Cheatography

Angular Cheat Sheet Cheat Sheet

by rajanvora via cheatography.com/141179/cs/30219/

ng commands

ng new *app-name*

(--routing) add routing

ng add *@angular/material*

(install and configure the Angular Material library)

ng build

(build your application and places directory called "dist")

ng serve *(build and serve on 4200)*

ng serve -o *(automatically open the browser)*

ng serve -p 666 -o *(use port 666)*

ng generate component *csr*

or ng g c *csr*

ng g c *csr* --flat *(generate component without folder)*

ng g s *csr-data (generate service objects)*

ng g cl *models/csr*

(generates the csr class inside of the models folder)

ng g i *models/csr*

(generates the csr interface inside the models folder)

ng g p *shared/init-caps*

(generates the init-caps pipes)

Filters

Filters (cont)

data | json

Convert a JavaScript object into JSON string.

{{ json_e xpr ession | json : spacing}}

e.g.

```
<pre id=" def aul t-s pac ing ">
  {{ {'name ':' value'} | json:4 }}
</p re>
output:
{
    " nam e": " val ue"
}
```

array | limitT o:limit

Creates a new array containing only a spe cified number of elements in an array.

{{ limitT o_e xpr ession | limitTo : limit : begin}}

text | linky 1

Finds links in text input and turns them into html links.

* Requires ngSanitize Module

<span ng-bin d-h tml ="li nky _ex pre ssion | linky"> </s pan>

e,g,

```
<div ng-bin d-h tml ="sn ippet | linky">
</d iv>
<div ng-bin d-h tml ="sn ipp etW ith Sin gleURL |
linky: '_b lan k'">
</d iv>
<div ng-bin d-h tml ="sn ipp etW ith Sin gleURL |
linky: '_s elf':
{rel: 'nofol low '}">
</d iv>
```

string | lowercase

Converts string to lowercase.

{{ lowerc ase _ex pre ssion | lowerc ase}}

number | number *[:frac tio nSize]*

Formats a number as text.

If the input is not a number an empty string is returned.

{{ number _ex pre ssion | number : fracti onS ize}}

e.g.

Default format ting:

amount | currency ***[:symbol]***

{{ curren cy_ exp ression | currency : symbol : fracti onS ize}}

e.g.

```
<span id="currency-no-fractions">
  {{amount | currency:"USD$":0}}
</span>
```

Formats a number as a currency (ie $1,234.56).

date | date ***[:format]***

{{ date_e xpr ession | date : format : timezone}}

e.g.

```
<span ng-non-bindable>
  {{1288323623006 | date:' MM/ dd/yyyy @ h:mma'}}
</span>
<span>
  {{'1288323623006' | date:' MM/ dd/yyyy @ h:mma'}}
</span>
```

output:

```
{{1288323623006 | date:' MM/ dd/yyyy @ h:mma'}}:
  12/ 15/ 202 1@1 1:40PM
```

array | filter :ex pre ssion

Selects a subset of items from array.

Expression takes string |O b j ec t|f un c t - ion()

{{ filter _ex pre ssion | filter : expression : comparator : anyPro per tyKey}}

e.g.

```
ng-rep eat ="friend in friends | filter :se arc - hTe xt"
```

By **rajanvora**
cheatography.com/rajanvora/

Published 15th December, 2021.
Last updated 15th December, 2021.

Filters (cont)

```
<span id='nu mbe r-d efa ult '>
  {{val | number}}
</s pan>
No fractions:
<sp an>
  {{val | number:0}}
</s pan>
Negative number:
<sp an>
  {{-val | number:4}}
</s pan>
```

array | orderB y:p red ica[te reverse]

```
Predicate is functi  on ( * )|  str  in g | Array.
Reverse is boolean
*{{ orderB y_e xpr ession | orderBy :
expression : reverse : compar ator}}*
e.g.
ng-rep eat ="friend in friends | orderB y:' -ag -
e'"
```

string | uppercase

```
Converts string to uppercase.
{{ upperc ase _ex pre ssion | upperc ase}}
<h1>{{title | upperc ase }}< /h1>
```

Forms

```
In Angular, there are 2 types: template-
driven(easier to use)
 and reacti ve( rec omm ended for large forms)
```

Templa te- driven:

```
Import FormsM odule in app.mo dule.ts
Sample:
```

.ts

```
import { Component, OnInit } from '@angu -
lar /core';
    @Co mpo nent({
                sel ector: 'app-n ewu ser',
templa teUrl: './new use r.c omp one nt.h tml',
styleUrls: ['./ne wus er.c om pon ent.scss']
   })
    export class Newuse rCo mponent implements
OnInit {
// Data
```

Forms (cont)

```
user = {username: '', email: '', password: ''};
constr uctor() { }
ngOnIn it(): void { }
/**
* Called when the user clicks in the " Reg ist er"
* button.
*/
onSubmit() {
consol e.l og( 'User: ', this.u ser.us ern ame);
consol e.l og( 'Email: ', this.u ser.em ail);
consol e.l og( 'Pa ssword: ', this.u ser.pa ssw -
ord);
}
}
```

.html

```
<di v>
<form novalidate #regis ter For m="n gFo rm"
(ngSub mit )="o nSu bmi t() ">
<!-- User -->
<p>
<ma t-f orm -fi eld>
<input
matInput
placeh old er= " Use rna me"
type="t ext "
[(ngMo del )]= " use r.u ser nam e"
name="u ser nam e"
#usern ame ="ng Mod el"
requir ed>
<ma t-error
*ngIf=
" use rna me.e rr ors ?.r equ ire d">
Username is required
</m at- err or>
</m at- for m-f iel d>
</p>
<!-- Email -->
```

Forms (cont)

```
<p>
<ma t-f orm -fi eld>
<input matInput
placeh old er= " Ema il"
type="e mai l"
[(ngMo del )]= " use r.e mai l"
name="e mai l"
#email ="ng Mod el"
requir ed>
<ma t-error
*ngIf= " ema il.e rr ors ?.r equ ire d">
Email is required
</m at- err or>
</m at- for m-f iel d>
</p>
<!-- Password -->
<p>
<ma t-f orm -fi eld>
<input
matInput
placeh old er= " Pas swo rd"
type="p ass wor d" [
(ngMod el) ]="u ser.pa ssw ord "
name="p ass wor d"
#passw ord ="ng Mod el"
requir ed>
<ma t-error
*ngIf= " pas swo rd.e rr ors ?.r equ ire d">
Password is required
</m at- err or>
</m at- for m-f iel d>
</p>
<p>
<button type="s ubm it"
mat-button
[disab led ]="r egi ste rFo rm.f or m.i nva lid ">
```

Forms (cont)

```
Register user
</b utt on>
                          </p>
</f orm>
</d iv>
this example uses two way binding and templa teRef
 (for regist erForm)
```

Reactive Forms:

```
import Reacti veF orm sModule in app.mo dule.ts
all the form logic and validation are done
controller
create a model
```

.ts

```
export class User {
      use rname: string;
      email: string;
      pas sword: string;
}
```

.ts

```
import { Component, OnInit } from '@angu -
lar /core';
import { User } from '../sh are d/u ser';
@Compo nent({
      sel ector: 'app-n ewu ser',
      tem pla teUrl: './new use r.c omp one nt.h -
tml',
      sty leUrls: ['./ne wus er.c om pon -
ent.scss']
})
export class Newuse rCo mponent implements OnInit
{
     // Register form
      reg ist erForm: FormGroup;
         // form for submitting the new user
      myUser: User;
         // the user generated from the form
      @Vi ewC hil d(' new use rForm') userFo rmD -
ire ctive:
             For mGr oup Dir ective;
     // reference to the form in the HTML
template,
     // in order to perform validation
     /*
```

Forms (cont)

```
       * The errors being shown for each field.
       * The form will automa tically update with
         * the errors stored here.
       */
        for mErrors = {
              'us ern ame': '',
'email': '',
'passw ord': ''
       }
     /*
         * Messages that will be shown in the
         * mat-error elements for each type of
validation error.
         */
        val ida tio nMe ssages = {
'usern ame': {
'requi red':
'Username is requir ed.',
                'mi nle ngth':
'Username must be at
least 3 characters long.',
'maxle ngth':
'Username cannot be
more than 20 characters long.'
     },
      'pa ssw ord': {
'requi red':
'Password is requir ed.',
'minle ngth':
'Password must be at
least 8 characters long.'
     },
      'em ail': {
'requi red':
'Email is requir ed.',
'email':
'Email not in valid format.'
```

Forms (cont)

```
     }
};

/**
* Inject a FormBu ilder for creating a FormGroup.
*/
constr uct or( private fb: FormBu ilder) {
    thi s.c rea teF orm();
}

/**
* Create the comment form with the injected
FormBu ilder.
*/
create Form(){
      thi s.r egi ste rForm = this.f b.g roup({
      use rname: ['',
            [Va lid ato rs.r eq uired,
              Val ida tor s.m inL eng th(3),
              Val ida tor s.m axL eng th(20)] ],
      pas sword: ['',
          [Va lid ato rs.r eq uired,
            Val ida tor s.m inL eng th(8)] ],
      email: ['',
        [Va lid ato rs.r eq uired,
          Val ida tor s.e mail] ],
});
this.r egi ste rFo rm.v al ueC han ges.su bsc ribe(
     data => this.o nVa lue Cha nge d());
// every time a value changes inside the form, the
// onValu eCh anged() method will be triggered
this.o nVa lue Cha nged();
// reset validation messages
}
/**
* Validate the form after a value change.
*/
onValu eCh anged() {
      if( !th is.r eg ist erForm) { return; }
```

Forms (cont)

```
     // in case the form hasn't been created yet
      const form = this.r egi ste rForm;
           // the form values are constantly
changing,
         // that's why we have to take a snapshot
// Validate the form
     for (const field in this.f orm Errors) {
         // Iterate the form field by field
         if (this.f or mEr ror s.h asO wnP rop -
ert y(f ield)) {
this.f orm Err ors [field] = '';
// clear previous error message (if any)
const control = form.g et( field);
if (control && contro l.dirty
&& !contr ol.v alid) {
// If this form field has been
// touched an it's not valid
const messages =
this.v ali dat ion Mes sag es[ field];
for (const key in contro l.e rrors) {
if (contr ol.e rr ors.ha sOw nPr ope rty (key)) {
// Add the corres ponding error messages
// to the array of form errors.
// The form mat-error elements will update
// immediatly with the new form errors.
this.f orm Err ors [field] +=
messag es[key] + ' ';
                                    }
}
      }
             }
       }
}
/**
* Called when the user clicks the " Sub mit "
button in the form
*/
onSubm it(){
   // Create a User object from the form data
```

Forms (cont)

```
this.m yUser = this.r egi ste rFo rm.v alue;
   // TODO: send the form data to somewhere else
   // Reset the form
                thi s.r egi ste rFo rm.r eset({
username: '',
email: '',
password: ''
});
this.u ser For mDi rec tiv e.r ese tFo rm();
}
}
```

.html

```
<di v>
    <form
        nov alidate
        [fo rmG rou p]= " reg ist erF orm "
        #ne wus erF orm ="ng For m"
        (ng Sub mit )="o nSu bmi t() ">
        <p>
<ma t-f orm -fi eld>
<input
matInput
formCo ntr olN ame ="us ern ame "
placeh old er= " Use rna me"
type="t ext "
requir ed>
<ma t-error
*ngIf= " for mEr ror s.u ser nam e">
{{formErrors.username}}
</m at- err or>
</m at- for m-f iel d>
</p>
<p>
<ma t-f orm -fi eld>
<input
matInput
```

Forms (cont)

```
formCo ntr olN ame ="pa ssw ord "
placeh old er= " Pas swo rd"
type="p ass wor d"
requir ed>
<ma t-error
*ngIf= " for mEr ror s.p ass wor d">
{{formErrors.password}}
</m at- err or>
  </m at- for m-f iel d>
</p>
<p>
<ma t-f orm -fi eld>
<input
matInput
formCo ntr olN ame ="em ail "
placeh old er= " Ema il"
type="e mai l"
requir ed>
<ma t-error
*ngIf= " for mEr ror s.e mai l">
{{formErrors.email}}
</m at- err or>
              </m at- for m-f iel d>
      </p>
    <button type="s ubm it"
        [di sab led ]="r egi ste rFo rm.i nv ali d"
        mat -ra ise d-b utton
        col or= " pri mar y">
Submit
    </b utt on>
    </f orm>
</d iv>
```

Directives

```
ng-app="plaintext"
ng-bin d[- htm l-u nsafe]=exp res sio n"
ng-bin d-t emp -
late="string{{expression}}string{{expression}}"
ng-change=" exp res sio n"
ng-che cked=" boo lea n"
ng-cla ss[ -ev en| -odd]=er ing |ob jec t"
ng-[db l]c lick=exp res sio n"
ng-cloak=" boo lea n"
ng-con tro ller=pla int ext "
ng-dis abled=" boo lea n"
<fo rm| ng-formame="p lai nte xt">ng-form=" -
pla int ext "
ng-hid e|show=" boo lea n"
ng-href="plaintext{{string}}"
ng-inc lude=" str ing<ng -in cludesrc="st -
rin g"
  onload="expression" autosc rol l="e xpr ess -
ion ">
ng-init=" exp res sio n"
<input ng-pat tern=" /r  ege  x/ "
  ng-min  length ng-max  length ng-req  uired
<input ng-list=" del imi ter |re gex ">
<input type="c  he c k ng-tru  e- value=
pl  ain  te x t "
  ng-fal  se - v alue= ain  te x t ">
ng-model=" exp res sio n"
ng-mou sedown=" exp res sio n"
ng-mou see nter=exp res sio n"
ng-mou sel eave=exp res sio n"
ng-mou semove=" exp res sio n"
ng-mou seover=" exp res sio n"
ng-mou seup=" exp res sio n"
<select ng-mul  tiple
ng-non -bi ndable
ng-opt ions=" select [as label] [group by group]
  for ([key,] value) in object |ar ray "
ng-plu ral ize |<n g-p lur alize= " num -
ber "
  when="object" offset ="nu mbe r">
ng-rea donly=" exp res sio n"
ng-repeat=" ([key,] value) in object |ar ray "
<option ng-sel ected=" boo lea n">
ng-src=" str ing "
```

Cheatography

Angular Cheat Sheet Cheat Sheet

by rajanvora via cheatography.com/141179/cs/30219/

Directives (cont)

ng-style=" str ing |ob jec t"

ng-submit=" exp res sio n"

ng-switch=" exp res sio n"**>|switch** on=" exp - res sio n">

ng-swi tc h - When= ain te x t "

ng-swi tc h - de fault

ng-vie w|< ng- vie w>

ng-bin d-html= exp res sio n"

Routing

With routing, you can introduce navigation between screens
(actually, between Angular compon ents) in your app.
Define routes in **app-ro uti ng.m od ule.ts**
Instan tiate the router in html as :
<ro ute r-o utl et> </r out er- out let>
To redirect the user to a defined route
use the **routerLink** directive

Data Binding

Angular components are defined in three files:
an HTML file for the layout (view),
a TypeScript file for the logic (contr oller), and a CSS file for the style.
One-way data binding is the mechanism for rendering in the
view objects defined in the controller (property binding)
and for allowing the view to call methods in the con troller (event binding).
Two-way data binding, where using the notation [(obje ct)],
a bidire ctional relati onship between the view and
the controller is establ ished, so any changes on the bound
object from the controller will be reproduced in the
view and vice versa.

Structural Directives

Template-Reference Variables

Inside the template of a component, we can assign a
ref erence to an HTML element so we can access its content
from other elements inside the DOM.

RxJS

A library for reactive programming in JS, an asy nch ron ous -pr ogr amming paradigm where it exists an
entity called Observ abl e<T >, which consists in a value
of type T that changes over time.
Our applic ation components can subscribe to this observ able,
bec oming observers by implem enting a callback which will be
tri ggered whenever the value changes.
The main method of observable objects is sub scr ibe (data => {}), which enables us to ask Angular to
notify us whenever the data changes.
Other intere sting functions: map, pipe, filter, delay..

Services

Components without UI
ng g s servic es/ dat afetch
Tell Angular to inject this service in all of the app components that ask for it, so let’s add it to
the providers section of the **app.mo dule.ts**file
To use it in any component of our app,
you just have to ask for it in the constr uctor.

Promises

```
Structural directives allow the developers to
include
 some code logic inside the HTML template in a very
quick
 and easy way in order to determine when and how
many
  times an HTML element has to be rendered
```

```
These are JS mechanism for async programming,
where a
  pending value is returned, which might be
available soon
  (re solve) or never (reject)
Promises allow you to specify what to do when the
answer to
 your request arrives or when something goes
wrong, and
  mea nwhile you can continue with the execution of
your program.
.servi ce.ts
@Injec table({
      pro vid edIn: 'root'
})
export class UserSe rvice {
```

Promises (cont)

```
      con str uctor() {}
      get Use rs(): Promis e<U ser> {
            return new Promise(
                  fun cti on( res olve, reject){
                        // get the data from
some API...
                         if( suc ces sful) {
                               // Data was
succes sfully retrieved
                                res olv e(r -
esult);
                        } else {
                               // There was an
error retrieving the data
                                rej ect (er -
ror);
                        }
});
       }
}
```

.ts (component that consumes the service)

```
@Compo nent({
      sel ector: 'app-u sers',
      tem pla teUrl: './use rs.c om pon ent.html',
      sty leUrls: ['./us ers.co mpo nen t.s css']
})
export class UsersC omp onent implements OnInit {
      con str uct or( private userSe rvice:
UserSe rvice) {}
      myU sers: User[];
      get Use rs(){
this.u ser Ser vic e.g etU sers()
.then( users => this.m yUsers = users)
                               // the Promise
was resolved
.catch (error => consol e.l og( error))
// the Promise was rejected
}
      ngO nIn it(): void {}
}
```

HTTP Request

```
The HttpClient class underlies the JavaScript
XMLHttpRequest
  object and returns an observable with the
server -re sponse
 body encoded as an object of the specified class.
Sample:
Image a GET request to http:/ /lo cal hos t:1 -
234 /items
  returns following JSON
[
{
" nam e": " Por celain cup",
" pri ce": 9.99,
" qua nti ty": 20
},
{
" nam e": " Photo frame",
" pri ce": 5.99,
" qua nti ty": 50
}
]
create a model to capture the data
export class Item {
name: string;
price: number;
quantity: number;
}
After importing HttpCl ien tModule in app.mo -
dule.ts
.servi ce.ts
import { Inject able, Inject } from '@angu -
lar /core';
import { Item } from '../sh are d/item'
import { Observable } from 'rxjs';
import { map, catchError } from 'rxjs/ ope rat -
ors';
import { HttpClient } from '@angu lar /co mmo -
n/h ttp';
@Injec table({
provid edIn: 'root'
})
export class ItemSe rvice {
baseURL = 'http: //l oca lho st: 1234/'
```

HTTP Request (cont)

```
/**
* Injects an HTTPClient and the BaseURL
* into the service.
* @param http HTTPClient used for making
* HTTP requests to the backend.
*/
constr uct or( private http: HttpCl ient) { }
/**
* Return the list of items from the API,
* as an array of Item objects
*/
getIte ms(): Observ abl e<I tem []> {
return this.h ttp.ge t<I tem []>
(this.b aseURL + 'items');
// make the HTTP GET request
}
/**
* Send a new item to the API, as
* an Item object
*/
addIte m(item: Item): Observ abl e<I tem> {
return this.h ttp.po st< Ite m>
(this.b aseURL + 'items', item);
// make the HTTP POST request
}
}
```

component to consume the service:

.ts

```
import { Component, OnInit } from '@angu -
lar /core';
import { ItemSe rvice } from '../se rvi -
ces /item';
import { Item } from '../sh are d/i tem';
@Compo nent({
selector: 'app-i tems',
templa teUrl: './ite ms.c om pon ent.html',
styleUrls: ['./it ems.co mpo nen t.s css']
})
```

HTTP Request (cont)

```
export class ItemsC omp onent implements OnInit {
constr uct or( private itemSe rvice: ItemSe rvice)
{}
myItems: Item[];
ngOnIn it(): void {
this.i tem Ser vic e.g etI tems()
.subsc rib e(items => this.m yItems = items,
// any time the value of the Observable
// changes, update the myItems object
error => consol e.l og( err or));
// if there is an error, log it to the
// console
}
}
```

Workflow

Steps to creating a reactive form:

1. Create the Domain Model
2. Create the Controller with references to View
3. Create the View
4. Add Valida tions
5. Add Submit Validation Control
6. Add Dynamic Behaviors