@Component

template	define template (using `)
templateUrl	define template
host	the element in which a component is attached to

helpers

(click)	js default, propagates the `click` event to **all** the parent components. So if want to not allow this, `return false`

syntax

```
<inventory-app></inventory-app>
// or
<div invent ory -ap p>< /di v>
input: ['name'] // or
@Input() name: string // or
input: ['inside: outside'] //
avoid
@Input ('o uts ide') inside:
string // avoid
// set classes condit ionally
[class.se lec ted ]="i sSe lec -
ted (my Pro duc t)"
// (p122/644)
src="{{product.imageUrl}} //
wrong
[src]= " pro duc t.i mag eUr l"
// right
```

best practices

use `template` when the view is not much & vice versa. the drawback of using `template` is not having syntax hightlight

using the `host` option is nice because it means we can **encapsulate** the `app-ar ti cle` marup **within** our component. By using the `host` option, we're able to configure our **host element** from **within** the component.

isolate the data structures from the component code

law of demeter a given object should assume as little as possible about the structure or properties of other objects

fat models, skinny controllers

when buiding a new angular app, mockup the design & then break it down into components

normally, author wouldn't pass more than 5 arguments to a function.

cli

watchman	OSX: `brew`, Linux: embercli, Window: native Nodejs watcher
ng new `app`	create a new ng2 app
ng serve	run app through `http` built in. `Window: --host 0.0.0.0`
ng generate component `co mponent`	create a new component

resources

terminology

```
#newtitle
```
is called `resolve`. makes the variable `n ewtitle` available to the **expressions** within the **view**. `newtitle` is an **object** (typeof `HTMLIn put Element`) that represents this `input` DOM element

```
newtitle
```
template variable

```
Article[] or Array< Art icl e>
```
generics

```
{{ }}
```
template binding

```
private curren tPr oduct: Produc t
```
local component state

knowledge

- one of the big ideas behind Angular is the idea of `components`.

- the fundamental idea behind `component s`: we'll teach the browser **new tags** that have custom functionality.

- `components` are the **new** version of `dire ctives ng-1`

angular1's **dependency injection** used the **annotation** concept behind the scenes

when boot an Angular app, you're not booting a component directly, but instead you create an `NgModule` which points to the component you want to load.

By **addyosami**

cheatography.com/addyosami/

Not published yet.
Last updated 25th January, 2017.
Page 1 of 2.

knowledge (cont)

you have to declare **components** in a `NgMo dule` before you can use them in your templates

Angular 1, **directives** match globally. Angular 2, need to **explicitly** specify **which components** you want to use

JavaScript, **by default**, **propagates** the `clic k` event to **all the parent components**

`href=""` (empty link) === **reload page**

an angular2 is nothing more than a **tree** of **components**. **top level Component** is the application itself.that's what the browser will render when **booting** (a.k.a **bootstrapping**) the app.

`@Component` annotation is where you configure your component. Primary, `@Comp onent` will configure how the **outside world** will interact with your component.

`[]: input, (): output.`
Data flows in to your component via **input bindings** and **events flow out** of your component through **output binding**.
Think of the set of **input + output bindings** as defining the **public API** of your component.

In Angular, you send data out of components via **outputs**.

`(onPro duc tSe lected):` the name of the **output** we want to **listen** on
`produc tWa sSe lected:` the function we want to call when **something new** is on this **output**
`$event:` special variable that represents the thing emitted on the **output**

knowledge (cont)

when we specify that a **component** takes an **input**, it is expected that the definition class will have **an instance variable** that will receive the value

`<button (click )="i ncr eas e() " >In c</ but ton>`

In this case, the **event** is **internal** to the **component**. we can also expose **public event** (component `output`) that allow the component to talk to the **outside** world

An `EventE mitter` is simply **an object** that helps you implement the `Observer Pattern`. That is, it's an object that can maintain a list of **subscribers** and **publish** events to them.

When we assign an `EventE mitter` to an **output** Angular **automatically subscribes** for us. But can add subscriptions by **our own**.

every component must be declared in one `NgModule` before it can be used in a template

The **recommended way** in Angular 2, and in many modern web frameworks (such as React), is to adopt a pattern of **one-way data binding**. That is, your **data flows only down through components**. If you need to **make changes**, you **emit events** that cause changes to happen **at the top** which then trickle down.

By **addyosami**

cheatography.com/addyosami/

Not published yet.
Last updated 25th January, 2017.
Page 2 of 2.