WHAT IS ANGULAR?

BOOTSTRAP

Lets take a look at angulars bootstrap library. We can install it using **npm**

```
npm install -g @angular/cli
ng new demo_angular
```

This creates a new folder on your filesystem called demo_angular

WHAT JUST HAPPENED?

Angular has now created a whole filestructure for us.
Lets look at the most essential files

- **demo**
  - **src**
    - **app**
      - app.component.css
      - app.component.html
      - app.component.ts
      - app.module.ts
    - index.html

NGMODULE

The basic building blocks of all Angular applications are NgModules. Modules collect related code into functional sets.

```
# app.module.ts
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent]
})
```

Every model defines what Components, Modules and dependencies that should be loaded for each module.

ROOT AND FEATURE MODULES

Each application has at least one **Root Module** and typically many more **feature modules**.

Each module consists of **Components** and **Services**

The root module defines bootstraping and

COMPONENT

Each component consists of a class, a css-selector and a html template.

```
# app.component.ts
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'demo';
}
```

TEMPLATE

A template combines HTML with angular markup.

```
# app.component.html
<div>The title is {{ title }}</div>

# app.component.ts
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
```

The HTML will be rendered in the HTML-element found by **selector**

INLINE HTML

The template can also be rendered using inline syntax.

```
@Component({
  selector: 'app-root',
  template: `
    <h1>{{title}}</h1>
    `
})
```

The backtick (`) lets us compose a string over multipleline.

TEMPLATE DIRECTIVES

Template directives provide program logic

```
<h1>{{title}}</h1>
```

DATA BINDINGS

Binding markup lets us connect data and the DOM.
There are two types of data bindings:

- Event binding lets your app respond to user input.
- Property binding lets you interpolate values.

INTERPOLATION

Interpolation is the practice of embedding expressions into our HTML. We use the double curly brackets `{ { } }` to indicate that we want to interpolate an expression. Thus the following expressions:

```
{{ title }}
```

Would interpolate the value from the variable `title` defined in our component.

MORE COMPLEX EXPRESSIONS

We can use the same syntax to define more complex expressions

```
{{ (age > 18) ? 'Above 18' : 'Less than 18'}}
```

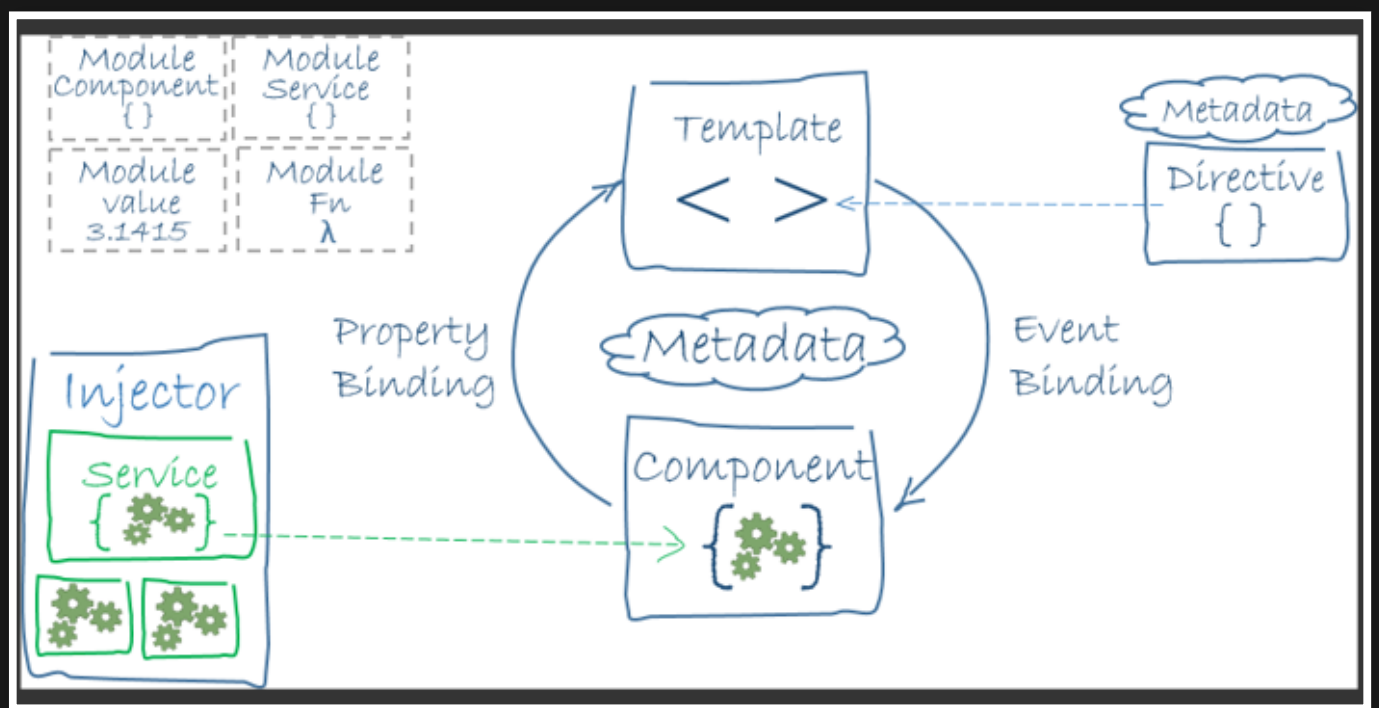
SERVICES

Services can be used to share logic between multiple components. This helps us to keep components clean and testable, by delegating **business logic** to services.

DEPENDENCY INJECTION(DI)

When we use services in components, or other services, we inject them into our code at runtime. This means that they are decoupled from each other, and can be replaced easily. A common use case is to inject a mocked service when testing our code.

OVERVIEW


Module
Component
{ }
Module
Service
{ }
Module
value
3.1415
Module
Fn
λ
Template
< >
Metadata
Directive
{ }
Property
Binding
Metadata
Event
Binding
Injector
Service
{ }
Component
{ }