Getting started

Setup

- Create an index.html
- Install and configure TypeScript
- Setup the build environment
  - Setup unit testing (Karma, Mocha, Chai, Sinon, Jasmine, Istanbul)
  - Setup end-to-end testing (Protractor, Cypress)
  - Setup building (gulp, webpack)
- Add libraries
  - Core.js: Essential ES6 features, like Promise
  - Zone.js: Angular's change detection works by executing code in a zone
  - RxJS for Observables
  - Polyfills for older browsers
  - All Angular modules (`@angular/core`, `@angular/common/http`, etc)

Recommended: Follow the Angular quickstart guide

Angular CLI

- https://github.com/angular/angular-cli
- Provides a command line interface for working with Angular
- Maintained by the Angular team
- Works with Karma, Jasmine, Istanbul, Protractor and webpack
- Enforces the Angular style guide out of the box

```
npm install --global @angular/cli # or: npm i -g @angular/cli
ng new my-project
cd my-project
npm start
```

Using Angular CLI

- **`ng new <project-name>`**: generates a new application
- **`ng build`**: creates a folder `dist`, containing a distributable of your project
  - **`ng build --prod`** to also uglify and optimize your code
- **`ng serve`**: hosts your project on a web server
- **`ng test`**: run unittests with Karma and Jasmine
  - **`ng test --code-coverage`** to view code coverage statistics
- **`ng e2e`**: run end-to-end tests with Protractor
- **`ng generate`**: generate something
  - `ng generate component confirmation-modal`

Our first component

Create a reusable component, app.component.ts:

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

@Component({
    selector: 'app-root',
    template: '<h1>My application</h1>'
})
export class AppComponent {

}
```

And on your index.html:

```
<app-root></app-root>
```

Bind text on the page

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    title: string = 'My awesome web app';
}
```

app.component.html:

```
<h1>{{title}}</h1>
```

Bind data to DOM properties

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    isVisible = false;
    imageLocation = '/img/angular.svg';
}
```

app.component.html:

```
<p [hidden]="isVisible">Hidden title</h1>
<p [hidden]="!isVisible">Shown title</h1>

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

Handle a mouse click

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    foo() {
        console.log('Hi there!');
    }
}
```

app.component.html:

```
<button (click)="foo()">Print on console</button>
```

Within the parenthesis you can supply any event name.

Looping through data with *ngFor

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    cars = [
        { make: 'Opel', type: 'Astra' },
        { make: 'Porsche', type: '911' }
    ];
}
```

app.component.html:

```
<ul>
    <li *ngFor="let car of cars">{{car.make}} {{car.type}}</li>
</ul>
```

Show/hide with *ngIf

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    showText: boolean = false;

    toggle() {
        this.showText = !this.showText;
    }
}
```

app.component.html:

```
<button (click)="toggle()">Toggle text</button>
<div *ngIf="showText">Hi there!</div>
```

Set CSS classes conditionally

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

@Component({
    selector: 'app-root',
    templateUrl: 'app.component.html'
})
export class AppComponent {
    cars = [
        { make: 'Opel', type: 'Astra', price: 29995 },
        { make: 'Porsche', type: '911', price: 145000 },
        { make: 'Fiat', type: 'Uno', price: 1800 }
    ];
}
```

app.component.html:

```
<ul>
    <li *ngFor="let car of cars" [class.expensive]="car.price > 15000">
        {{car.make}} {{car.type}}
    </li>
</ul>
```

Data binding summary

Data direction	Syntax

One-way
From data to view
	`{{expression}}`
	`[target]="expression"`
	`bind-target="expression"`

One-way From view to data	`(target)="statement"` `on-target="statement`
Two-way	`[(target)]="expression"` `bindon-target="expression`

Additional binding types

One-way, from data to view binding types

Type	Target	Examples
Property	Native element, other component or directive	`<img [src]="imageUrl">` `<edit-user [user]="currentUser">`

Class	`className` property	`<li [class.selected]="selected">`
		`  <a>New item</a>`
		`</li>`

Style	`style` property	`<button [style.background] = "isWarning ? 'red' : 'green'">`
Attribute	Attribute (directly)	`<button [attr.aria-label]="help"></button>`

Recap

- Setup using Node.js, TypeScript and several libraries
- Creating a component with `@Component()` and a template
- Basic data binding with `*ngFor` and `*ngIf`
- Syntaxes:
  - `{{name}}` - double curly brackets interpolation
  - `(keydown)` - parenthesis for event binding
  - `[disabled]` - square brackets for attribute/property binding