Data;
    resolve?: ResolveData;
    children?: Route[];
    loadChildren?: LoadChildren;
}
```

Artem Koziar

http://temich.in.ua/

a@temich.in.ua

artko@ciklum.com

artem.Koziar@kantarretail.com

Skype: Isharkichl

Self: http://temich.in.ua/-i/angular2-4kantar.pdf

TNX