# کامپوننت چیست؟

اصلی‌ترین واحد در انگولار است. هر اپلیکیشن انگولار از مجموعه‌ای از کامپوننت‌ها تشکیل می‌شود.

# اجزای کامپوننت

آنگولار از معماری MVC استفاده می‌کند و به همین خاطر هر کامپوننت شامل یک کلاس TypeScript برای Logic و فایل‌هایی برای HTML و CSS برای UI است.

# ارتباط UI و Logic

در آنگولار این ارتباط از طریق Binding انجام می‌شود.

# انواع Binding در آنگولار

* String Interpolation: برای نمایش خروجی از آن استفاده می‌شود، یعنی Binding به صورت یکطرفه است. {{}}
* Property Binding: از نوع Binding یکطرفه است و برای مقداردهی Property های HTML Tag مقصد استفاده می‌شود. []
* Event Binding: Bind کردن مقادیر یک HTML Tag به کلاس TypeScript مربوط به Logic. ()
* Two-way Binding: ترکیبی از Property Binding (ارتباط یکطرفه از Logic به UI) و Event Binding (ارتباط یکطرفه از UI به Logic). [()]

# انتقال داده بین Component ها

* ارسال از کامپوننت پدر به فرزند: در کامپوننت فرزند یک property تعریف می‌کنیم که با Decoratorی با نام @Input() مشخص می‌شود و از طریق selector آن کامپوننت در پدر قابل دسترس است.
* ارسال از کامپوننت فرزند به پدر: این کار باید از طریق event انجام شود، پس در کامپوننت فرزند یک event تعریف می‌کنیم که با Decoratorی با نام Output مشخص می‌شود و از طریق selector آن کامپوننت در پدر قابل دسترس است.

# Style دهی به کامپوننت‌ها

* ViewEncapsulation.Emulated (پیش فرض): Style ها از HTML اصلی به Component منتقل می شوند و همچنین Style های تعریف شده برای آن کامپوننت فقط برای آن کامپوننت است.
* ViewEncapsulation.Native: Style های HTML اصلی به کامپوننت منتقل نمی‌شوند و همچنین Style های تعریف شده برای آن کامپوننت فقط برای آن کامپوننت است.
* ViewEncapsulation.None: Styleهای کامپوننت به کل اپلیکیشن انتقال می‌یابد، دقت شود اپلیکیشن‌هایی که دارای اجزای None و Native در خود هستند. همه اجزای دارای کپسول‌سازی None در همه اجزای دارای کپسوله سازی Natvie، Style خود را منتشر می‌کنند.

# دسترسی به Dom Element

* در Logic کامپوننت: با استفاده از دکوریتور ViewChild که نام local reference را می‌پذیرد.
* در HTML: با استفاده از local reference .

# ارسال HTML به کامپوننت فرزند

هر چیزی که داخل selector مربوط به کامپوننت فرزند قرار بگیرد را می‌توان با استفاده ng-content در کامپوننت فرزند استفاده کرد.

# Lifecycle

هر کامپوننت از لحظه ایجاد تا نابودی مراحلی را طی می‌کند که انگولار به ما اجازه می‌دهد به این مراحل دست پیدا کنیم و کارهای مورد نظرمان را انجام دهیم، به هر مرحله چرخه‌ی حیات Hook گوید.

* ngOnChanges: اولین هوکی که پس از تغییر اطلاعات ورودی Bind شده رخ می‌دهد، این متد می‌باشد.
* ngOnInit: این هوک پس از ngOnChanges رخ داده و اگر مقادیر پیشفرضی به یک کامپوننت اختصاص داده باشیم، جایگزین خواهد شد.
* ngDoCheck: پس از بررسی‌های فوق این هوک اجرا شده و در نهایت هر تغییری که در طی برنامه رخ دهد را مورد بررسی قرار می‌دهد.
* ngAfterContentInit: این هوک پس از مقداردهی اولیه تگ ng-content (که در کامپوننت فرزند استفاده می‌شود و placeholderی است به HTML ارسالی از کامپوننت والد) فعال می‌شود.
* ngAfterContentChecked: این هوک بررسی می‌کند که آیا مقادیر موجود در تگ ng-content تغییر کرده است یا خیر؟ در صورت تغییر فعال می‌شود.
* ngAfterViewInit: این هوک پس از مقداردهی اولیه‌ی childViewها درون کامپوننت فعال می‌شود.
* ngAfterViewChecked: این هوک بررسی می‌کند که آیا مقادیر موجود در لوکال رفرنس‌ها childView تغییر کرده است یا خیر؟ در صورت تغییر فعال می‌شود.
* ngOnDestroy: آخرین هوکی که فعال می‌شود، ngOnDestroy نام دارد که به هنگام تخریب یا عدم استفاده از یک کامپوننت به چرخه حیات پایان می‌بخشد.

برای دسترسی به هر هوک فوق یک Interface با همان نام در angular/core تعریف شده است که باید آن را پیاده سازی کرد.