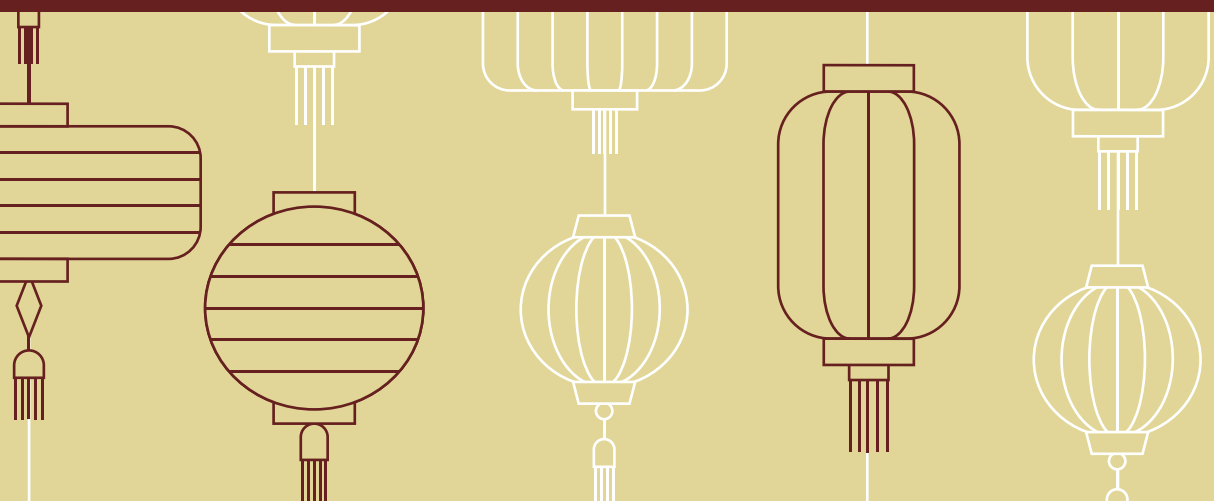


Angular :- Focused on extending HTML markup with custom attributes & dynamic content, instead of using Javascript to generate the HTML at runtime.

Semantic Versioning (SEM VER)

* Both a strategy & a naming convention that helps you know exactly what to expect from each new release.

The angular framework was built to solve specific types of problems when it comes to web applications.

Angular was originally designed to build single-page applications, or SPAs for short.

Single-page Applications (SPAs) :- websites where content updates dynamically without the need to refresh the browser.

Angular and its ecosystem.

⇒ Without Angular, developers had to embed the HTML templates inside javascript files & manipulate the DOM directly.

⇒ Now, there's reusable components, standard libraries, & routing. This is different from other modern Frameworks that focus on just building complex user interfaces or maximizing compatibility with other libraries & framework.

⇒ Angular comes with its own ecosystem of developer tools & workflows.

⇒ **Angular cli** is the official command line tool for Angular projects. It's open source

⇒ Angular's command line tool is called **ng**. It's a robust utility you can use to do all sorts of things in your project.

ng build : compile code & output files into a build directory.

ng serve : compile code, launch a dev server, & watch for file changes.

ng generate : add new files using Angular's built-in boilerplates.

ng lint & **ng test** : run Angular's lint & unit testing tools.

* **Angular Language Service**

- tells code editors how to work with Angular code.

* **Angular Dev Tools**

- A browser extension that gives you detailed debugging information in real time when running your application in a browser.

- **Angular Schematics**

- Collection of custom code to reuse across multiple projects.

Angular Decorators.

What are Decorators ?

- functions that return functions.

They are not specific to Angular framework, but they are a core piece of all Angular projects.

- Decorators in supply metadata about a piece of code telling Angular what the code should do, and how it should inject that code into other parts of the application.

- Decorators are marked by placing an @ symbol in front of decorator name. They're invoked @ runtime. & expect arguments to be passed in either between the parentheses or immediately after the closing parentheses of the decorator.

Common Decorators. https://angular.io/docs

* @NgModule() for modules
* @Component() for components
* @Injectable() for services.
* @Pipe() for pipes
* @Directive for directives
* @Input for data flowing into a component
* @Output for data flowing out of a component

that send & receive data from dom.

* common decorators errors include missing parentheses, missing properties, missing values, and adding semicolon at the end of the decorator statement.

- In older versions of Angular, decorator errors were hard to diagnose because Angular emitted errors to the console on execution & not a compile time.

- `ng build` compiles your code, & halts if there are errors.

`ng-serve` Compiles your code, reports errors, & watches for changes.

Angular Modules.

→ Angular modules represent a collection of closely related files that work together. This might include things like Components, Services, or even pipes.

→ App module File is the First module Angular loads at App launch.

Angular modules are always Typescript classes.
We use the NgModule decorator, to declare a class as a module.

- The NgModule decorator takes a configuration object as its only argument. The declaration's property is an array of all the components, directives, and pipes that belong to this module.

- the `imports` property is an array of all the external modules we want to make available inside this module's template.

- We import the `AppRoutingModule`, which gives us access to Angular's page navigation features.

- The `BrowserModule`, which is all the low-level infrastructure Angular needs in all projects.

- The `CommonModule`, which has all of Angular's built-in pipes & directives

- The `HTTP Client Module`, which gives us access to the HTTP client service for making fetch requests

- The next config option is called exports, & it's an array of all code in this module we want to make available for use in other modules.

- The provider's property is an array of all code we want to make available in this module through Angular's dependency injection.

- As a general rule, you put modules in the imports array & things like services in the providers array.

- Special property called bootstrap, this property tells Angular which components to load when you launch your App.

- When using a module architecture, you must have at least one module in your project, AKA root module.

The Angular CLI generates the following basic `AppModule` when creating a new application.

src/app/app.module.ts (default AppModule)

```
// imports
import { BrowserModule } from '@angular/platform-
browser';
import { NgModule } from '@angular/core';

import { AppComponent } from './app.component';

// @NgModule decorator with its metadata
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

At the top are the import statements. The next section is where you configure the `@NgModule` by stating what components and directives belong to it (`declarations`) as well as which other modules it uses (`imports`). For more information on the structure of an `@NgModule`, be sure to read Bootstrapping.

Angular Components.

- Angular components :- main building blocks in all Angular Apps.

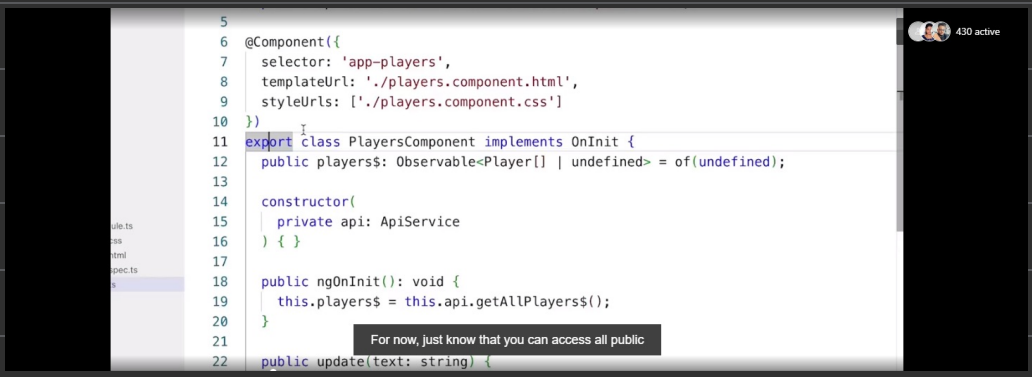
- Components have four parts.

- **HTML Templates** :- defines the UI for component.

- **Css styles** :- Styles you want to Apply to the template.

- **typescript** :- class that controls the component's behavior and a

- **component selector** :- that tells Angular where to put the component in the DOM.



- We export the Component class itself on line 11. This is standard TypeScript class, & we use it to inject dependencies, tap into Lifecycles hooks & make data available in the template.

- you can access all **public** or **protected** class members from inside the HTML template.

- **players$** has initial value of an observable that returns undefined.

- **Observable** = think of them as a Promise

- data that returns more than once, putting the dollar sign at the end of the property name is the standard convention when working with observables in Angular.

- In the constructor of our Component class, this is where you inject dependency & initialize code.

- Angular calls the constructor only once for each component service or other dependency.

- ngOnit() is a lifecycle hook built into Angular. Just think of them as a way to tell Angular when to execute certain piece of code.

Template reference variables are a quick way to use data from one part of the template in other parts of the same template without having to pass the data into the component's controller class.

Template variables always start with a hash and are followed by a custom string.

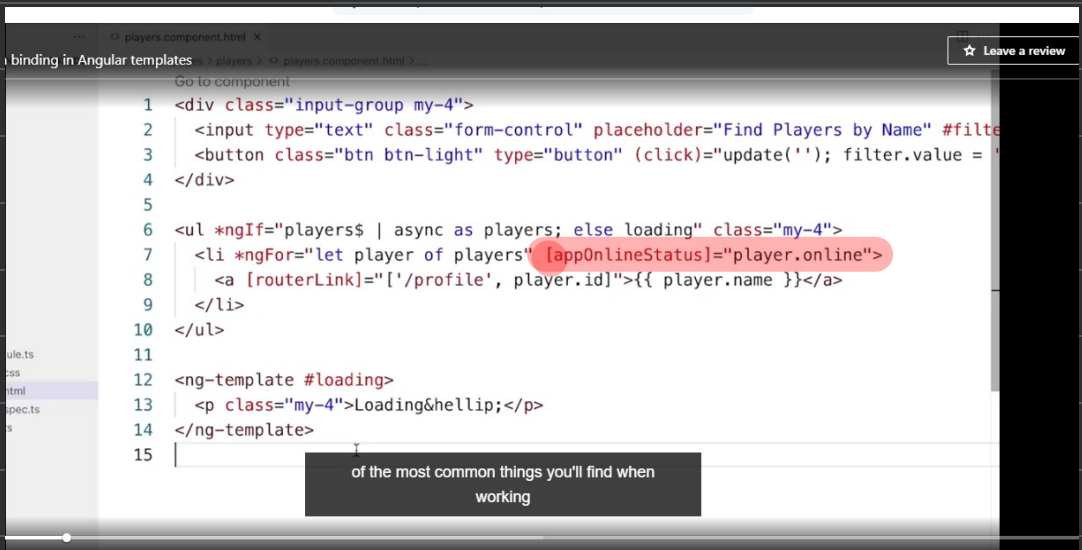
there are three common types of data binding in Angular, event binding, property binding, and interpolation.

- **Event Binding** Lets us quickly assign actions to DOM events like keyup & click.

we put the name of the event inside the parentheses, keyup in our case, and assign one or more actions to run when that event happens on this element.

```
#filter (keyup) = "update (filter.value)"
```

Reference Variables & data binding in Angular Templates



- Always use the Component.html ending for the file names for your component templates.

- Property binding lets us dynamically assign values to the attributes on a standard HTML element or to properties on a Angular component.

- We declare property bindings with square brackets. We put the name of the attribute or property inside the square brackets & assign it a value.

- **Interpolation** lets us display dynamic content as text in the UI. This is done by two curly braces.

Structural directives change the layout of your UI by adding & removing DOM elements.

- These directives always start with an asterisk followed by the directive name.

Standalone Components

* Available in Angular v14+
* lets you use files without modules.

```
@Component {                    // false by default
    standalone: true,
}
```

Angular pipes.

- **pure Functions** :- A Function that returns the same output for the same inputs.

Standard pipes

- Angular includes 13 pipes with all new projects.
- Helps format things like dates, numbers, and text.