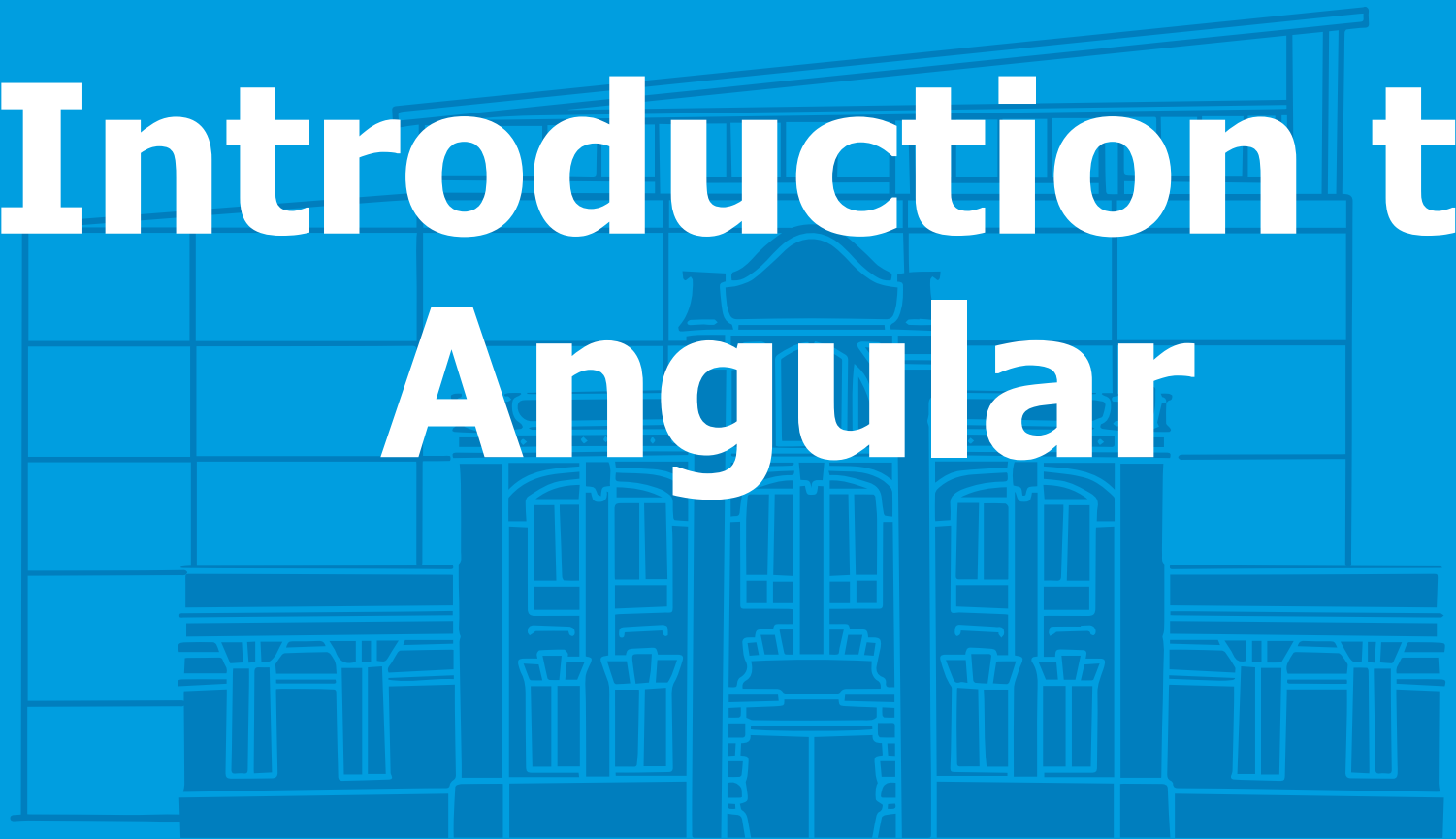
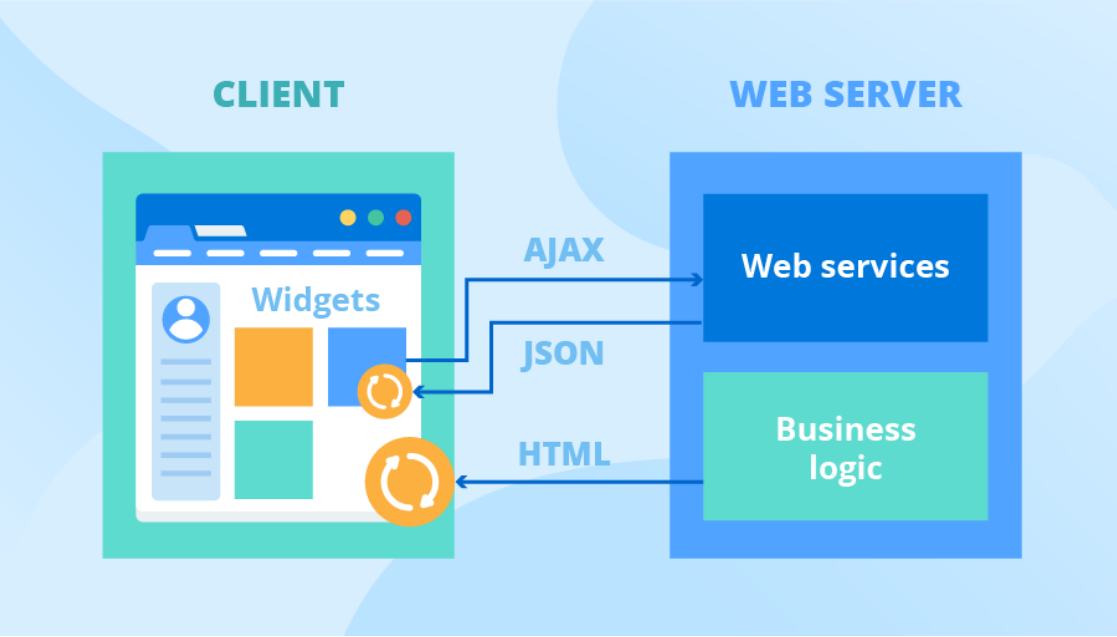
netmind
a bts company

Angular 14 - 02

Introduction to Angular

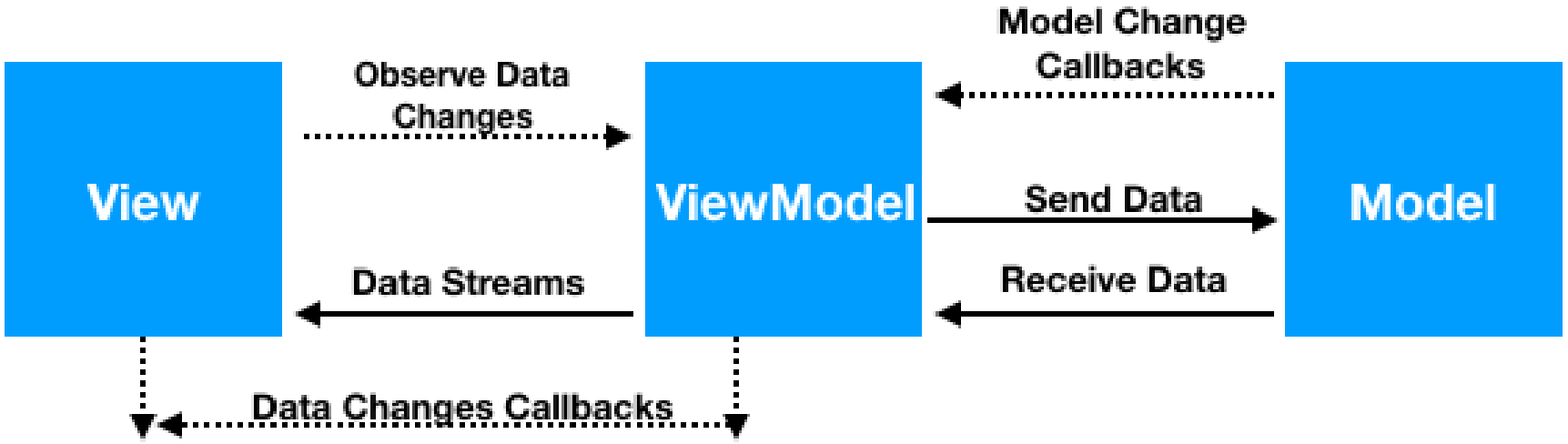


The current model of web applications



- The **browser** is the new platform
- The **cloud** allows access to content from no matter what device.
- **HTML5** and its technologies are the basis of this "framework"
- The new model of **SPA applications** (Single Page Applications) is better adapted to this architecture
- An **MVVC**-type architecture pattern is used, which facilitates the separation of responsibilities.
- Information transfer formats tend to be optimized, using **JSON** fundamentally.
  - Changes in **responsive pages** occur mainly on modified information (AJAX requests).

MVVM pattern



New Assumptions for Web Applications

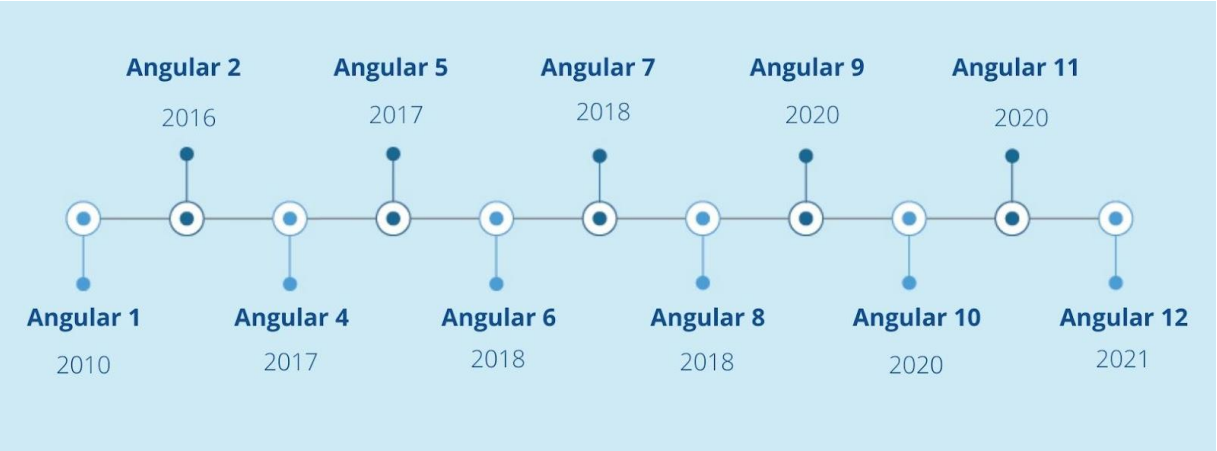
- The **user experience** (UX) has to be fluid
  - Animations and transitions help interpret dialogue with the UI
  - The new design proposals allow the creation of sites with a very similar experience on mobile and desktop devices.
  - The current speed of the JavaScript engines allows code execution taking advantage of all the hardware resources of the device.
- **Adaptation** to the device is essential for programming to be unique.
- New **APIs** available from JavaScript: Web Workers, Web Sockets, localStorage, sessionStorage, Notifications, File API, Drag&Drop, etc.

Angular

- Angular is a modern **MVVC framework** and platform that is used to build enterprise **Single-page Web Applications** (or SPAs) using HTML and TypeScript.

- Angular is **written in TypeScript**.
- It implements **core and optional functionality** as a set of TypeScript libraries that you import into your apps.
- Angular is an **opinionated framework** which means that it specifies a certain style and certain rules that developers need to follow and adhere to while developing apps with Angular.

Angular Versions



Version 14

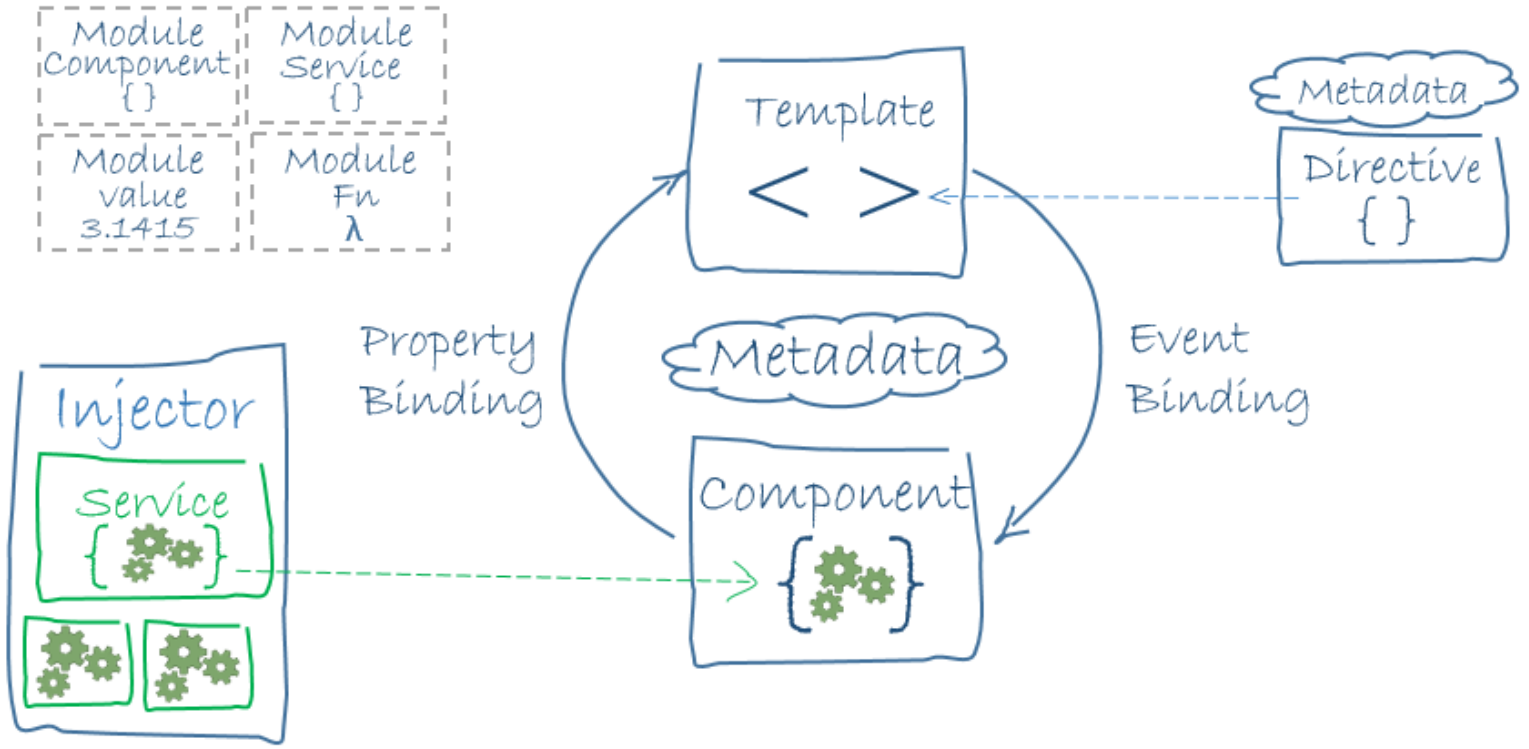
https://blog.angular.io/angular-v14-is-now-available-391a6db736af

- Simplifying Angular with Standalone Components
- Typed Angular Forms
- Streamlined best practices
- Streamlined page title accessibility
- Extended developer diagnostics
- Tree-shakeable error messages
- More built-in improvements
- Bind to protected component members
- Optional injectors in Embedded Views
- NgModel OnPush
- Built-in primitives and tooling
- New primitives in the Angular CDK
- Angular CLI enhancements
- Experimental ESM Application Builds

Angular approach

- **Separation of responsibilities** (first SOLID principle)
  - It encourages the separation of HTML manipulation and control logic.
- Separation between the server and the Web page.
- **Well structured** in several ways:
  - The design of the user interface
  - business logic
  - Unit and usage tests (Jasmine, Protractor, Karma, etc.)
  - Even in the official documentation Tests templates are provided for each demo that is shown.

Angular architecture

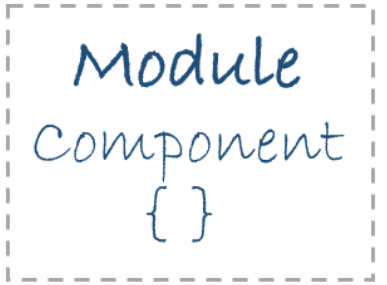


The Angular architecture is based on components that take on different **responsibilities** as indicated by **decorators**.
https://angular.io/docs

Angular architecture

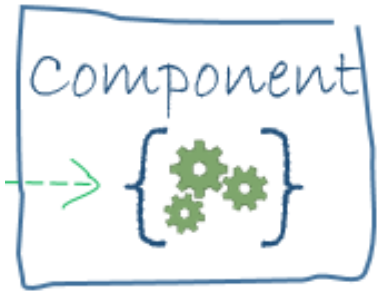
Modules

- A block dedicated to a single purpose: group angular elements.
- They contain and export (make public) components (classes, objects, …)
  - app.component.ts → export class AppComponent {}
- When we need a component must be imported.
  - import {AppComponent} from './app.component'
- Some modules are libreries of other modules → façade



Components

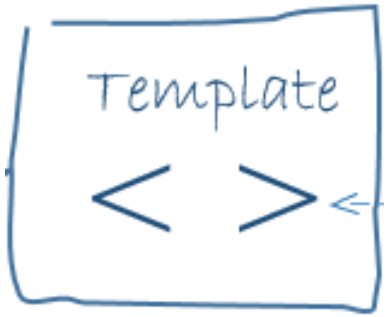
- They manage a block of the App view
- Classes → interacts with the API of the view
- They are created and destroyed as the user moves through the app.
- Can have child components



Angular architecture

Templates

- A template is a block dedicated to a single purpose: the HTML.
- Template syntax: https://angular.io/guide/template-syntax



Metadata

- Attached to classes to turn them into components
- Class decorators:
  - **@Component**: defines a class as a component
  - **@Directive**: defines a directive (adds functionality to a DOM element)
  - **@Pipe**: defines a pipe (e.g. filter)
  - **@Injectable**: Indicates that the class has dependencies that must be injected into the constructor

Angular architecture

Metadata (cont)

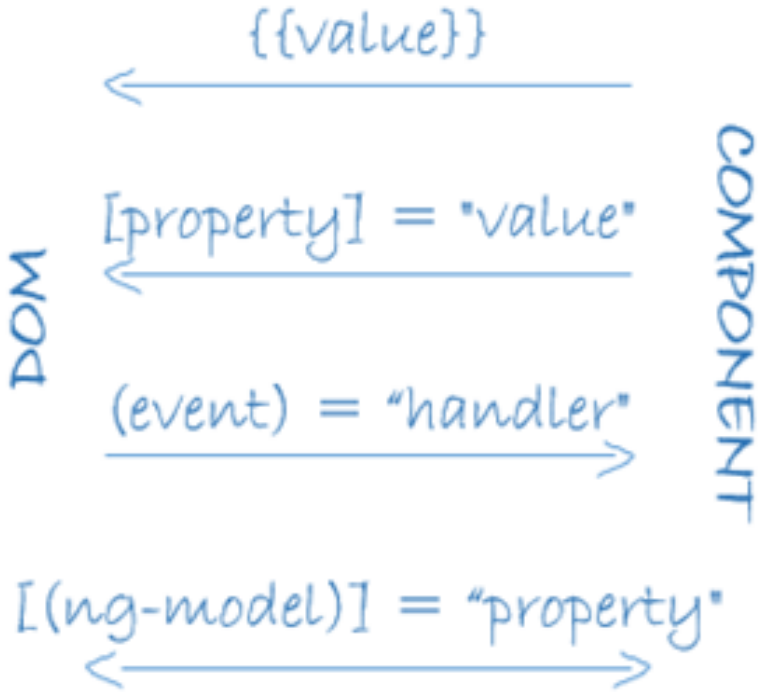
- Class Attribute Decorators
  - @Input() myProperty: an [input] property that can be updated via binding
  - @Output() myEvent = new EventEmitter(): an [output] property that fires an event
  - @HostBinding('[class.valid]') isValid: Binds to a host property (eg CSS)
  - @HostListener('click', ['$event']) onClick(e) {...} : Bind to a host event (eg onclick)
  - @ContentChild(myPredicate) myChildComponent;
  - @ContentChildren(myPredicate) myChildComponents;
  - @ViewChild(myPredicate) myChildComponent;
  - @ViewChildren(myPredicate) myChildComponents;

- Route Decorators
  - @RouteConfig: allows to define routes for the component
  - @CanActivate: Defines a function that the router should call first to determine whether to activate the component

Angular architecture

Data Binding

- It allows to put data into the html and transform user events into actions and data updates.
- Component to DOM binding:
  - {{worth}}
  - [property] = "value"
- Binding from DOM to component
  - (event) = “handler”
  - double sense binding
  - [(ng-model)] = “property”
- It also allows data binding between parent and child components.



Angular architecture

Directives

- Templates are dynamic → they are rendered according to directive instructions
- It is a class with directive metadata @Directive
  - structural
  - of attribute

Services

- Value, function, feature that our application can need.
- It is a class with a specific purpose (SR)
- Components are service consumers.

Angular architecture

- Inyección de Dependencias
  - Suministra una nueva instancia de clase con todas sus dependencias
  - Servicios necesarios → en función del tipo de los parámetros de constr.

S.O.L.I.D. principles recap

Initial letter	Acronym	Concept
S	**SRP**	**Single responsibility principle**. The notion that an object should only have a single responsibility. (used in Angular)
O	**OCP**	**Open/Close Principle**. The notion that "software entities…should be open for extension, but closed for modification".
L	**LSP**	**Liskov Substitution Principle**. The notion that "program objects should be replaceable by instances of their subtypes without altering the proper functioning of the program."
I	**ISP**	**Interface Segregation Principle**. The notion that "many specific client interfaces are better than one general purpose interface."
D	**DIP**	**Dependency inversion principle**. The notion that one should "Depend on Abstractions. Do not depend on concretions."

Anatomy of an Angular Application

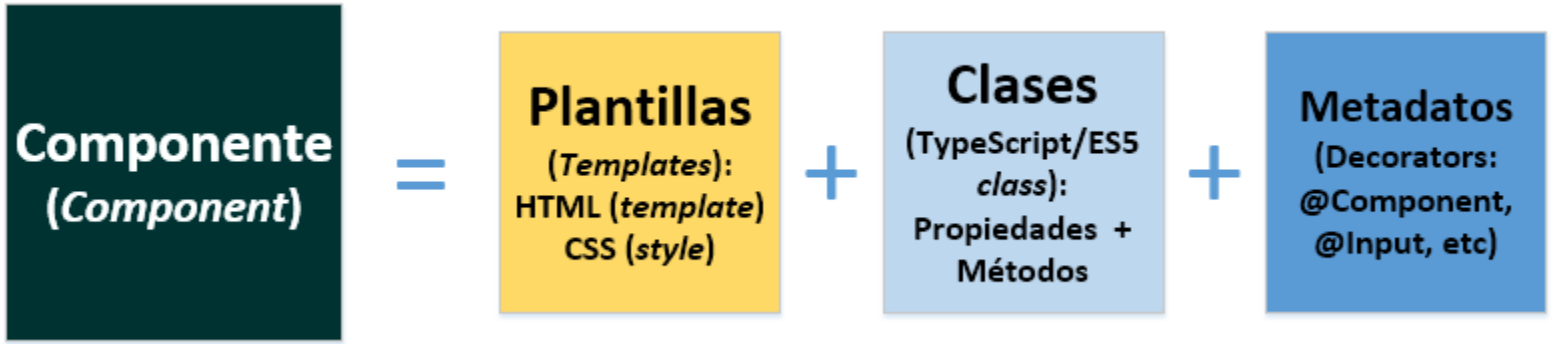
Angular applications are made up of individual components, built independently.

These components get functionality from the services available in Angular

The developer can build their own services.

Anatomy of an Angular Application

A component is made up of 3 main blocks:



- A template, which contains the HTML part (the views) and optionally, CSS styles
- A class, which provides the functional part, and has properties and methods.
- Metadata, which configure the behavior of both and their relationship with each other.

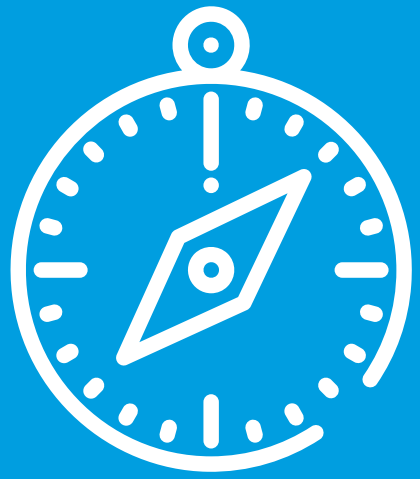
Web standards on which Angular is based

Web Components

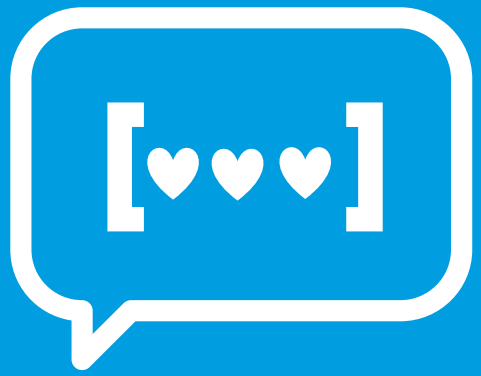
- Allow the creation of reusable widgets or components in web documents and web applications.
  - The intent behind them is to bring component-based software engineering to the Web.
  - The component model allows for the encapsulation and interoperability of individual HTML elements.
  - https://www.webcomponents.org/specs

Web Workers

- They allow the execution of heavy calculations inside the context of the different thread, which leaves the main thread of execution free, and able to handle user input and make the user interface non-blocking.
  - https://www.w3.org/TR/2021/NOTE-workers-20210128/



Next steps



We would like to know your opinion!

Please, let us know what you think about the content.
From Netmind we want to say thank you, we appreciate time and effort you have taking in answering all of that is important in order to improve our training plans so that you will always be satisfied with having chosen us
quality@netmind.es

netmind
a bts company

Thanks!

Follow us: