# Angular

FALICITADOR: WENDELL ANTILDES

# Angular

Angular é um framework mantido pelo Google;

Utiliza HTML como template para construir views;

Faz ligação dos dados através de diretivas;

Injeção de dependência e inversão de controle nativos;

Funciona bem em projetos com arquitetura Web/Web Services;

Usa Typescript nas versões mais atuais;

Normalmente utilizado para desenvolver SPAs (Single Page Applications);

Projeto bem ativo;

# Angular

Traz qualidade e robustez em termos de estrutura de código;

É de grátis;

Põe ênfase na separação de responsabilidades**;**

Possibilidade de criar aplicativos mobile através do Cordova, Ionic;

Possibilidade de criar aplicações Desktop;

Desenvolvimento multiplataforma;

# Site

https://angular.io

# Versões

| Versão | Descrição |
| --- | --- |
| Angular 1.x (AngularJS) | Javascript, Arquitetura MVC |
| Angular 2.x | Reescrita da aplicação em TypeScript, quebra de compatibilidade, Arquitetura Componentes/Serviços |
| Angular 4.x | Redução do tamanho do framework, mais veloz, Angular Universal |
| Angular 5.x | Performance, melhorias |

# Quem está usando Angular?

Microsoft Customers

Rockstar Games

YouTube TV

VMware Open Source Software

NBA.com

Udacity

AIESEC

Citibank Customer Service

Google Express

# História da web

# Páginas estáticas

```
<html>
  <body>
    <h1>Mattias homepage</h1>
    <p>Welcome to my homepage</p>
  </body>
</html>
```

# Geradores de páginas

```
<html>
<body>
<h1><?php $name="Mattias"; echo $name ?> homepage</h1>
<p>Welcome to my homepage</p>
</body>
</html>
```

# Javascript, DOM, JQUERY

```
<html>
<body>
<h1></h1>
<p>Welcome to my homepage</p>
<script>
    $("h1").innerHTML("Mattias Homepage");
</script>
</body>
</html>
```

# Jquery - Vantagens

JQuery cresceu como uma ótima ferramenta para manipulação DOM;

Diminui a quantidade de código;

# Jquery - Desvantagens

Código espaguete sem estrutura;

Seletores criam alto acoplamento;

Não possui nível suficiente de abstrações;

```
$('#someId').find('some-class').append($('<span>').class('foo'))
```

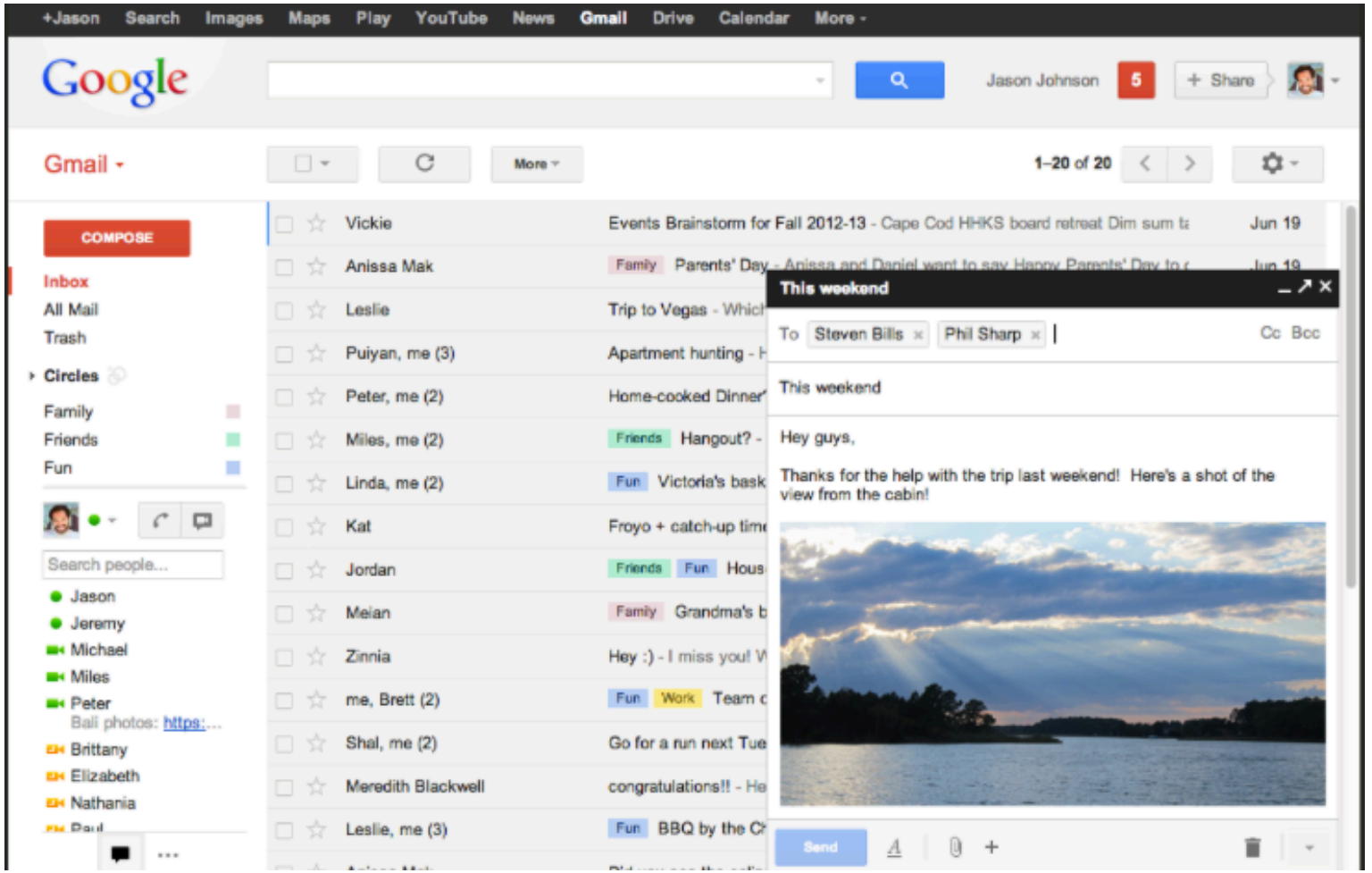
# Jquery

Single view – OK;

Múltiplas views – OK;

Múltiplas views aninhadas – Misericórdia. (Muitos webapps pertencem a essa categoria);

# WEB APPS


+Jason Search Images Maps Play YouTube News Gmail Drive Calendar More
Google
Jason Johnson 5 + Share
Gmail
More
1–20 of 20
COMPOSE
Inbox
All Mail
Trash
Circles
Family
Friends
Fun
Search people...
Jason
Jeremy
Michael
Miles
Peter
Brittany
Elizabeth
Nathania
Vickie Events Brainstorm for Fall 2012-13 - Cape Cod HHKS board retreat Dim sum ta Jun 19
Anissa Mak Family Parents' Day
Leslie Trip to Vegas
Puiyan, me (3) Apartment hunting
Peter, me (2) Home-cooked Dinner
Miles, me (2) Friends Hangout?
Linda, me (2) Fun Victoria's bask
Kat Froyo + catch-up tim
Jordan Friends Fun Hous
Meian Family Grandma's b
Zinnia Hey :) - I miss you!
me, Brett (2) Fun Work Team
Shal, me (2) Go for a run next Tue
Meredith Blackwell congratulations!!
Leslie, me (3) Fun BBQ by the C
This weekend
To Steven Bills Phil Sharp
Cc Bcc
This weekend
Hey guys,
Thanks for the help with the trip last weekend! Here's a shot of the view from the cabin!
Send

# Linguagem

TYPESCRIPT

- TypeScript é um superset de JavaScript;
- Ferramental rico na IDE;
- ECMA6: classes, fortemente tipada;
  - **var** s = 'Hello';
  - s = 123; // Cannot convert 'number' to 'string'.

# Typescript x Javascript

```
class HelloWorld {
  text: string;
  constructor(text: string) {
    this.text = text;
  }
}
let txt = new HelloWorld("Olá mundo!");
console.log(txt);
```

```
var HelloWorld = (function () {
  function HelloWorld(text) {
    this.text = text;
  }
return HelloWorld;
}());
var txt = new HelloWorld("Olá mundo!");
console.log(txt);
```

# Funcionamento clássico

Funcionamento de uma SPA: a aplicação (html, css e javascript) é baixada inteiramente no primeiro acesso;

Renderização no navegador;

# Servidor

Servidor web de arquivos estáticos

# Funcionamento Universal

Renderização das páginas no servidor;

Páginas acessadas primeiramente são entregues primeiro;

Acesso à primeira página da aplicação mais rapidamente;

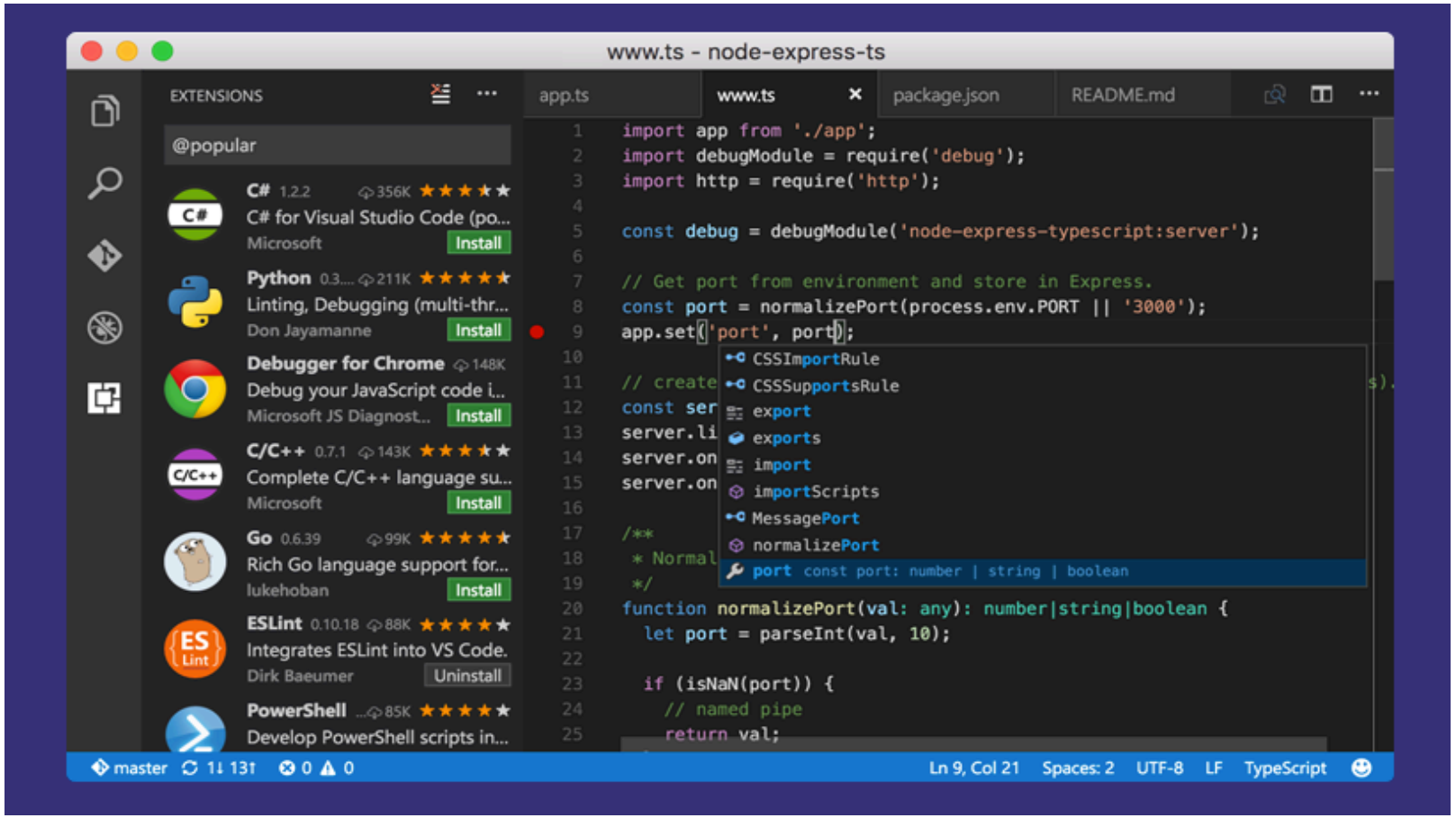
# Servidor Server Side Rendering

Node.js

.NET

# IDE

Visual Studio Code



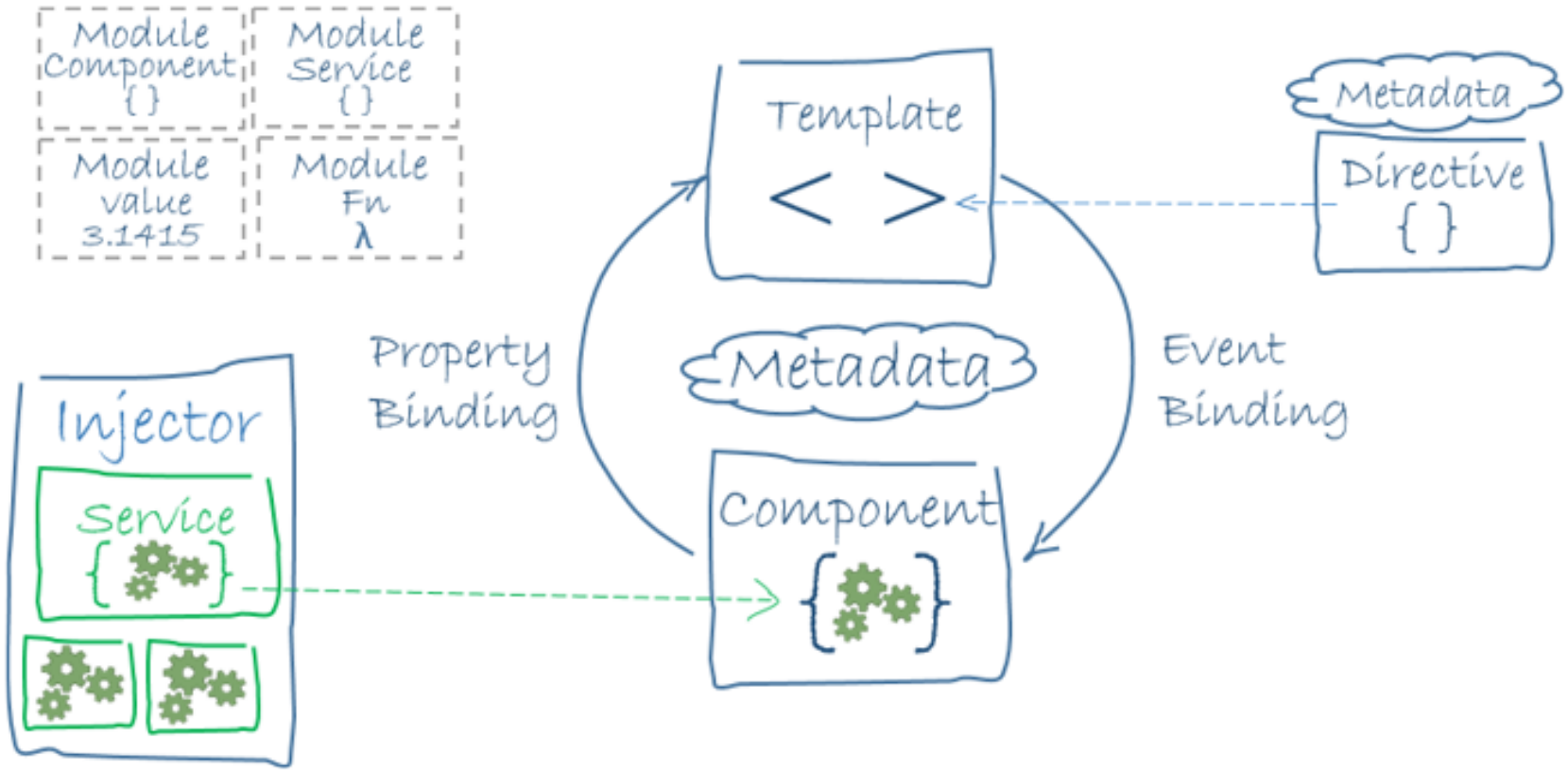
# Uso de comandos

ng serve

ng build

ng g component login

# Arquitetura



# Model

src/app/hero.ts

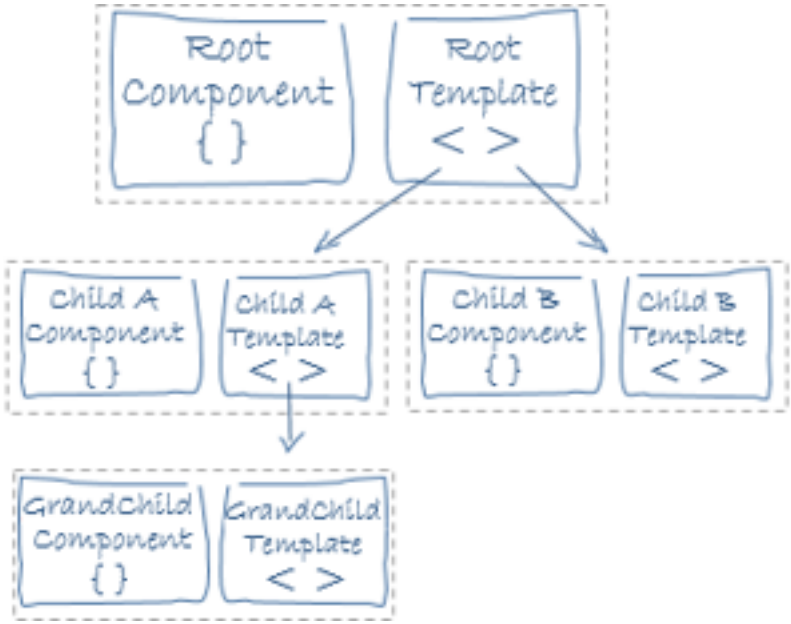
```
export class Hero {
  id: number;
  name: string;
}
```

# Componente

src/app/heroes/heroes.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero';

@Component({
  selector: 'app-heroes',
  templateUrl: './heroes.component.html',
  styleUrls: ['./heroes.component.css']
})
export class HeroesComponent implements OnInit {
  hero: Hero = {
    id: 1,
    name: 'Windstorm'
  };

  constructor() { }

  ngOnInit() {
  }

}
```

# Árvore de Componentes

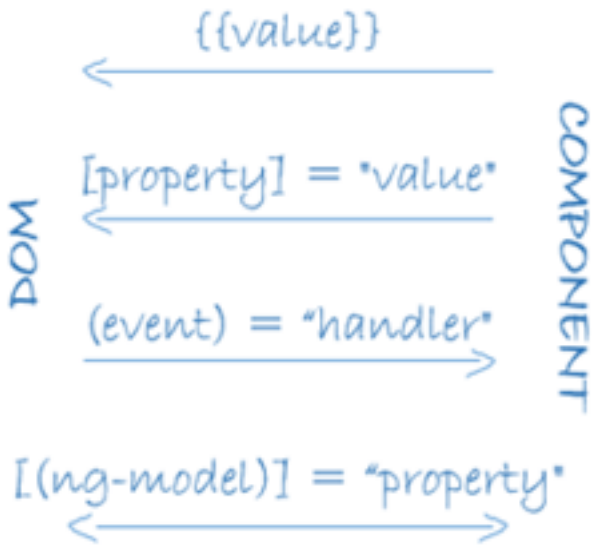


# Template

heroes.component.html (HeroesComponent's template)

```
<h2>{{ hero.name }} Details</h2>
<div><span>id: </span>{{hero.id}}</div>
<div><span>name: </span>{{hero.name}}</div>
```

# Data binding



# Data binding

src/app/hero-list.component.html (binding)

```
<li>{{hero.name}}</li>
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
<li (click)="selectHero(hero)"></li>
```

# Data binding

src/app/hero-detail.component.html (ngModel)

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

# Diretivas

src/app/hero-list.component.html (structural)

```
<li *ngFor="let hero of heroes"></li>
<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

# Serviços

src/app/hero.service.ts (class)

```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

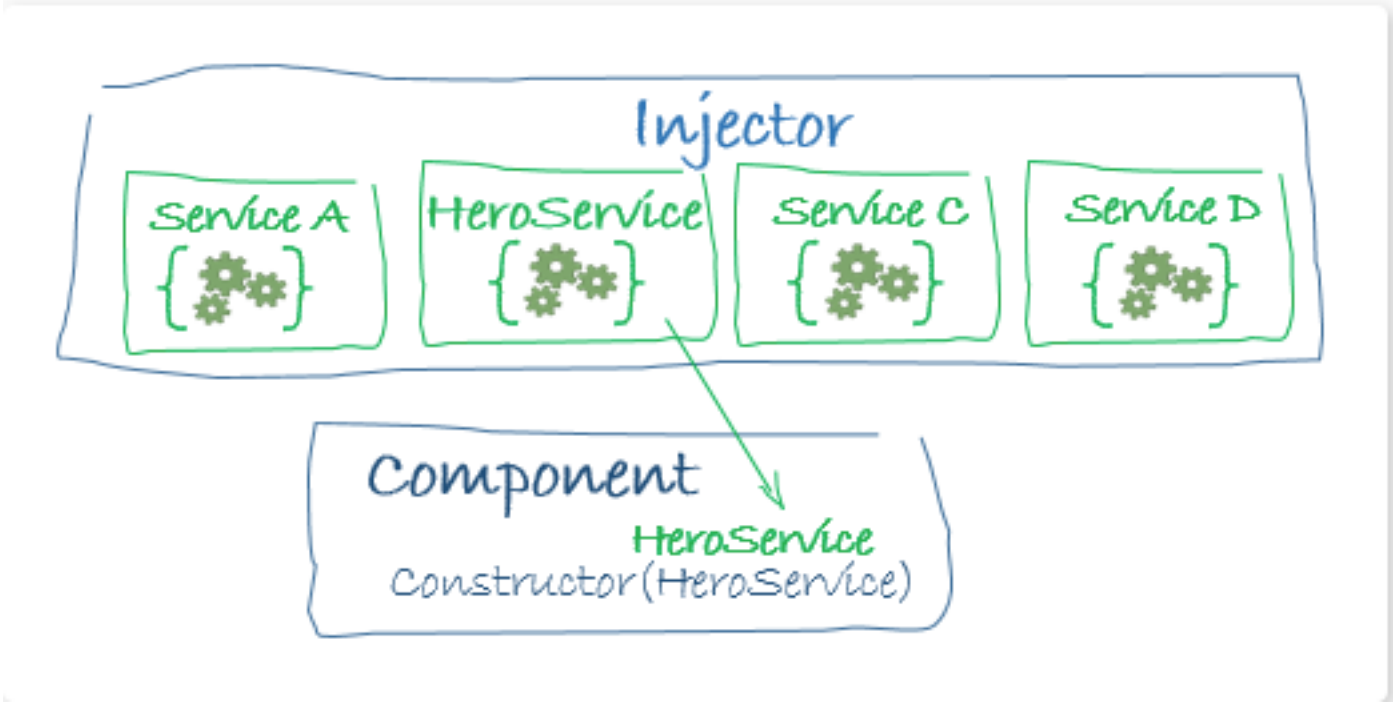
# Injeção de Dependência

# Injeção de Dependência

src/app/hero-list.component.ts (constructor)

```
constructor(private service: HeroService) { }
```

# Injeção de Dependência



# Exemplo de login

Exemplo de código

# Exemplo login em ASP .NET MVC RAZOR

Exemplo

# Exemplo de um loading

Exemplo código;

# Requisição HTTP

```
/** GET heroes from the server */
getHeroes (): Observable<Hero[]> {
  return this.http.get<Hero[]>(this.heroesUrl)
}
```

# Angular x ASP .NET MVC Razor

| | Angular | Asp .Net MVC Razor |
|---|---|---|
| Arquitetura | Componentes | MVC |
| Renderização | Cliente ou Servidor | Servidor |
| Manipulação DOM | Two-way model binding | Jquery ou outras bibliotecas |
| Tipo de aplicação | SPA | MPA |

# Prós e Contras

| Vantagens | Desvantagens |
| --- | --- |
| Reusabilidade | Curva de aprendizado |
| Legilibilidade | Verbosidade |
| Manutembilidade | Complexidade |
| Testes | |
| Two-way data binding | |
| Diretivas | |
| Injeção de dependência | |
| Angular Universal | |

# Futuro já começou

Angular

React

Vue.js

Web Assembly

# Referências


https://angular.io

https://angular.io/guide/architecture

https://www.madewithangular.com/categories/angular

https://tableless.com.br/angular-2-vale-pena/

https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-angular-development/

https://technologyconversations.com/2014/07/10/server-vs-client-side-rendering-angularjs-vs-server-side-mvc/