## Instalaciones necesarias

Para este curso necesitamos lo siguiente:

1. NodeJS: <https://nodejs.org/es/>
2. Google Chrome: <https://www.google.es/chrome/browser/desktop/>
3. TypeScript: <http://www.typescriptlang.org/>
4. Angular CLI: <https://cli.angular.io/>
5. <https://github.com/angular/angular-cli>
6. Ionic <http://ionicframework.com/getting-started/>
7. Editores de Texto:
   1. Atom https://atom.io/
   2. Visual Studio Code: <https://code.visualstudio.com/>
8. Plugins recomendados para ATOM:
   1. Angular 2 Type Script Snippets
   2. Atom Bootstrap3
   3. Atom Typescript
   4. File Icons
   5. Platformio Ide Terminal
   6. V Bootstrap4
   7. autocomplete-js-import
9. Plugins recomendados para Visual Studio Code:
10. Angular 2 TypeScript Emmet
11. Angular 5 Snippets – TypeScript, Html, Angular Material...
12. Angular Language Service
13. Angular v5 Snippets
14. Angular2-inline
15. Bootstrap 4 & Font Awesome snippets
16. HTML CSS Support
17. JavaScript (ES6) code snippets
18. JS-CSS-HTML Formatter
19. JSHint
20. Material Icon Theme
21. Prettier – Code Formatter
22. Terminal
23. TSLint
24. TypeScript Hero
25. TypeScript Importer
26. Instalar GIT https://git-scm.com/

## TypeScript y ES6

1. ¿Qué es TypeScript?
2. ¿Cómo usar TypeScript y utilizar ECMAScript 6?
3. Declaración de variables con "let" y constantes "const"
4. ¿Qué es y para qué sirve el archivo tsconfig.json?
5. Uso de los tipos de datos que ofrece TypeScript.
6. Strings de multilínea.
7. Parámetros obligatorios, por defecto y opcionales.
8. Beneficios de las funciones de flecha.
9. Uso y creación de interfaces.
10. Uso de módulos y ejemplos de los mismos.
11. Decoradores de clase.
12. Entre otros temas importantes para adentrarnos en Angular 2.

## Angular 2

1. ¿Qué son los componentes?
2. ¿Qué son las directivas estructurales?
3. Uso de plunker para nuestra primera interacción con Angular.
4. Trabajando de forma local un proyecto en Angular.
5. Una breve introducción sobre todos los archivos usados en el QuickStart de Angular.
6. Uso de Bootstrap 4 para nuestros estilos.
7. Crear archivos .HTML para que se encarguen de la estructura visual de nuestros componentes.
8. Crearemos una aplicación con 3 componentes re-utilizables.
9. \*ngFor y el \*ngIf

## Los componentes

Son clases que cumplen una tarea específica:



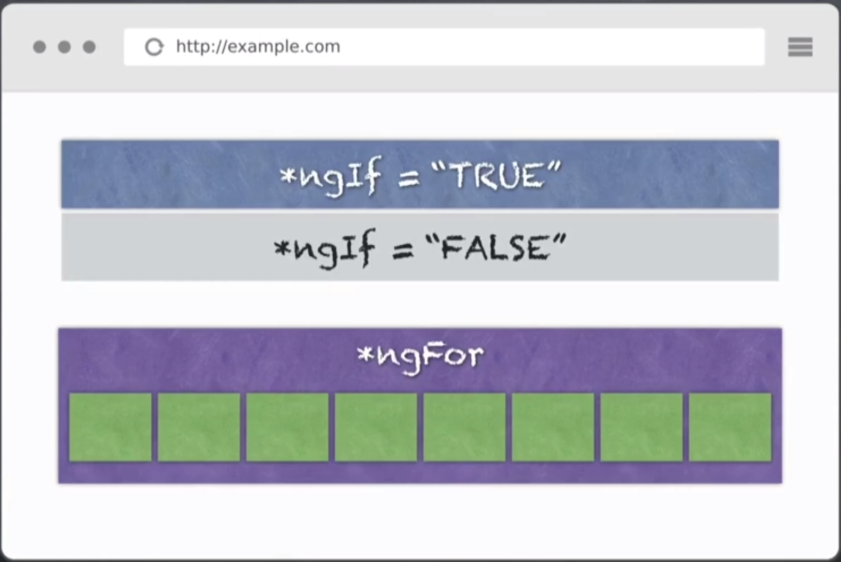
En el anterior grafico cada componente responde por sí mismo.

Los componentes son clases normales con un decorador específico.

## Directivas estructurales

Son instrucciones que le dicen al HTML que tiene que hacer:

* Ngif = “TRUE” o “FALSE” .- Muestra u oculta un contenedor (DIV)



* Ngfor = hace repeticiones de elementos HTML en nuestra página

## Qué es un decorador

Un decorador es una herramienta que tendremos a nuestra disposición en Javascript en un futuro próximo. Es una de las propuestas para formar parte del estándar ECMAscript 2016, conocido también como ES7. Sin embargo, ya están disponibles en TypeScript, por lo que podemos comenzar a usarlos ya en Angular.

Básicamente es una implementación de un patrón de diseño de software que en sí sirve para extender una función mediante otra función, pero sin tocar aquella original, que se está extendiendo. El decorador recibe una función como argumento (aquella que se quiere decorar) y devuelve esa función con alguna funcionalidad adicional.

Las funciones decoradoras comienzan por una "@" y a continuación tienen un nombre. Ese nombre es el de aquello que queramos decorar, que ya tiene que existir previamente. Podríamos decorar una función, una propiedad de una clase, una clase, etc.

Mira la primera línea del código del archivo .ts de tu componente principal.

import { Component } from '@angular/core';

Ese import nos está trayendo la clase Component. En la siguiente línea se decora a continuación, con el correspondiente "decorator". No es nuestro objetivo hablar sobre el patrón decorator en sí, ni ver las posibilidades de esta construcción que seguramente tendremos en el futuro ES7, así que vamos a centrarnos en lo que conseguimos hacer con Angular 2 mediante estos decoradores.

## Qué información se agrega por medio del decorador

Angular 2 usa los decoradores para registrar un componente, añadiendo información para que éste sea reconocido por otras partes de la aplicación. La forma de un decorador es la siguiente:

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

Como apreciarás, en el decorador estamos agregando diversas propiedades específicas del componente. Esa información en este caso concreto se conoce como "anotación" y lo que le entregamos son unos "metadatos" (metadata) que no hace más que describir al componente que se está creando. En este caso son los siguentes:

* selector: este es el nombre de la etiqueta nueva que crearemos cuando se procese el componente. Es la etiqueta que usarás cuando quieras colocar el componente en cualquier lugar del HTML.
* templateUrl: es el nombre del archivo .html con el contenido del componente, en otras palabras, el que tiene el código de la vista.
* styleUrls: es un array con todas las hojas de estilos CSS que deben procesarse como estilo local para este componente. Como ves, podríamos tener una única declaración de estilos, o varias si lo consideramos necesario.

**Nota:** Ese código de anotación o decoración del componente es generado por Angular CLI. Además, cuando creemos nuevos componentes usaremos el mismo Angular CLI para obtener el scaffolding (esqueleto) del cual partiremos. Por tanto, no hace falta que memorices la sintaxis para la decoración, porque te la darán hecha. En todo caso tendrás que modificarla si quieres cambiar el comportamiento del componente, los nombres de archivos del template (vista), hojas de estilo, etc.