

Angular → framework per creare pagine web tanto client e server

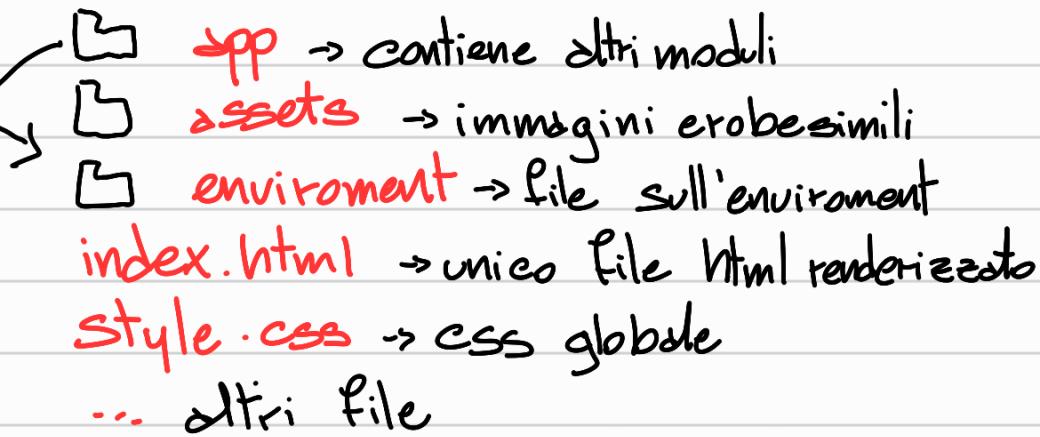
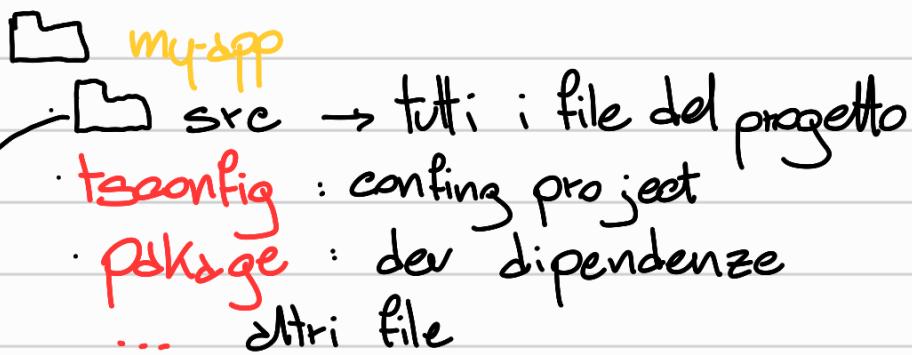
Setup

npm install -g @angular/cli ⇒ scarica modulo node
set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned → abilita a powershell
ng new my-app → crea environment per my app

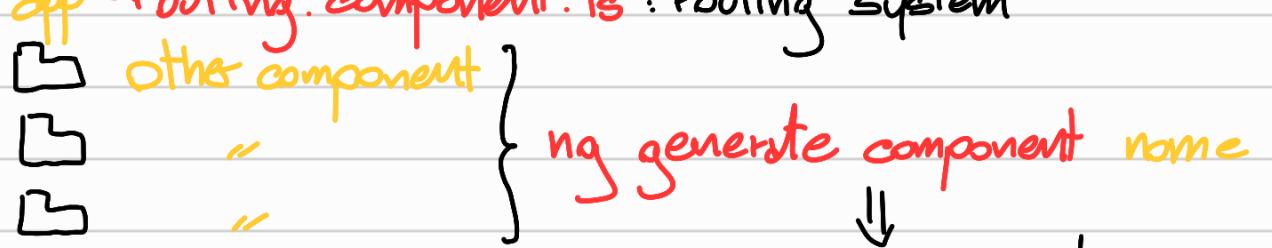
Avvio server

ng serve → avviare il server in debug mode

Struttura progetto (folders)

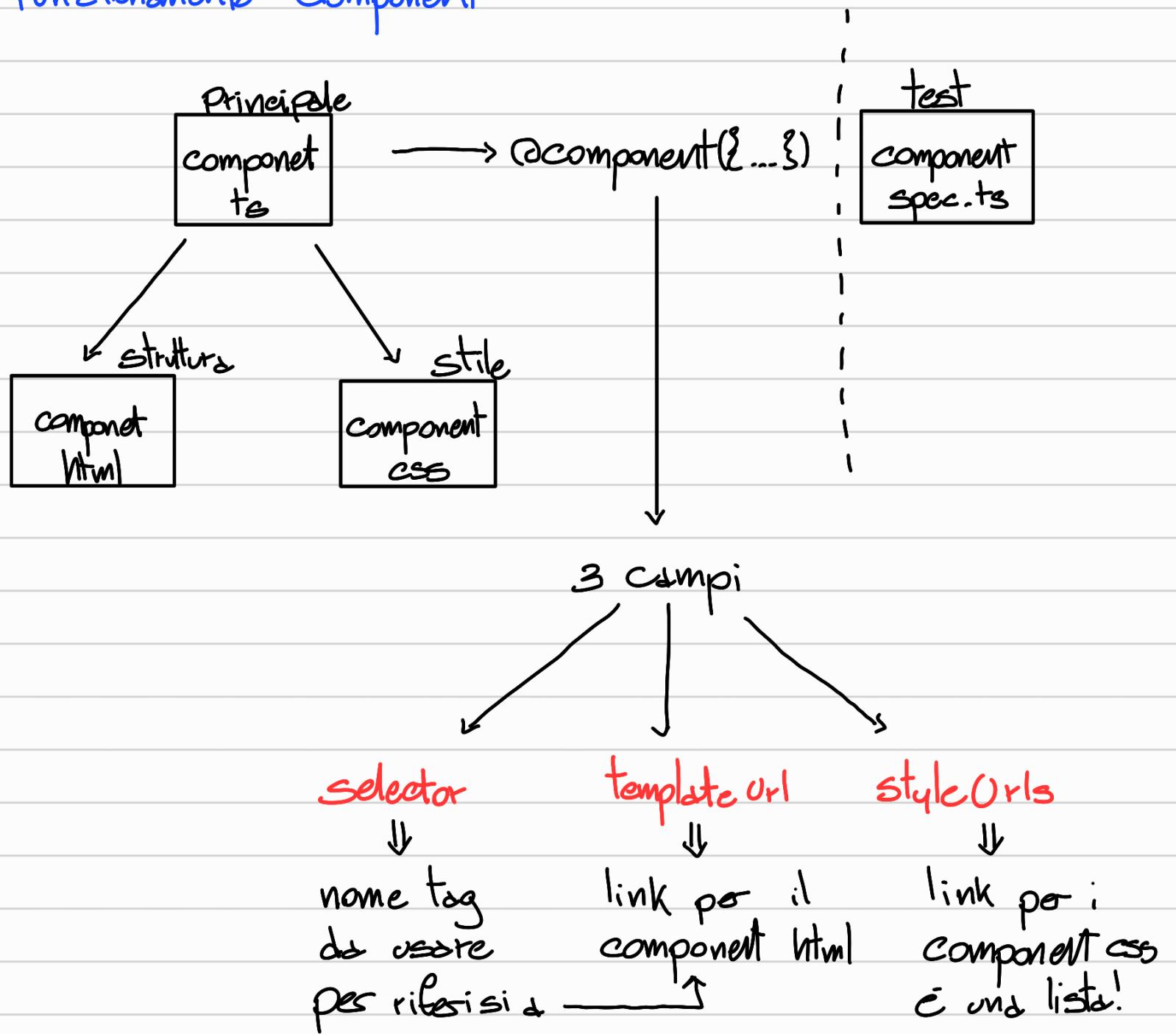


app.component.html : struttura html
app.component.css : css del modulo
app.component.ts : comportamento del componente



crea un componente nome già settato con robe di default

Funzionamento Component



Interface

Per tipi in comune con più componenti bisogna :

1 creare `types` in `app`

2 creare `export interface nomeTipo { ... }`

3 importa lo que escribes:

```
import {nameTipo} from '../types/nameTipo'
```

Interpolation

il file HTML può leggere ciò che sta nel file ts del componente
usando la sintassi `{{ nome Campo }}` e all'interno della
classe dove esso ci `Nome Campo`

Eventi

ts → dichiaro il metodo `foo()` {{

html → dentro il tag setto: <div (click)='foo()' >

tipi di evento metodo da chiamare

ngIf e ng-container

direttiva da mettere su un tag nell'html per poterne cambiare il comportamento in base a delle condizioni

```
<div *ngIf = "mostra" > → mostra è un campo della classe  
    <p> contenuto </p>  
</div>
```

`ng-container` permette di mappare tutti i casi di `ngIf`

`foo()` è una funzione modifica il valore di `mostro`?

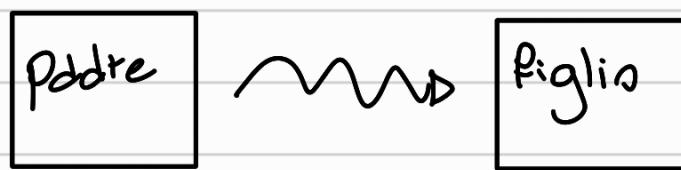
ngFor
permette di scrivere più volte un tag mappato con
questa direttiva

struttura for each!

```
<ul>  
  cli "ngFor = " let single of listElement " > single  
</ul>
```

↓ ↓
 single group
 element

Passaggio tra component padre e component figlio



HTML: <selector-figlio
[book] = "singleBook">
</selector-figlio>
ts: @input(book: Book
= {> Book})

Il passaggio avviene come un "settere un attributo"

Passaggio tra component figlio e component padre



HTML: <selector-figlio
(emit Book) = foo(\$event)>
</selector-figlio>

HTML: <div (click) = "addcast()">
contento
</div>

ts: foo(event: any){
 console.log(event)}

ts: @output() emit Book
= new EventEmitter<Book>()

add cast()

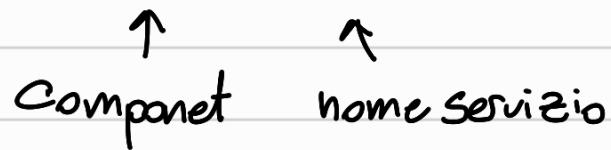
this.emit Book.emit
(this.book)

Questo passaggio avviene come un passaggio di eventi dal figlio al padre

NB: Input, Output, Event Emitter vanno inclusi nell'import di angular/core

Servizi: funzioni esterne che possono essere chiamate da un component. Evitano i passaggi tra componenti

ng generate service books/books



Se voglio usare un servizio devo inserirlo come parametro del costruttore e posso accedere ai metodi del servizio attraverso l'istanza del costruttore, oppure inizializzarlo con new

Dependency injection: fornire un qualcosa che serve per un componente in modo automatico

① Injectable {

②) providedIn: 'root' \Rightarrow si trova sui servizi li rende globali



Se vogliamo avere un servizio locale
solo per i libri (per esempio) dobbiamo
tagliarla e registrare il servizio dove ne
abbiamo bisogno

→ ts component : @ Ng Module {

:
}) providers : [Book service],

Moduli custom : ng generate module books crea un
↓
nuovo modulo, insieme di componenti

dopo averlo creato bisogna registrarlo in app.module.ts
e in books.module.ts bisogna inserire i componenti
che usa in @ Ng Module → declaration : [BooksComponent]
e poi far esportare il nuovo modulo a livello globale con
@ Ng Module → exports : [BooksComponent]

Attenzione : nella declaration bisogna inserire anche
tutti i moduli "figlio", mentre in export
solo il padre

↓
in questo modo è possibile spostare nel file -
system i figli dentro il padre

Routing : gestire le roote della web app.

steps

1. creare file app-routing.module.ts in app
2. popolarlo con @ Ng Module e aggiungerla agli import
di app.module.ts

3. Seguire il quick start nel sito di angular
4. infine in **const routes** bisogna creare le route
formato: {path: '/cart', component: CartComponent}
 ↑ ↑
 route da quale componente
 mappare visualizzare
5. inserite in **app-component.html**:
<router-outlet> </router-outlet> ⇒ componente che si occupa
interamente del routing!

