



TypeScript



ANGULAR

Basics



TypeScript

Overview

TypeScript!

Browser engines only can interpret pure JavaScript

JavaScripts' specification is called
ECMAScript <YEAR>

TypeScript is "a layer on top" of JavaScript

After compilation, the stuff is gone and
you're left with pure JavaScript



TypeScript

Motivation

But why?

Helps to catch serious and silly mistakes in
your code

Your codebase will become well structured

Your codebase will become almost self-
documenting

You'll also appreciate the improved
autocompletion

But really, why?

JavaScript is dynamically typed

Static types make the code easier to work
with

It will probably not matter if you code on
your own... but imagine a company with
100 employees...

It's a solid tool and having it in your tool
belt won't make harm

Examples of pure JavaScript

```
1  if (0 == "") {
2    // It is! But why??
3  }
4  -----
5  The left operand is of the type Number.
6  The right operand is of the type String.
7  The right operand is coerced to the type Number: 0 == Number('')
8  which results in 0 == 0, which is true
9
10
11 if (1 < x < 3) {
12   // True for *any* value of x!
13 }
14 -----
15 const tempVarA = 1 < x
16 const tempVarB = tempVarA < 3
17 So 1 < x is either true or false.
18 Then the next step is true < 3 or false < 3.
19
20 const obj = { width: 10, height: 15 };
21 // Why is this NaN? Spelling is hard!
22 const area = obj.width * obj.heigth;
23 -----
24 This is a typo that you'll eventually find.
25 Eventually.
```

So...

Since you may use an ugly style in your code (and it would still work), it makes sense to impose some kind of stricter language on your programmers for the sake of readability and maintainability.

TypeScript lets you set the types of your variables, parameters and return-values.


```
const obj = { width: 10, height: 15 };  
const area = obj.width * obj.heighth;
```

Property 'heighth' does not exist on type '{ width: number; height: number; }'. Did you mean 'height'?

```
console.log(4 / []);
```

The right-hand side of an arithmetic operation must be of type 'any', 'number', 'bigint' or an enum type.
type.

But...

the types get erased during compilation to
JavaScript!

No Reflection!



TypeScript

Examples

Primitive Types

JavaScript

There is already a small set of primitive types available

- boolean
- bigint
- null
- number
- string
- symbol
- undefined

Primitive Types

TypeScript

Extends this list with a few more

- any
(allow anything... good for ports and external dependencies)
- unknown
(ensure someone using this type declares it)
- never
(it's impossible that this type may happen)
- void
(function returns undefined or has no return value at all)

Types By Inference

JavaScript

No types

No type inference

TypeScript

Most of the time it will infer the correct type all by itself

```
let helloWorld = "Hello World";
```


```
let helloWorld: string
```

Defining Types

Interfaces

Sometimes it cannot infer the type. Then you have to specify it manually.

```
const user = {  
  name: "Hayes",  
  id: 0,  
};
```



```
interface User {  
  name: string;  
  id: number;  
}
```

```
const user: User = {  
  name: "Hayes",  
  id: 0,  
};
```

Defining Types

TypeScript then can warn you, if you provide an object that doesn't match the interface.

```
interface User {  
  name: string;  
  id: number;  
}
```

```
const user: User = {  
  username: "Hayes",
```

Type '{ username: string; id: number; }' is not assignable to type 'User'.
Object literal may only specify known properties, and 'username' does not exist in type 'User'.

not exist in type 'User'.

```
  id: 0,  
};
```

Parameters

You can use interfaces to annotate parameters and return values to functions.

```
function getAdminUser(): User {  
    //...  
}  
  
function deleteUser(user: User) {  
    // ...  
}
```


Composing Types: Unions

You can create complex types by combining simple ones with unions.

```
type WindowStates = "open" | "closed" | "minimized";  
type LockStates = "locked" | "unlocked";  
type PositiveOddNumbersUnderTen = 1 | 3 | 5 | 7 | 9;
```

```
function getLength(obj: string | string[]) {  
    return obj.length;  
}
```

Typeof

You can learn the type of a variable by using typeof.

```
function wrapInArray(obj: string | string[]) {  
  if (typeof obj === "string") {  
    return [obj];  
  }  
  return obj;  
}
```

(parameter) obj: string

Composing Types: Generics

You can specify generics as well.

```
type StringArray = Array<string>;  
type NumberArray = Array<number>;  
type ObjectWithNameArray = Array<{ name: string }>;
```

Composing Types: Generics

```
interface Backpack<Type> {  
  add: (obj: Type) => void;  
  get: () => Type;  
}  
  
// This line is a shortcut to tell TypeScript there is a  
// constant called `backpack`, and to not worry about where it came from.  
declare const backpack: Backpack<string>;  
  
// object is a string, because we declared it above as the variable part of  
const object = backpack.get();  
  
// Since the backpack variable is a string, you can't pass a number to the a  
backpack.add(23);
```

Argument of type 'number' is not assignable to parameter of type 'string'.

'string'.



Overview

History

For along time, **Google Web Toolkit** (GWT) was the de-facto standard for web development.

It was written in Java and developers had to write Java code which was translated into JavaScript code for the backend and JavaScript code for the frontend.

To add one label in HTML you had to write code in Java, compile it and transform it into HTML and JavaScript to display in the web browser.

History

In 2008, after it became clear that web apps were becoming increasingly complicated, which significantly raised the compile time it took using the older framework **Google Web Toolkit** (GWT), Google tried to come up with a new plan...

2010 AngularJS was released.

It was built to provide an architecture for maintainable single-page web applications.

History

It tried to enable web designers (not developers) to write apps without knowing anything about what's going on behind the scenes.

This effectively shifted the development task from the backend to the frontend.

Angular was named after the angular brackets used in HTML...



ANGULAR VERSIONS



The change to Angular 2 was a breaking change requiring a rewrite of all applications written in AngularJS up to that point.

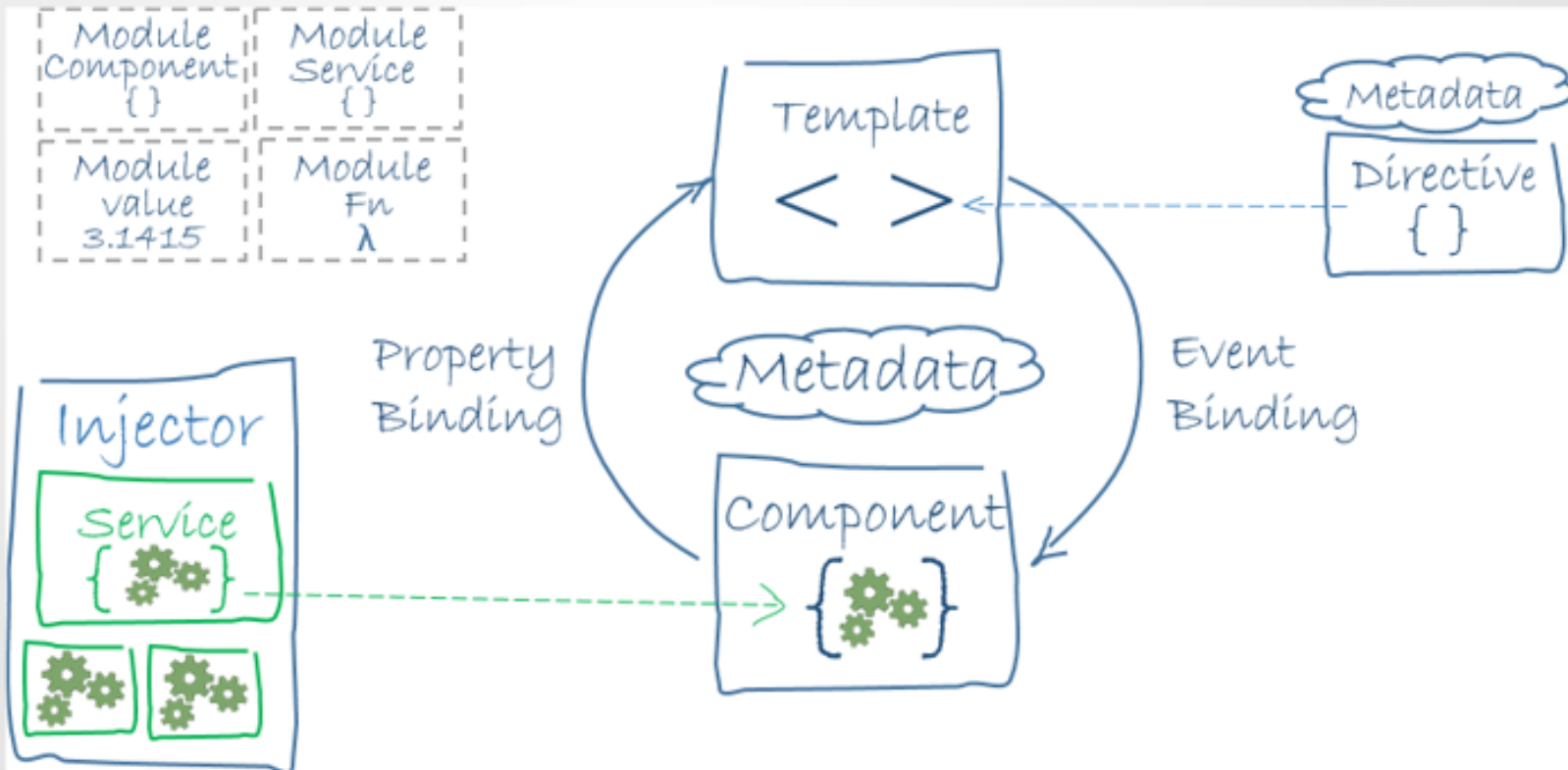
This didn't go well with the community :)



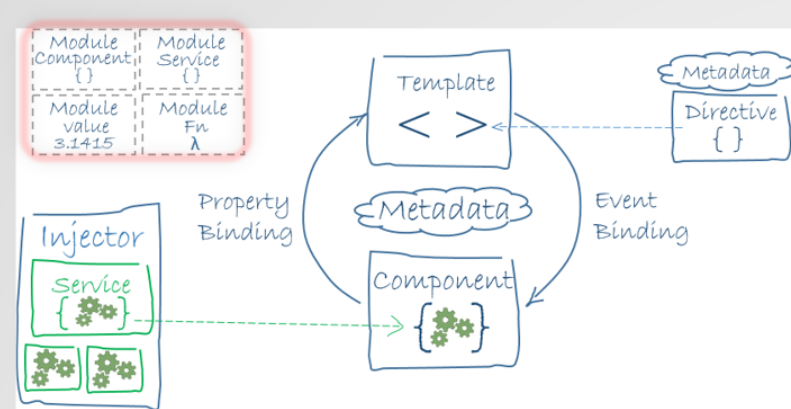
Structure

Angular: Structure

Basic structure of an Angular project.

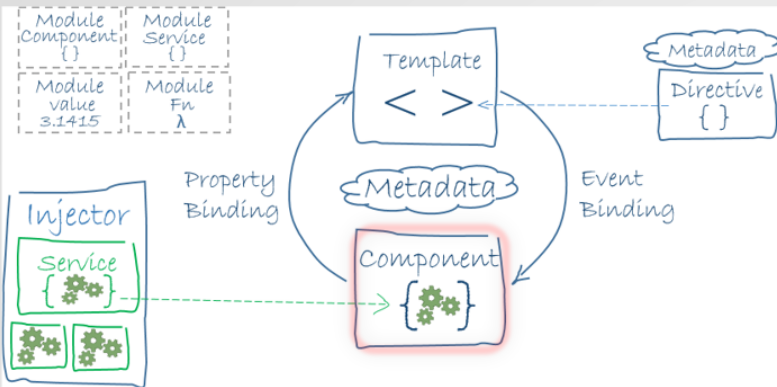


Modules



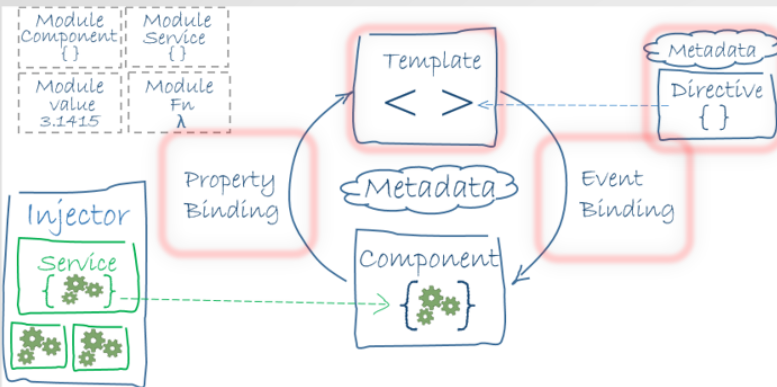
- Modules are the building blocks of Angular.
- An **NgModule** consists of **components**.
- The set of components is dedicated to an application-domain, a workflow or a closely related set of capabilities.
- Every app has the root-module called **AppModule** and contains multiple functional modules.

Components



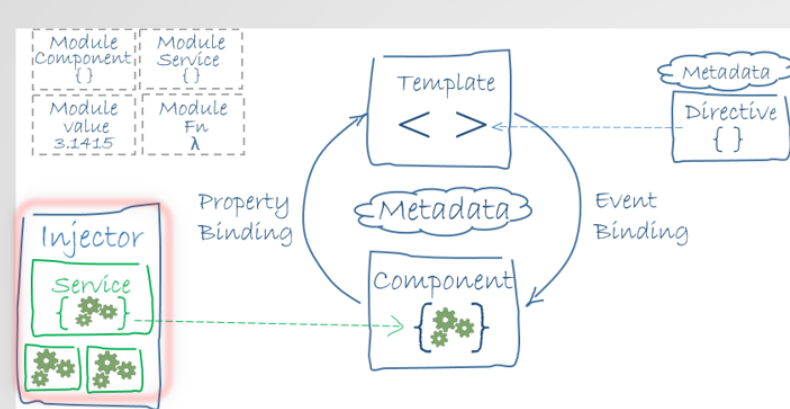
- The root component connects the component hierarchy with the Document Object Model (DOM).
- It has a HTML template that defines a view.

Templates, Directives, Data Binding



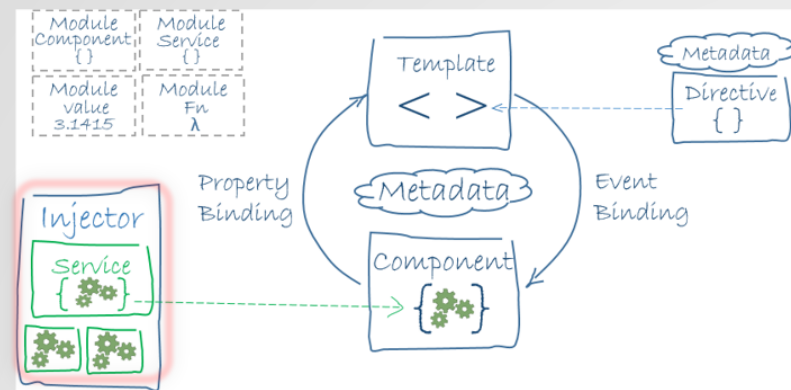
- A template combines HTML with Angular Markup that can modify HTML before they are displayed
- Directives provide program logic
- Data Binding connects your application data and the DOM
 - Property Binding
 - Event Binding

Services, Dependency Injection



- Services share common application logic between components
- DI lets you keep your component and classes lean and efficient.

Routing



- Routes from an URL entered in the browser to a specific view within your one-page application
- Can lazy-load modules



Examples

Component-Template

```
1 <h2>Products</h2>
2
3 <div *ngFor="let product of products">
4
5     <h3>
6         <a
7             [title]="product.name + ' details'"
8             [routerLink]="['/products', product.id]">
9             {{ product.name }}
10        </a>
11    </h3>
12
13    <p *ngIf="product.description">
14        Description: {{ product.description }}
15    </p>
16
17    <button type="button" (click)="share()">
18        Share
19    </button>
20
21    <app-product-alerts
22        [product]="product"
23        (notify)="onNotify()">
24    </app-product-alerts>
25
26 </div>
```

Component-Code

```
1  import { Component } from '@angular/core';
2
3  import { products } from '../products';
4
5  @Component({
6    selector: 'app-product-list',
7    templateUrl: './product-list.component.html',
8    styleUrls: ['./product-list.component.css']
9  })
10 export class ProductListComponent {
11
12    products = [...products];
13
14    share() {
15      window.alert('The product has been shared!');
16    }
17
18    onNotify() {
19      window.alert('You'll be notified when the product goes on sale');
20    }
21 }
```

Service

```
1 import { HttpClient } from '@angular/common/http';
2 import { Product } from './products';
3 import { Injectable } from '@angular/core';
4
5 @Injectable({
6   providedIn: 'root'
7 })
8 export class CartService {
9   items: Product[] = [];
10
11   constructor(
12     private http: HttpClient
13   ) {}
14
15   addToCart(product: Product) {
16     this.items.push(product);
17   }
18
19   getItems() {
20     return this.items;
21   }
22
23   clearCart() {
24     this.items = [];
25     return this.items;
26   }
27
28   getShippingPrices() {
29     return this.http.get<{type: string, price: number}[]>
30       ('/assets/shipping.json');
31   }
32 }
```



References

- <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>
- <https://www.typescriptlang.org/docs/handbook/typescript-in-5-minutes.html>
- <https://angular.io/guide/architecture>
- <https://www.clariontech.com/blog/angular-framework-from-its-first-steps-to-adulthood#:~:text=Angular%20Versions%20History,an%20architecture%20%26%20maintainable%20web%20apps.>
-