Angular for beginners

Three components

- TypeScript
- Rx.JS
- Angular

TypeScript

TypeScript

TypeScript = New Amazing JavaScript

It is a superset of JavaScript.

TypeScript

New key features

- compiling to JS
- naming
- types
- syntax features
- API (polyfills)
- frequent problem

TypeScript

Compiling to JS

- source code compiling to plain JS code
- debugging TypeScript code via source mapping

TypeScript

Naming

```
some-thing.ts # some plain class or interface
some.service.ts # some special class like sirvice, etc...
index.ts # reserved name
```

TypeScript

Types

Static typing and type checking at compile time.

- primitives:
  - boolean
  - number
  - string
  - any (default)
- arrays
- classes / interfaces

TypeScript

Types

```
class C1 {
    private a: number;
    public b: string;
    c: boolean;

    constructor(argument: number,
        private field: string) { }
}

interface ClassInterface { method() }
interface ObjInterface { a: number }

let a: ObjInterface = { a: 1 };
```

TypeScript

Syntax features

Variable declaration

```
let variable: number = 1;
const constant: string;
let array:boolean[];

var some; // bad style
```

TypeScript

Syntax features

String definition

```
// Template Strings (strings that use backticks)
// String Interpolation with Template Strings
let username = 'Tyrone';
let greeting = `Hi ${username}, how are you?`;

// Multiline Strings with Template Strings
let multiline = `This is an example
of a multiline string`;
```

TypeScript

Syntax features

Function definition

```
let add1 = (a: number) => a + 1;
let addN = (a: number, n: number) => {
  return a + n;
}

function some() {} // old syntax, bad style
```

TypeScript

Syntax features

Foreach

```
let list = ['a', 'b', 'c'];

for (let i in list) {
   console.log(i); // "0", "1", "2",
}

for (let i of list) {
   console.log(i); // "a", "b", "c"
}
```

TypeScript

Syntax features

Imports and exports

```
import * as m from "SomeModule";
import * from "SomeModule";
import { some } from "SomeModule";

export const a = 1;
export let b = 2;
export function f1() { }
export class C1 { }
```

TypeScript

Syntax features

index.ts - file for export some from directory

Syntax

```
export * from './some';
export * from './some-other';
```

TypeScript

Syntax features

Decorators

```
@SomeDecorator
someDeclaration
```

TypeScript

API (polyfills)

Array methods

Syntax

```
let arr = [1, 2, 3]

console.log(arr.filter(x => x < 2).toString()); // [1]
console.log(arr.map(x => x + 1).toString()); // [2, 3, 4]
console.log(arr.reduce((a, b) => a + b).toString()); // 6
```

TypeScript

Frequent problem

'this' in TypeScript

```
class Foo {
  x = 3;
  print() { console.log('x is ' + this.x); }
}

let f = new Foo();
f.print(); // Prints 'x is 3' as expected

// Use the class method in an object literal
let z = { x: 10, p: f.print };
z.p(); // Prints 'x is 10'

let p = z.p;
p(); // Prints 'x is undefined'
```

TypeScript

Frequent problem

'this' in TypeScript (fixes)

Use Instance Functions

```
class MyClass {
    private status = "blah";

    public run = () => { // <-- note syntax here
        alert(this.status);
    }
}
let x = new MyClass();
$(document).ready(x.run);
```

TypeScript

Frequent problem

'this' in TypeScript (fixes)

Local Fat Arrow

```
let x = new SomeClass();
someCallback((n, m) => x.doSomething(n, m));
```

TypeScript

Frequent problem

'this' in TypeScript (fixes)

Function.bind

```
let x = new SomeClass();
window.setTimeout(x.someMethod.bind(x), 100);
```

Rx.JS

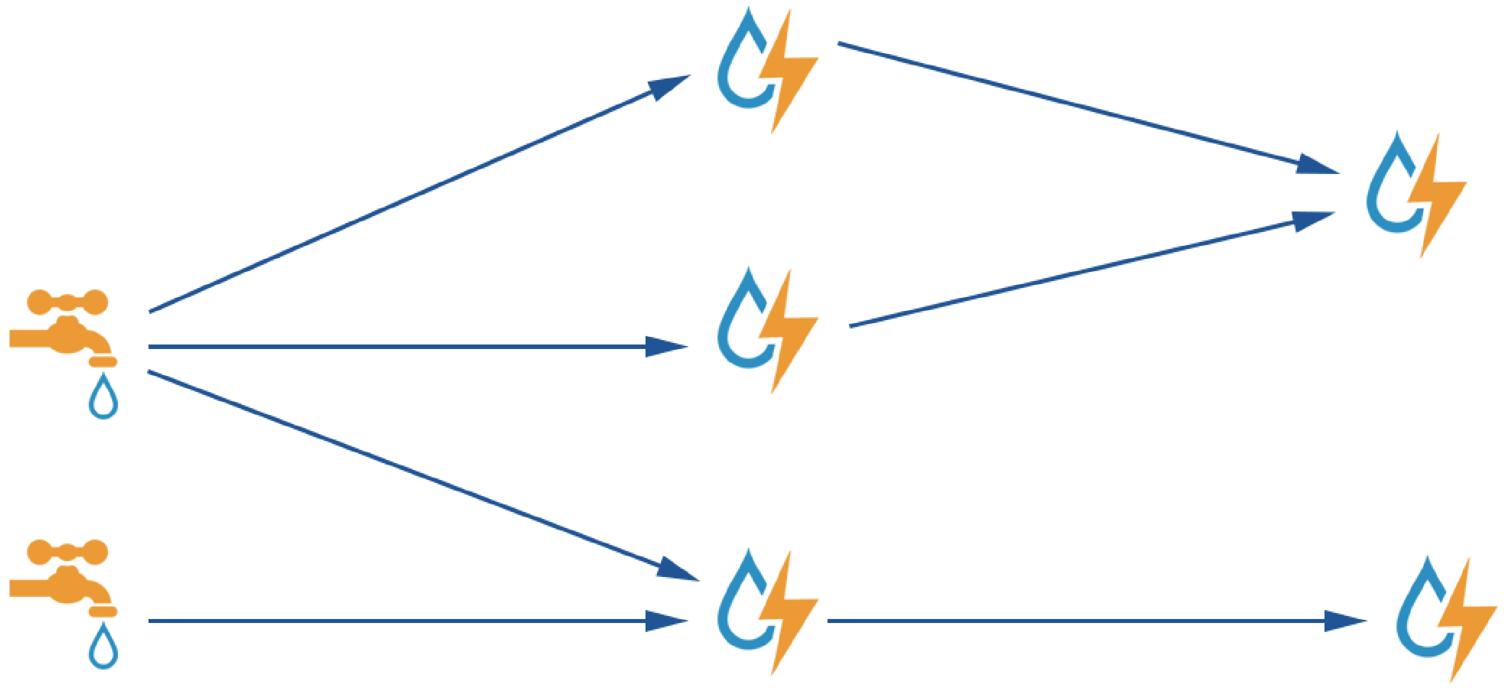
Rx.JS

Conceptions

- reactive programming
- data streams

Rx.JS

Conceptions



Rx.JS

Observable and Subjects

Observable

```
import { Observable } from 'rxjs';

const observable = new Observable(observer => {
  setTimeout(() =>
      observer.next('hello from Observable!'), 1000);
});

observable.subscribe(v => console.log(v));
```

Rx.JS

Observable and Subjects

Subjects

```
import { Subject } from 'rxjs';

const subject = new Subject();

subject.next('missed message from Subject');

subject.subscribe(v => console.log(v));

subject.next('hello from subject!');
```

Rx.JS

Simple flow

```
import { fromPromise } from 'rxjs';

// Create an Observable out of a promise
const data = fromPromise(fetch('/api/endpoint'));
// Subscribe to begin listening for async result
data.subscribe({
 next(response) { console.log(response); },
 error(err) { console.error('Error: ' + err); },
 complete() { console.log('Completed'); }
});
```

Rx.JS

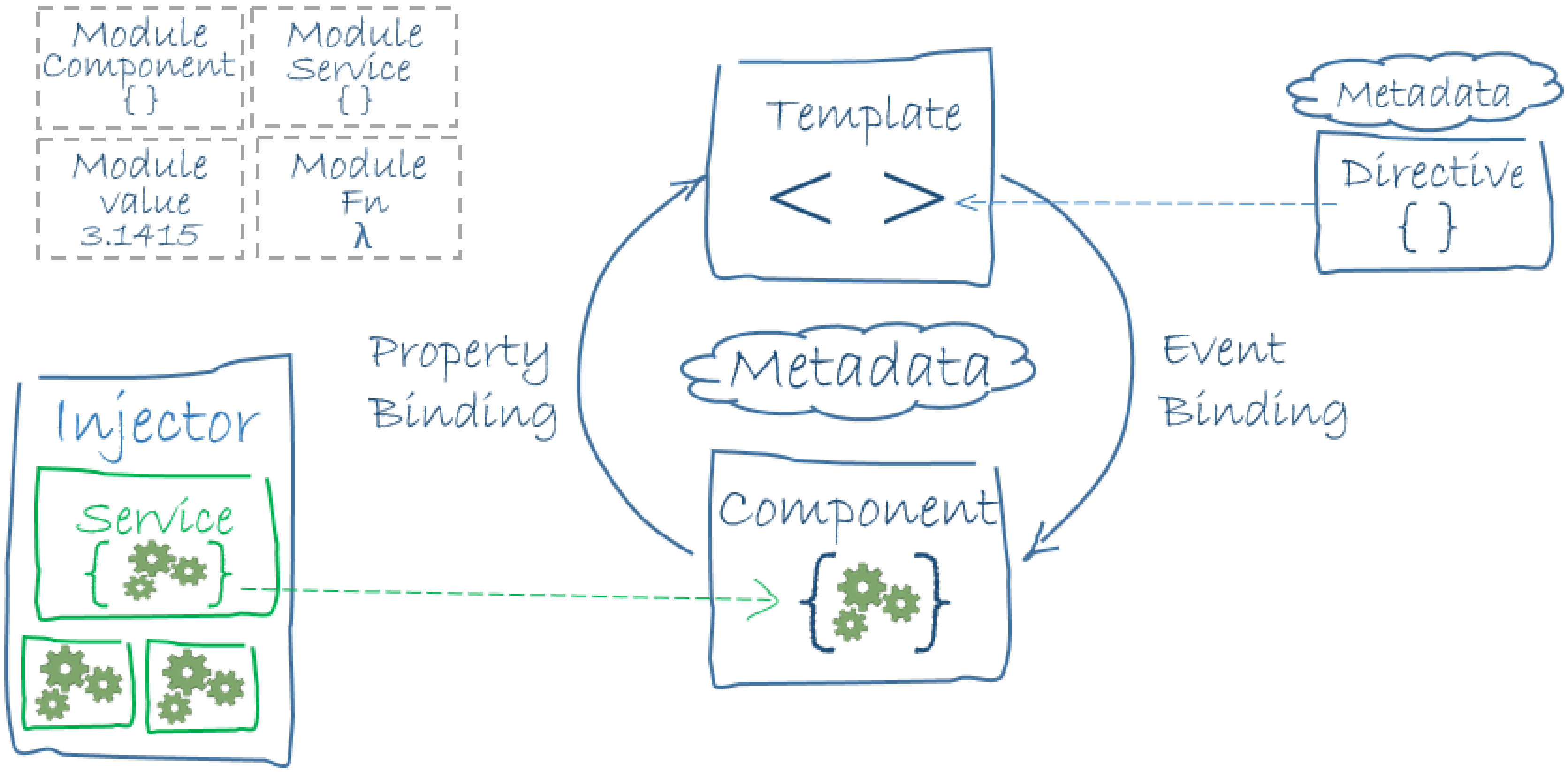
Complex flow

```
import { from } from 'rxjs';
import { filter, map } from 'rxjs/operators';

//emit (1,2,3,4,5)
const source = from([1, 2, 3, 4, 5]);

const example = source
  .pipe(map(val => val + 10))
  .pipe(filter(num => num % 2 === 0));

//output: [12, 14]
const subscribe = example.subscribe(val => console.log(val));
```

Angular

Angular

Architecture



Angular

Module

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Angular

Component

```
import {Component, EventEmitter, Input, OnInit, Output} from '@

@Component({
  selector: 'app-counter',
  templateUrl: './counter.component.html',
  styleUrls: ['./counter.component.css']
})
export class CounterComponent implements OnInit {
  constructor() { }
  ngOnInit() {}

  // ...
}
```

Angular

Component

```
export class CounterComponent implements OnInit {
  @Input() label: string;

  @Output() isEvenEvt: EventEmitter<boolean> = new EventEmitter

  @Output() valueChange: EventEmitter<number> = new EventEmitte
  _value = 0;

  @Input() get value() { return this._value; }
  set value(value: number) {
    this._value = value;
    this.valueChange.emit(this._value);
    this.isEvenEvt.emit(Math.abs(this._value % 2) === 1);
  }

  incr() { this.value++; }
  decr() { this.value--; }
}
```

Angular

Template

```
<app-counter
    label="Counter #1"
    #counter1 [value]="11"
    (isEvenEvt)="indicator1.setTitleByIsEven($event)">
</app-counter>
<app-counter [label]="'Counter #2'"
             #counter2
             [(value)]="counter1.value">
</app-counter>
<hr/>
<app-even-odd-indicator #indicator1></app-even-odd-indicator>
```

Angular

Directive

- Structural directives (NgFor and NgIf)
- Attribute directives (NgStyle and NgClass)

Angular

Directive NgIf

```
<button (click)="show=!show">{{show?'hide':'show'}}</button>
show = {{show}}
<br>
<div *ngIf="show">Text to show</div>
```

Angular

Directive NgFor

```
<li *ngFor="let item of items">{{ item }}</li>
```

Angular

Directive NgClass

```
<some-el [ngClass]="'a b'">...</some-el>
<some-el [ngClass]="['a', 'b']">...</some-el>
<some-el [ngClass]="{'a': 1 < 2, 'b': 2 > -3}"></some-el>
```

Angular

Directive NgStyle

```
<some-element [ngStyle]="{'font-style': styleExp}">
  demo
</some-element>
```

Angular

Directive Pipes

- syntax

```
{{ birthday | date:"MM/dd/yy" }}
```

- build in pipes (DatePipe, UpperCasePipe, LowerCasePipe, CurrencyPipe and PercentPipe)

Angular

Service

```
@Injectable({
  providedIn: 'root'
})
export class WeatherService {
  constructor(private http: HttpClient) { }

  get(): Observable<any> {
    return this.http
      .get<WeatherApiResponse>(url)
      .pipe<Weather>(map(response => new Weather(
        response.query.results.channel.location.country,
        response.query.results.channel.location.city,
        response.query.results.channel.item.condition.temp,
        response.query.results.channel.units.temperature,
        response.query.results.channel.item.condition.text,
      )));
  }
}
```

Angular

Routing

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import {MainPageComponent} from './main-page/main-page.componer
import {AboutPageComponent} from './about-page/about-page.compo

const routes: Routes = [
  {path: '', pathMatch: 'full', redirectTo: 'main'},
  {path: 'main', pathMatch: 'full', component: MainPageComponer
  {path: 'about', pathMatch: 'full', component: AboutPageCompor
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Angular

Project structure (folders)

Folder	Description
/e2e	end-to-end test app
/src	main source folders
/src/app	app module folder
/src/assets	image files and other assets
/src/environments	build configuration options

Angular

Project structure (files)

File	Description
package.json	npm project config
angular.json	angular project config
tsconfig*.json	typescript compiller configs
tslint*.json	typescript codestyle checker config

Angular

Project structure (files)

File	Description
browserlist	configures sharing of target browsers
favicon.ico	favicon file
index.html	main HTML page
main.ts	main entry point
polyfills.ts	polyfill scripts for browser support
styles.css	lists CSS files that supply styles for a project
test.ts	main entry point for unit tests

Angular

Angular CLI

```
npm install -g @angular/cli
ng new my-dream-app
ng generate <some> <some-name>
# ng build
# ng serve --proxy-config proxy.config.json
ng serve
```

Angular

Tools

- Ng-Bootstrap + Bootstrap 4
- Angular Materials
- JSON Server

FIN