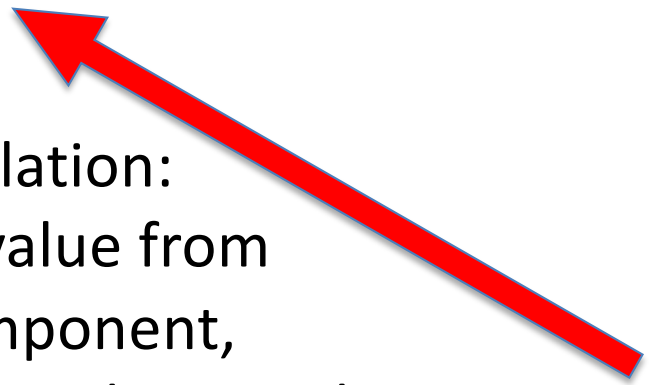
Q

How do we define a template?

Template

it is a standard HTML file, with binding to the Component and Angular directives

```
<tr *ngFor="let customer of customers;">
   <td>{{customer.name}}</td>
</tr>
```



Interpolation:
take a value from
the component,
put it into the template

Component

…
Customer x;
x.name="Pippo"

Angular directives

Structural directives can change the DOM layout by adding and removing DOM elements.

they are: **ngIf, ngFor, ngSwitch**

All structural Directives are preceded by Asterix symbol.

Attribute directives or **Style** directives can change the appearance or behavior of an element.

they are: **ngModel, ngClass, ngStyle**

Structural directives in templates: ngIf

in component:

```
export class AppComponent {
  showMe: boolean;
}
```

in template:

```
<p *ngIf="showMe">
  ShowMe is checked
</p>
```

also:

- ngIf else
- ngIf then else

```
<div *ngIf="condition; then thenBlock else elseBlock">
   This content is not shown
</div>

<ng-template #thenBlock>
   content to render when the condition is true.
</ng-template>

<ng-template #elseBlock>
   content to render when condition is false.
</ng-template>
```

see details and examples in
https://www.tektutorialshub.com/angular/angular-ngif-directive/

Structural directives in templates: ngSwitch

in component:

```
export class AppComponent {
  num: number= 0;
}
```

in template:

```
<div [ngSwitch]="num">
     <div *ngSwitchCase="'1'">One</div>
     <div *ngSwitchCase="'2'">Two</div>
     <div *ngSwitchCase="'3'">Three</div>
     <div *ngSwitchCase="'4'">Four</div>
     <div *ngSwitchCase="'5'">Five</div>
     <div *ngSwitchDefault>This is Default</div>
   </div>
```

see details and examples in
https://www.tektutorialshub.com/angular/angular-ngswitch-directive/

Structural directives in templates: ngFor

in component: define customer, and customers as an array of customer items

in template:

```
<tr *ngFor="let customer of customers;">
   <td>{{customer.customerNo}}</td>
   <td>{{customer.name}}</td>
   <td>{{customer.address}}</td>
   <td>{{customer.city}}</td>
   <td>{{customer.state}}</td>
</tr>
```

see details and examples in
https://www.tektutorialshub.com/angular/angular-ngfor-directive/

Q

How does the class pass values to the template?

interpolation

The simplest form of binding is interpolation (Template Expression).

Interpolation carries values from the class to the template.

Changes in property values are reflected in the interpolation.

Values must be strings, and are substituted to the interpolation.

```
//Template
{{title}}
{{getTitle()}}

//Component
title = 'Angular Interpolation Example';
getTitle(): string {
    return this.title;
}
```

See https://www.tektutorialshub.com/angular/interpolation-in-angular/

interpolation

Interpolation can also evaluate simple expressions, such as e.g. string concatenation or arithmetic operations. They can bind to any property that accepts a string.

```
<span> {{3*8}} </span>
<p>Show me <span class = "{{giveMeRed}}">red</span></p>
<p style.color={{giveMeRed}}>This is red</p>
```

Interpolation cannot cause effects in the state of the Component: no assignments, no instantiation of classes, no side effects.

They are the simplest form of one-way binding:

Component  Template

interpolation

Interpolation can be applied also to bind to a value of a property of an HTML element. We can bind *to any property that accepts a string*.

```
<p>Show me <span class = "{{giveMeRed}}">red</span></p>
<p style.color={{giveMeRed}}>This is red</p>


<a href="/product/{{productID}}">{{productName}}</a>
```

https://www.telerik.com/blogs/understanding-angular-property-binding-and-interpolation

property binding

To bind a property of an HTML element there is also a different syntax, e.g.

```
<img src="{{ myProperty }}">
<img [src]="myProperty">
```

```
<h1 [innerText]="title"></h1>
<button [disabled]="isDisabled">I am disabled</button>

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title="Angular Property Binding Example"
  isDisabled= true;
}
```

property binding

interpolation "injects" the value into the html, so when you say value="{{ hello }}" Angular is *inserting* your variable between the brackets. It can only be applied to strings.

property binding allows Angular to directly access the elements property in the html. this is a deeper access. When you say [value]="hello" Angular is grabbing the value property of the element, and *setting* your variable as that property's value.
It can be applied to any data type (e.g. objects).

special cases of property binding: class

Normal html:

```
<div class="red">red</div>
```

Property binding:

NOTE the double + single quote!

```
<div [className]="'red'">Test</div>
<div [className]="'red size20'">Test</div>
<div [ngClass]="'red size20'">
<div ngClass='red size20'>
```

NOTE the single quote only!

The ngClass attribute directive is more powerful, you can use arrays, objects and conditional assigments.

see https://www.tektutorialshub.com/angular/angular-ngclass-directive/

and https://www.tektutorialshub.com/angular/property-binding-in-angular/

special cases of property binding: style

Normal html:

```
<body style="background-color:grey;">
```

Property binding:

NOTE the double + single quote!

```
<p [style.background-color]="'grey'">
<p [style.color]="getColor()"
   [style.font-size.px]="'20'"
   [style.background-color]="status=='error' ? 'red': 'blue'">
   paragraph with multiple styles
</p>
```

.

see https://www.tektutorialshub.com/angular/angular-style-binding/

special cases of property binding: ngStyle

The Angular ngStyle directive allows us to set the many inline style of a HTML element using an expression. The expression can be evaluated at run time allowing us to dynamically change the style of our HTML element

```
<div [ngStyle]="{'color': color}">
```

for details, see
https://www.tektutorialshub.com/angular/angular-ngstyle-directive/

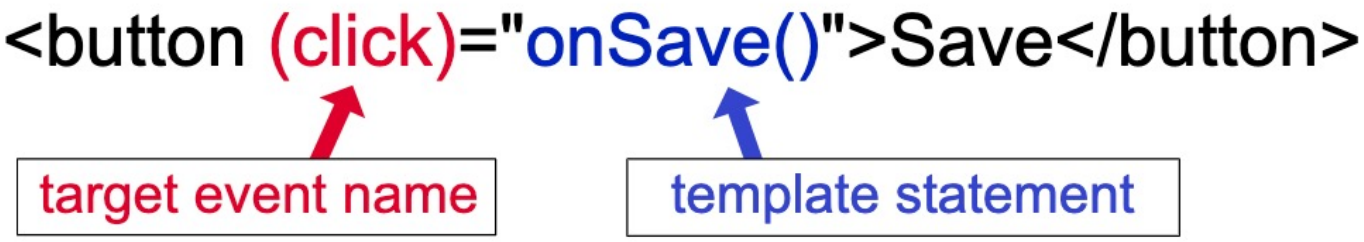
Q

How does the view pass values to the class?

Event binding

We can bind events such as keystroke, clicks, hover, touch, etc to a method in component.
It is one way from view to component.

For Example, when the user changes a input in a text box, we can update the model in the component, run some validations, etc. When the user submits the button, we can then save the model to the backend server.



Event binding

DOM Events carries the event payload. I.e the information about the event. We can access the event payload by using $event as an argument to the handler function.

template

```
<input (input)="handleInput($event)">
<p>You have entered {{value}}</p>
```

component

```
value=""
handleInput(event) {
  this.value=event.target.value
;
}
```

Event binding - example

app.component.ts

```
@Component({
    selector: 'app-root',
    templateUrl:
'./app.component.html',
    styleUrls:
['./app.component.css']
})
export class AppComponent {
    title = 'app works!';
    value=""
        handleInput(event) {
            this.value=event.t
arget.value;
        }
}
```

app.component.html

```
<h1>
  {{title}}
</h1>
<input
(input)="handleInput($event)"
>
<p>You have entered
{{value}}</p>
```

Event binding

The events are those of HTML DOM component:
http://www.w3schools.com/jsref/dom_obj_event.asp
by just removing the on prefix. onclick ---> (click)

- (focus)="myMethod()"
- (submit)="myMethod()"
- (cut)="myMethod()"
- (paste)="myMethod()"
- (keypress)="myMethod()"
- (mouseenter)="myMethod()"
- (mouseup)="myMethod()"
- (dblclick)="myMethod()"
- (dragover)="myMethod()"
- (blur)="myMethod()"
- (scroll)="myMethod()"
- (copy)="myMethod()"
- (keydown)="myMethod()"
- (keyup)="myMethod()"
- (mousedown)="myMethod()"
- (click)="myMethod()"
- (drag)="myMethod()"
- (drop)="myMethod()"

There are actually many more. see:

https://developer.mozilla.org/en-US/docs/Web/Events

Event binding

Instead of parentheses, you can also use the on- syntax:

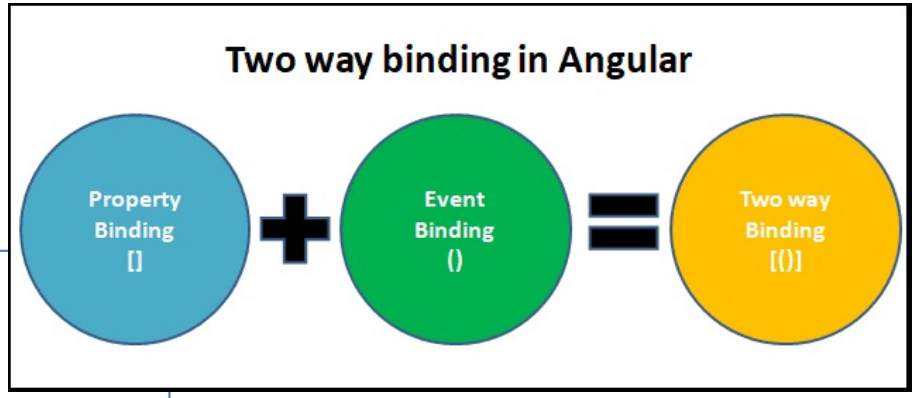
```
<button on-click="clickMe()">Click Me</button>
```

for details, see
https://www.tektutorialshub.com/angular/event-binding-in-angular/

Q

Can we have data passing in the two directions (view to class/class to view)?

two-way binding



Component:

```
value="";
clearValue() {
  this.value="";
}
```

Template:

```
<input type="text"
[(ngModel)]="value">
<p> You entered
{{value}}</p>
<button
(click)="clearValue()">
Clear</button>
```

1) when you type in the input, value is propagated from Template to Component (thanks to ngModel)

2) {{value}} is one-way binding from Component to Template

3) a click on the button triggers the call of clearValue(), which changes value, which is reflected both in <input> (thanks to the two-way binding of ngModel) and on <p> (thanks to one way binding given from interpolation).

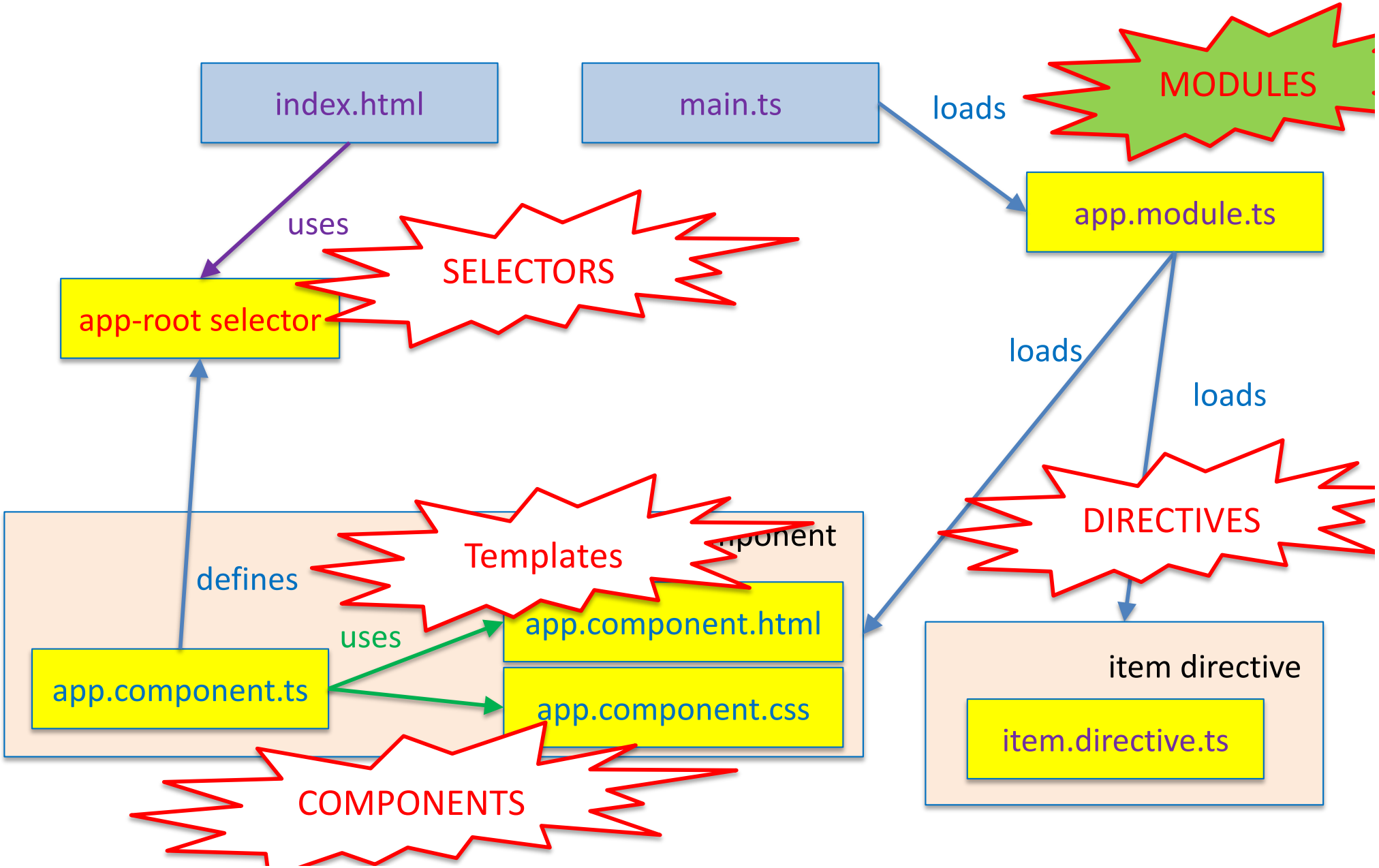
See also https://www.tektutorialshub.com/angular/ngmodel-two-way-data-binding-in-angular/

Q

So far we have seen only a pair template+component.

How do I compose a full app?

Putting the pieces together



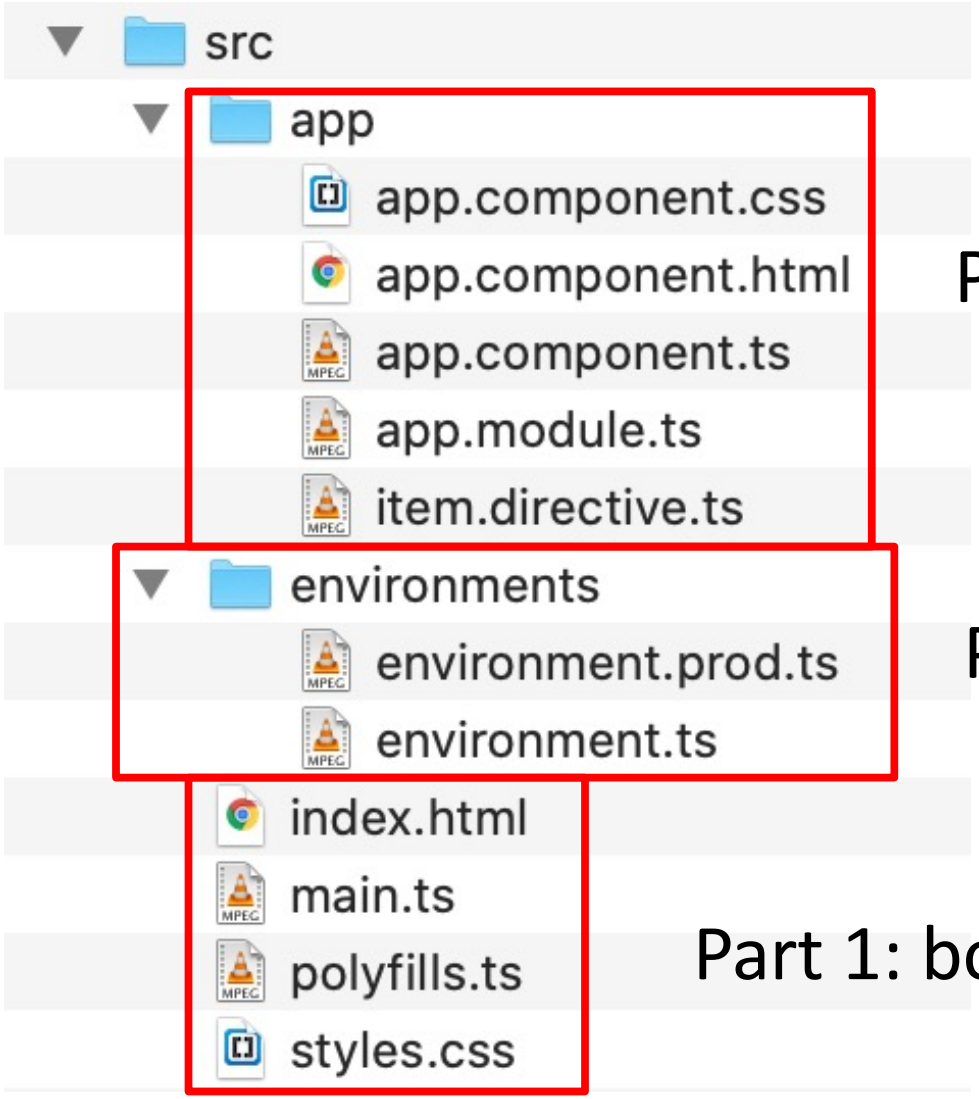
Modules: NgModule

The Angular Modules (or NgModules) are Angular ways of group together functional units: related components (but also directives, pipes and services, etc).

Every Component must belong to an Angular Module and cannot be part of more than one module. A component is registered in a Module by declaring it in the Module's metadata.

Every Angular app has a *root module*, conventionally named AppModule, which provides the bootstrap mechanism that launches the application.

Let's expand src



Part 3: the app

Part 2: deployment properties

Part 1: boot Section

Modules: NgModule

An app typically contains many functional modules.

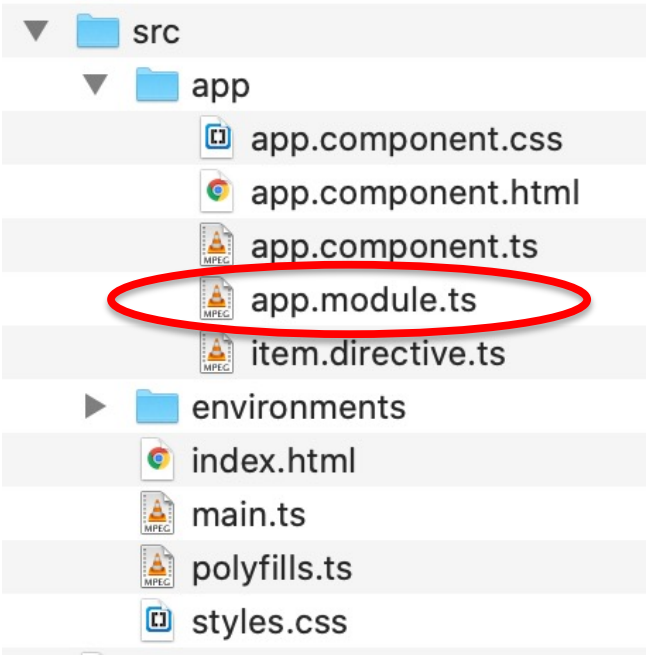
NgModules can import and export functionality from/to other NgModules.

Organizing code into distinct functional modules helps in managing development of complex applications, and in designing for reusability.

lazy-loading loads modules on demand—to minimize the amount of code that needs to be loaded at startup.

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import { AppComponent } from './app.component';
import { ItemDirective } from './item.directive';
// @NgModule decorator with its metadata
@NgModule({
  declarations: [
    AppComponent,
    ItemDirective
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```



Adding a component to a module: ng g – ng generate

To create a new module:

`ng g module newModule`

Assuming you already have a module:

`cd newModule` to change directory into the newModule folder

`ng g component newComponent` to create a component as a child of the module.

or

`ng g component myModule/new-component`

(specifying the path to the module you want to insert the component into)

For a full example, see

https://www.tektutorialshub.com/angular/angular-adding-child-component/