TypeScript

1. Introduction

- TypeScript starts from JavaScript syntax.
- TypeScript compiles to JavaScript.
- It supports static checking.
- It supports interfaces, classes, modules, and type inference.

TypeScript **compiles** to JavaScript, which then runs in browsers, Node.js, or other JavaScript engines.

2. Declaração de variaveis

```
var x = 1; // deprecated because its flaws
let y = 2; // new and strong declaration
const z = 3; // constants

let arr = [10, 20];
let [a, b] = arr; // array destructuring

Let o = { a: 1, b: 'hello', c: 10 }
Let {a, c} = o; // object destructuring -> let a = o.a;  let c = o.c;
```

3. Tipos de variáveis

Type	Example
boolean	let active: boolean = true;
number	let age: number = 20;
string	let name: string = "Ana";
array	let nums: number[] = [1, 2];
tuple	let t: [string, number];
enum	enum Color { Red, Green }
any	accepts any type
void	usually no return
null	absence of value
undefined	not assigned
never	value never occurs

```
// void
const a: void = undefined; // correct
const b: void = 10; // error

// null
const c: null = null; // correct
const d: null = undefined; // error
```

```
// undefined
const e: undefined = undefined; // correct
const f: undefined = null; //error

// never
const g: never = undefined; //error, nothing can be assigned
const h: never = null; //error
const i: never = 10; //error
```

4. Arrow functions

A shorter syntax for functions

```
let sum = (a, b) => a + b;
s = sum(10, 20);

// or

let data = [1, 2, 3];
let s = 0;
data.forEach((x) => s += x);
```

5. Classes

```
class Welcome {
  name: string;

  constructor(name: string) {
    this.name = name;
  }

  say() {
    return `Hello ${this.name}`;
  }
}
```

Keyword	Meaning
class	defines object structure
constructor	runs when object is created
this	current object
extends	inheritance
super	parent class

6. Interfaces

To define the structure of an object

```
interface Author {
  id: number;
  name: string;
  email: string;
}
```

7. Inheritance

The class inherits from another class

```
class Dog extends Animal {}
```

Angular Study Book

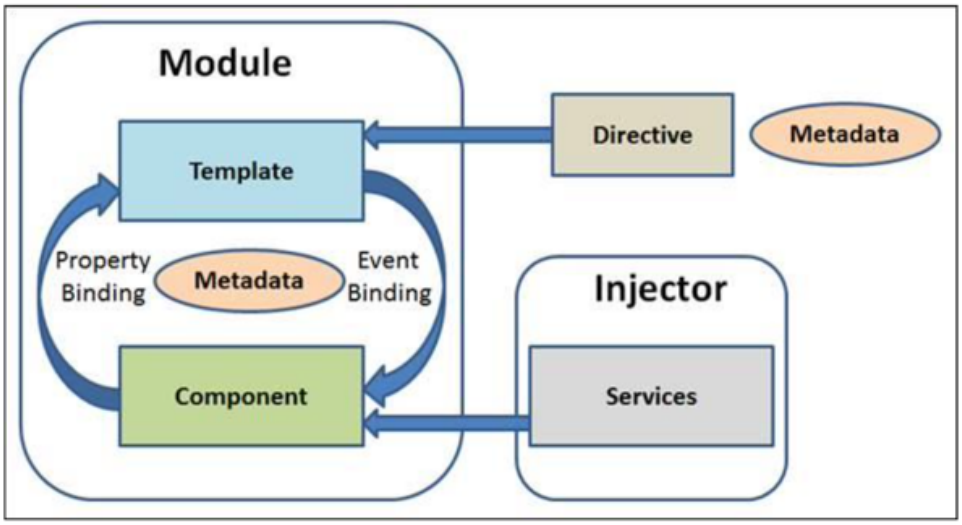
1. Core Angular ideas

Correct idea: a front-end framework based on TypeScript for building web applications.

Angular uses **component-based development** - uses TypeScript
AngularJS used more of an **MVC approach**.

2. Angular architecture

Concept	What it means
Component	UI building block
Template	HTML view of a component
Metadata	Information given through decorators
Data binding	Connects class and template
Directive	Adds behavior to DOM elements
Service	Reusable logic/data
Dependency Injection	Provides services/dependencies
Router	Maps URLs to views



3. Modules

A block of code which is designed to perform a single task

- It can be exported in form of class.
- every Angular application must have at least one module

4. Component

A **component** is: a class with logic + a template for the view + metadata.

```
@Component({
  selector: 'app-authors',
  templateUrl: './authors.html',
  styleUrl: './authors.css'
})
export class Authors {}
```

Correct statements:

- @Component tells Angular that the class is a component.
- selector defines the HTML tag used for the component.
- templateUrl points to the component view.
- The component class contains the logic.

5. Metadata and decorators

Metadata is the way to define how Angular process a class, a method or a property.

In Angular, metadata is defined using **decorators**.

Most important one: `@Component({...}`

How does Angular know that a TypeScript class is a component?

A. Because the file ends in .ts

B. Because the class uses the @Component decorator

C. Because it has an HTML file

D. Because it is inside src/app

6. Template

A template is the **HTML view** of a component.

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

7. Data binding

It is the connection bridge between View and the business logic of the application.

7.1 Four types of data binding

Type	Syntax	Direction
Interpolation	`{{ title }}`	class → template
Property binding	`[src]="imageUrl"`	class → template
Event binding	`(click)="select()"`	template → class
Two-way binding	`[(ngModel)]="name"`	both directions

7.2 Interpolation

Used to display the component value within HTML.

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

What does interpolation do?
A. Displays component data in the template
B. Handles route parameters
C. Imports a service
D. Creates a component

7.3 Property binding

Passes the property's value of a parent component to its child's property.

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

What does `[src]="imageUrl"` do?
A. Calls a method
B. Binds the image `src` property to a component variable
C. Creates a route
D. Defines a service

7.4 Event binding

Used to fire an event when we click on a component's name, or some change occurs in an input component.

```
<button (click)="onSelect(author)">Select</button>
```

What does `(click)` represent?
A. Property binding
B. Event binding
C. Interpolation
D. A route parameter

7.5 Two-way binding

Combines event and property binding in single notation by using ngModel
Important: ngModel needs FormsModule.

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

What does [(ngModel)] do?
A. Only displays data
B. Only listens to clicks
C. Synchronizes data between input and component property
D. Defines a route

8. Directives

A directive changes behavior or structure of DOM elements (extend HTML attributes)

```
<input [(ngModel)]="author.name">
-------------------------------------
<a routerLink="/authors">Authors</a>
---------------------------------------
<ul>
  @for (author of authors; track author.id) {
    <li>{{ author.name }}</li>}
</ul>
```

A directive is used to:
A. Extend/change HTML behavior
B. Compile TypeScript
C. Install Angular CLI
D. Replace package.json

9. Service

A service is used for reusable logic or data, typically a class with a narrow, well-defined purpose.

- data service;
- logging service;
- message service;
- API service.

A service is usually used to:
A. Store reusable logic/data
B. Define only HTML
C. Replace TypeScript
D. Create CSS classes

10. Dependency Injection

Dependency Injection means Angular gives a class what it needs.

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

Dependency Injection helps to:

A. Hard-code dependencies

B. Make components more reusable, maintainable, and testable

C. Remove TypeScript

D. Replace templates

11. Angular Router

Router = maps URL paths to components.

These are stored in `app.routes.ts`, inside `src/app`. This file exports a typed `Routes` constant.

```
// src/app/app.routes.ts
  export const routes: Routes = [
    { path: 'authors', component: Authors }
  ];
```

In main:

```
  import { RouterOutlet, RouterLink } from '@angular/router';
```

In HTML:

```
<nav>
  <a routerLink="/authors">Authors</a>   //nav fixo para todas as paginas
</nav>

<router-outlet></router-outlet>  //a pagina em que estamos
```

What is router-outlet?

A. Placeholder where routed components are displayed

B. A CSS class

C. A TypeScript type

D. A form input

Why use routerLink instead of href?

A. It lets Angular navigate without full page reload

B. It deletes routes

11.1 ActivatedRoute

Used to read route information.

```
constructor(private route: ActivatedRoute) {}
```

ActivatedRoute is used to:

A. Read current route data/parameters
B. Create CSS
C. Install Bootstrap
D. Declare an interface

11.2. Location

```
goBack(): void {
  this.location.back();
}
```

What does this.location.back() do?

A. Goes back in browser history
B. Deletes the component
C. Reloads TypeScript
D. Creates a service

12. Important files

File	Purpose
`src/app/app.ts`	root component logic
`src/app/app.html`	root component template
`src/app/app.css`	root component styles
`src/app/app.routes.ts`	route definitions
`src/styles.css`	global styles
`src/index.html`	main HTML page
`package.json`	dependencies/scripts
`angular.json`	Angular project config
`tsconfig.json`	TypeScript config

13. Bootstrap in Angular

The slides show Bootstrap being added by importing CSS and JS files in index.html.

One simple way to use Bootstrap in Angular is to:

A. Import Bootstrap files in index.html
B. Write Bootstrap inside tsconfig.json

1. A component is composed mainly of:

A. model, database, migration

B. class, template, metadata, styles

C. SQL, table, row

D. package, server, port

2. Which decorator defines an Angular component?

A. @Service

B. @Component

C. @Router

D. @Template

3. What does interpolation use?

A. ( )

B. [ ]

C. {{ }}

D. [( )]

4. What type of binding is this?

```
<button (click)="save()">Save</button>
```

A. Interpolation

B. Property binding

C. Event binding

D. Two-way binding

5.What type of binding is this?

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

A. Event binding

B. Two-way binding

C. Property binding only

D. Interpolation

6. What does this route mean?

```
{ path: 'authors', component: Authors }
```

A. When URL is /authors, show Authors component

B. Create an authors database table

C. Import CSS

D. Run Angular CLI

7.What is router-outlet?

A. Where routed components appear

B. A service

C. A TypeScript type

D. A Bootstrap class

8.What does `:id` mean?

```
{ path: 'authordetails/:id', component: AuthorDetails }
```

A. Static text

B. Dynamic route parameter

C. CSS id

D. Component selector

9.What does `export` do in TypeScript?

A. Makes code available to other files

B. Deletes code

C. Runs a server

D. Creates a template

10. What is an interface?

A. A CSS file

B. A definition of object structure

C. A route

D. A service instance

11. What is Dependency Injection?

A. A way for Angular to provide dependencies like services

B. A way to write HTML

C. A database migration

D. A CSS method

12. What is a service used for?

A. Reusable logic or data

B. HTML only

C. CSS only

D. Browser address bar only

13. Which file usually contains the application routes?

A. `app.routes.ts`

B. `styles.css`

C. `index.html`

D. `package-lock.json`

Angular Services + Django REST Framework

1. Big Picture

```
Django + DRF API  <->  Angular Service  <->  Angular Components
```

Part	Role
Django REST Framework	Creates REST API endpoints that send and receive data, usually JSON.
Serializer	Converts Django model objects to API data and validates incoming data.
Angular Service	Calls the API using fetch() and contains reusable data logic.
Angular Component	Displays data and reacts to user actions.
AsyncPipe	Unwraps asynchronous values like Promises in the template.

2. Web Services and REST

Web service - a service that allows applications/devices to exchange data using web technologies.

JSON/XML - common formats used to transmit data between systems.

REST - Representational State Transfer, an architectural style for web services.

REST stands for:

A. Remote Server Transfer

B. Representational State Transfer

C. Reactive State Template

D. Request Service Type

3. Django REST Framework (DRF)

DRF - Django REST Framework, a Python library used to create REST APIs integrated with Django.

API publication - making endpoints available so clients can request data.

Authentication policies - rules for who can access the API.

4. Django settings.py Configuration

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'corsheaders',
]

MIDDLEWARE = [
    ...
    'corsheaders.middleware.CorsMiddleware',
    ...
]

REST_FRAMEWORK = {
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.AllowAny'
    ]
}

CORS_ORIGIN_ALLOW_ALL = True
```

Setting	Meaning
`rest_framework`	Enables Django REST Framework.
`corsheaders`	Enables CORS handling.
`CorsMiddleware`	Processes cross-origin requests.
`AllowAny`	Allows access to any request/user.
`CORS_ORIGIN_ALLOW_ALL = True`	Allows requests from all origins, useful in development.

Why is CORS important when Angular communicates with Django?

A. Angular and Django may run on different origins

B. It compiles TypeScript

C. It creates database models

D. It replaces serializers

5. DRF Serializers

A serializer converts Django model data into a format that can be sent through the API, usually JSON. It can also validate incoming data before saving.

What is the purpose of a DRF serializer?

A. Convert model/database data into API data format
B. Create Angular templates
C. Style HTML elements
D. Run ng serve

In a serializer, what does fields define?

A. The database password
B. The model fields exposed by the API
C. The Angular routes
D. The CSS classes

6. Angular Services

Components should not directly handle things like fetching server data, validating input, or logging. It should delegate those tasks to services.

Component	Service
Presents UI and handles data binding.	Contains reusable logic and data operations.
Calls methods when user clicks buttons.	Calls the server/API using fetch().
Should stay focused on user experience.	Can be reused by multiple components.

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

@Injectable({
  providedIn: 'root'
})
export class AuthorService {
  private baseUrl: string = 'http://localhost:8000/ws';
}
```

Why should a component use a service?

A. To delegate logic like fetching data from the server
B. To replace its HTML template
C. To compile TypeScript manually
D. To remove all routing

Which decorator is used to define an injectable Angular service?

A. @Component
B. @Injectable
C. @DirectiveOnly
D. @RouterOutlet

What does providedIn: 'root' mean?

A. The service is available globally in the app

B. The service is only CSS

C. The service is deleted after one request

D. The service becomes a route

9. Dependency Injection

Dependency Injection means Angular provides a service/dependency to a component.

```
import { inject } from '@angular/core';
import { AuthorService } from '../author-service';

export class Authors {
  authorService: AuthorService = inject(AuthorService);
}
```

What does inject(AuthorService) do?

A. Gets an instance of AuthorService through Angular DI

B. Creates a Django serializer

C. Converts JSON into XML

D. Creates a CSS class

What does an async function usually return?

A. A Promise

B. A CSS class

C. A Django model

D. A route path

Why use JSON.stringify(au)?

A. To convert the author object into a JSON string for the request body

B. To convert CSS to HTML

C. To create an Angular service

D. To read route parameters

What does authors$ | async do?

A. Resolves asynchronous data in the template

B. Deletes the list

C. Creates a Django view

D. Converts CSS to JSON

What does route.snapshot.params['id'] read?

A. The id from the current route URL

B. The author email

C. The CSS class

D. The Django settings

What does +id do in TypeScript?

A. Converts id to a number

B. Converts id to a string

C. Deletes the id

D. Creates a Promise

AuthorService in Angular is the same as a REST web service.

A. True

B. False

Extra MCQ Practice

Which status code means invalid data was sent?

A. 201

B. 204

C. 400

D. 404

Which status code means successful response with no body?

A. 201 CREATED

B. 204 NO CONTENT

C. 400 BAD REQUEST

D. 404 NOT FOUND

Which one is Django/DRF code?

A. @api_view(['GET'])

B. @Injectable({ providedIn: 'root' })

C. authors$ | async

D. inject(AuthorService)

Which one is Angular code?

A. inject(AuthorService)

B. Author.objects.all()

C. AuthorSerializer(authors, many=True)

D. Response(serializer.data)

What does data.json() do?

A. Reads the response body as JSON

B. Creates a JSON file

C. Deletes the response

D. Creates a Django model