

LAPORAN PERTEMUAN KE-11
TENTANG Menggunakan Framework
Angular.js
PRAKTIKUM TEKNOLOGI WEB
(Dosen : Bagas Triaji, S.Kom.,M.kom.)



Disusun oleh :

Nama : Muhamad Adhi Winata

NIM : 215610059

Kelas : SI-2

PROGRAM STUDI SISTEM INFORMASI
PROGRAM SARJANA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA
YOGYAKARTA
2024

A. TUJUAN

Mahasiswa mampu membuat script menggunakan Framework Angular.js

B. DASAR TEORI

Apa Angular js?

- AngularJS adalah Framework JavaScript
- AngularJS adalah framework JavaScript yang ditulis dalam JavaScript.
- AngularJS didistribusikan sebagai file JavaScript, dan dapat ditambahkan ke halaman web dengan tag skrip:

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/angular.min.js"></script>
```

AngularJS Memperluas HTML

- AngularJS memperluas HTML dengan ng-directives .
- Arahan ng-app mendefinisikan aplikasi AngularJS.
- Direktif ng-model mengikat nilai kontrol HTML (input, pilih, textarea) ke data aplikasi.
- Arahan ng -bind mengikat data aplikasi ke tampilan HTML. contoh:

```
<!DOCTYPE html>
```

```
<html>
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.9/
angular.min.js"></script>
<body>
<div ng-app="">
<p>Input something in the input box:</p>
<p>Name: <input type="text" ng-model="name"></p>
<p ng-bind="name"></p>
</div>
</body>
</html>
```

- AngularJS dimulai secara otomatis saat halaman web dimuat.
- Direktif ng-app memberi tahu AngularJS bahwa elemen <div> adalah "pemilik" dari aplikasi AngularJS .
- Direktif ng-model mengikat nilai kolom input ke nama variabel aplikasi .
- Arahan ng-bind mengikat konten elemen <p> ke nama variabel aplikasi .

Ekspresi AngularJS

Ekspresi AngularJS dapat ditulis di dalam kurung ganda: .{{ expression }}

Ekspresi AngularJS juga dapat ditulis di dalam direktif: .ng-bind="expression"

AngularJS akan menyelesaikan ekspresi, dan mengembalikan hasilnya tepat di tempat ekspresi ditulis.

Ekspresi AngularJS sangat mirip dengan ekspresi JavaScript: Mereka dapat berisi literal,

operator, dan variabel.

Contoh: `{{ 5 + 5 }}` atau `{{ firstName + " " + lastName }}`

```
<!DOCTYPE html>
<html>
<script src="https://ajax.googleapis.com/
ajax/libs/angularjs/1.6.9/angular.min.js">
</script>
<body>
<div ng-app>
<p>Penjumlahan 5+5 = {{ 5 + 5 }}</p>
</div>
</body>
78
</html>
```

Filter AngularJS

AngularJS menyediakan filter untuk mengubah data:

currency: Memformat angka ke format mata uang.

date: Memformat tanggal ke format yang ditentukan.

filter: Pilih subset item dari array.

json: Memformat objek menjadi string JSON.

limitTo: Membatasi array/string, menjadi sejumlah elemen/karakter tertentu.

lowercase: Memformat string menjadi huruf kecil.

number: Memformat angka menjadi string.

orderBy: Memesan array dengan ekspresi.

uppercase Memformat string menjadi huruf besar.

C. PRAKTIK

1. Pada angularjs terdapt ekspresi penjumlahan dengan memberikan nilai awal dengan menggunakan `ng-init=""`. Buatlah program berikut:

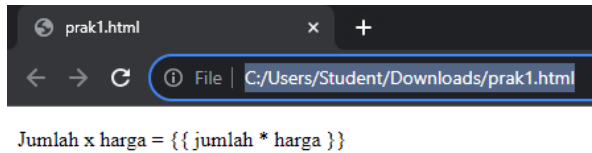
```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="" ng-init="jumlah=2;harga=5000">
<p>Jumlah x harga = {{ jumlah * harga }}</p>
```

```

</div>
</body>
</html>

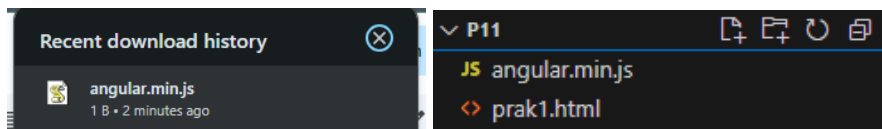
```

output :



<script src="angular.min.js"></script> anda bisa download dan diletakan di folder yang sama dengan programmnya.

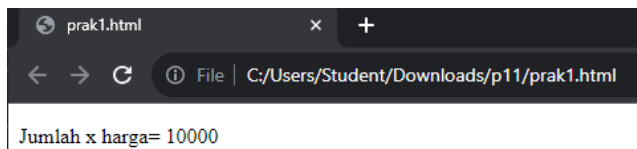
2. Tambahkan library "angular.min.js" dengan cara download dari <https://code.angularjs.org/1.8.2/> simpan ke folder kerja Anda



3. Cara yang sama bisa anda coba dengan:

<p>Jumlah x harga= </p>

Buka/eksekusi menggunakan browser, amati hasilnya :



4. Menggunakan ng-app, ng-controller, ng-model, direktif ng-app mendefinisikan aplikasi, direktif ng-controller mendefinisikan controller.

```

<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<p>Isikan Nama, Alamat</p>
<div ng-app="myApp" ng-controller="myCtrl">
Nama..: <input type="text" ng-model="nama"><br>
Alamat: <input type="text" ng-model="alamat"><br>
<br>
Yang tampil Nama : {{nama + " Alamat :" + alamat}}
</div>
<script>
var app = angular.module('myApp', []);
app.controller('myCtrl', function($scope) {
$scope.nama = "";

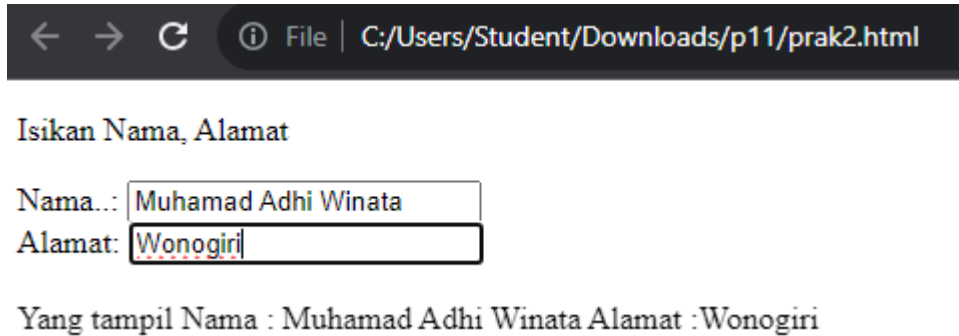
```

```

$scope.alamat= "";
});
</script>
</body>
</html>

```

output :



Isikan Nama, Alamat

Nama.: Muhamad Adhi Winata

Alamat: Wonogiri

Yang tampil Nama : Muhamad Adhi Winata Alamat : Wonogiri

5. Modul dan Pengontrol dalam File

Secara umum dalam aplikasi AngularJS untuk meletakkan modul dan pengontrol dalam file JavaScript. Berikut ini buatlah file "myApp.js" berisi definisi modul aplikasi, sedangkan "myCtrl.js" berisi pengontrol:

file praktik5.html

```

<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<p>Isikan Nama, Alamat</p>
<div ng-app="myApp" ng-controller="myCtrl">
Nama.: <input type="text" ng-model="nama"><br>
Alamat: <input type="text" ng-model="alamat"><br>
<br>
Yang tampil Nama : {{nama + " Alamat : " + alamat}}
</div>
<script src="myApp.js"></script>
<script src="myCtrl.js"></script>
</body>
</html>

```

buatlah file "myApp.js"

```
var app = angular.module('myApp', []);
```

dan file "myCtrl.js"

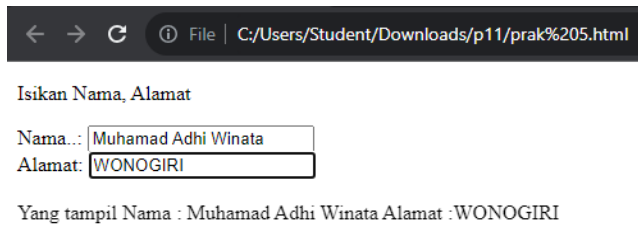
```

app.controller('myCtrl', function($scope) {
    $scope.nama = "";
    $scope.alamat= "";

```

```
});
```

Amati hasilnya



Isikan Nama, Alamat

Nama.:

Alamat:

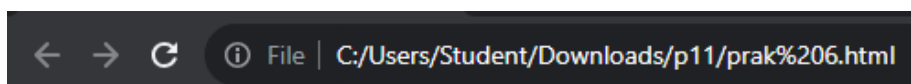
Yang tampil Nama : Muhamad Adhi Winata Alamat : WONOGIRI

Pembahasan : Program ini menggunakan AngularJS untuk menciptakan sebuah halaman web yang memungkinkan pengguna untuk menginputkan nama dan alamat. Informasi yang dimasukkan oleh pengguna akan secara otomatis ditampilkan kembali di halaman menggunakan teknik data binding dengan format `{{nama + " Alamat : " + alamat}}`. Dengan pengaturan modul "myApp" dan controller "myCtrl" yang terpisah dalam file myApp.js dan myCtrl.js, hal ini memungkinkan halaman untuk selalu terupdate secara langsung berdasarkan input yang diberikan oleh pengguna.

6. Menambahkan filter lowercase dan uppercase

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="utdiCtrl">
<p>{{ sinkat| uppercase }}</p>
<p>{{ perpanjangan| lowercase }}</p>
</div>
<script>
angular.module('myApp',
                []).controller('utdiCtrl',
function($scope) {
$scope.sinkat = "utdi",
$scope.perpanjangan = "UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA"
});
</script>
</body>
</html>
```

output :



UTDI

universitas teknologi digital indonesia

Pembahasan :

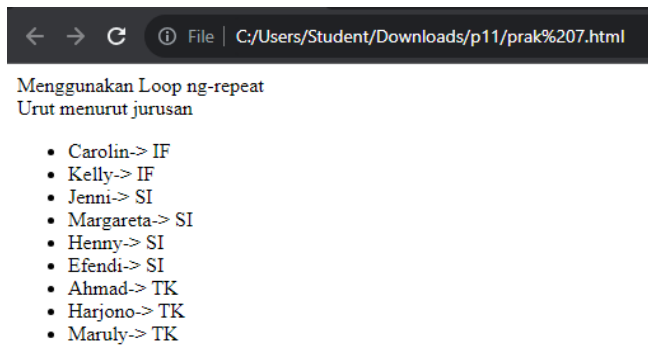
Program ini menggunakan AngularJS untuk melakukan transformasi teks berdasarkan aturan tertentu. Paragraf pertama menampilkan kata "utdi" dengan huruf besar, sedangkan paragraf kedua menampilkan kalimat "UNIVERSITAS TEKNOLOGI DIGITAL INDONESIA" dengan huruf kecil. Modul `myApp` dan controller `utdiCtrl` menginisialisasi variabel `sinkat` dan `perpanjangan`, yang kemudian ditampilkan dengan menerapkan filter huruf sesuai dengan kebutuhan.

7. Menggunakan ng-repeat dan orderBy

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="utdiCtrl">
Menggunakan Loop ng-repeat<br>
Urut menurut jurusan<br>
<ul>
<li ng-repeat="x in mhs | orderBy:'jurusan'">
{{ x.nama + '-> ' + x.jurusan }}
</li>
</ul>
</div>

<script>
angular.module('myApp', []).controller('utdiCtrl',
function($scope) {
$scope.mhs = [
{nama:'Jenni',jurusan:'SI'},
{nama:'Carolyn',jurusan:'IF'},
{nama:'Margareta',jurusan:'SI'},
{nama:'Henny',jurusan:'SI'},
{nama:'Ahmad',jurusan:'TK'},
{nama:'Kelly',jurusan:'IF'},
{nama:'Harjono',jurusan:'TK'},
{nama:'Maruly',jurusan:'TK'},
{nama:'Efendi',jurusan:'SI'}
];
});
</script>
</body>
</html>
```

Output :



Pembahasan :

Program ini menggunakan AngularJS untuk menampilkan daftar mahasiswa yang telah diurutkan berdasarkan jurusan mereka. Dengan menggunakan direktif `ng-repeat` dan filter `orderBy`, setiap entri mahasiswa ditampilkan dalam format "nama -> jurusan". Modul `myApp` dan controller `utdiCtrl` bertanggung jawab untuk menginisialisasi array `mhs` yang berisi objek-objek mahasiswa, yang kemudian diurutkan dan ditampilkan secara dinamis sesuai dengan pengaturan yang telah ditetapkan.

8. Menggunakan ng-repeat dan filter

```
<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="mhsCtrl">
<p>Isikan huruf di kolom input ini :
<input type="text" ng-model="test"></p>
<ul>
<li ng-repeat="x in mhs | filter:test">
{{ x }}
</li>
</ul>
</div>
<script>
angular.module('myApp', []).controller('mhsCtrl',
function($scope) {
$scope.mhs = [
'Carolin',
'Kelly',
'Jenni',
'Margareta',
```

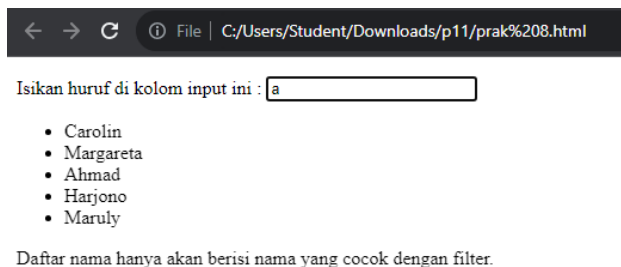


```

'Henny',
'Efendi',
'Ahmad',
'Harjono',
'Maruly',
];
});
</script>
<p>Daftar nama hanya akan berisi nama yang cocok dengan
filter.</p>
</body>
</html>

```

Output :



Pembahasan :

Program ini menggunakan AngularJS untuk menampilkan daftar nama mahasiswa yang bisa difilter berdasarkan teks yang dimasukkan oleh pengguna ke dalam input. Modul `myApp` dan controller `mhsCtrl` menginisialisasi array `mhs` yang berisi daftar nama-nama mahasiswa. Ketika pengguna memasukkan teks ke dalam input, daftar nama mahasiswa yang cocok dengan teks tersebut akan langsung ditampilkan secara dinamis dengan menggunakan fitur filter `ng-model`.

9. praktik 9

```

<!DOCTYPE html>
<html>
<script src="angular.min.js"></script>
<body>
<div ng-app="myApp" ng-controller="mhsCtrl">
<table border="1">
<tr>
<th ng-click="urutkan('nama') ">NAMA</th>
<th ng-click="urutkan('jurusan') ">JURUSAN</th>
</tr>
<tr ng-repeat="x in mhs | orderBy:urutMenurut">
<td>{{x.nama}}</td>

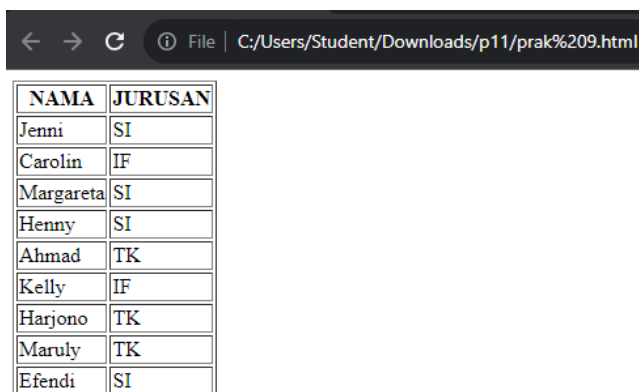
```

```

<td>{{x.jurusan}}</td>
</tr>
</table>
</div>
<script>
angular.module('myApp', []).controller('mhsCtrl',
function($scope) {
$scope.mhs = [
{nama:'Jenni', jurusan:'SI'},
{nama:'Carolyn', jurusan:'IF'},
{nama:'Margareta', jurusan:'SI'},
{nama:'Henny', jurusan:'SI'},
{nama:'Ahmad', jurusan:'TK'},
{nama:'Kelly', jurusan:'IF'},
{nama:'Harjono', jurusan:'TK'},
{nama:'Maruly', jurusan:'TK'},
{nama:'Efendi', jurusan:'SI'}
];
$scope.urutkan = function(x) {
$scope.urutMenurut = x;
}
});
</script>
</body>
</html>

```

output :



NAMA	JURUSAN
Jenni	SI
Carolyn	IF
Margareta	SI
Henny	SI
Ahmad	TK
Kelly	IF
Harjono	TK
Maruly	TK
Efendi	SI

Pembahasan :

Program di atas merupakan sebuah halaman HTML yang menggunakan AngularJS untuk menampilkan data mahasiswa dalam sebuah tabel. AngularJS digunakan untuk mengatur kontroler ('mhsCtrl') yang memanipulasi data mahasiswa yang disimpan dalam array '\$scope.mhs'. Setiap objek mahasiswa memiliki properti 'nama' dan

`jurusan`. Tabel menampilkan daftar mahasiswa yang dapat diurutkan berdasarkan kolom `NAMA` atau `JURUSAN` saat kolom tersebut diklik, dengan menggunakan fungsi `urutkan` yang mengatur variabel `\$scope.urutMenurut`. AngularJS juga memungkinkan penggunaan direktif `ng-repeat` untuk mengulang data mahasiswa dalam baris-baris tabel sesuai dengan array yang ada.

D. LATIHAN

Mengerjakan dan membuat laporan tiap step pada tutorial

<https://angular.dev/tutorials/learn-angular> . Dari Introduction hingga Create Pipe

Komponen di Angular

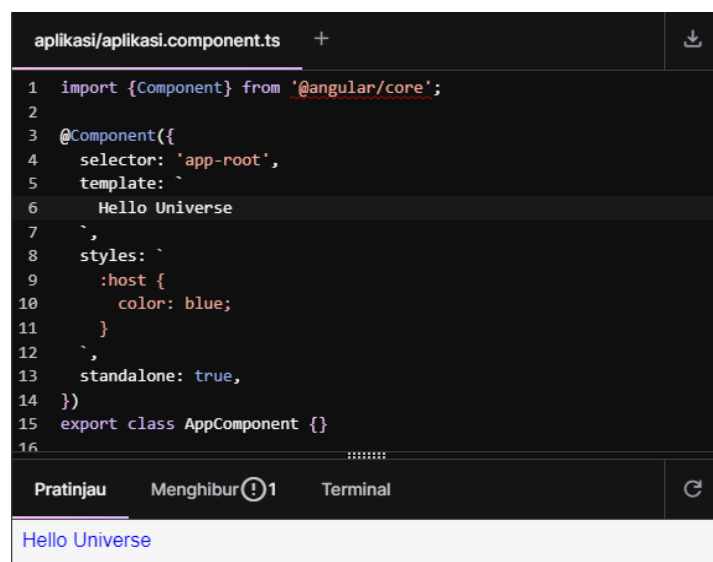
Komponen adalah blok bangunan dasar untuk aplikasi Angular apa pun. Setiap komponen memiliki tiga bagian:

- Kelas TypeScript
- Templat HTML
- gaya CSS

Dalam aktivitas ini, Anda akan mempelajari cara memperbarui template dan gaya komponen.

1. Perbarui templat komponen

Perbarui template properti untuk dibaca Hello Universe



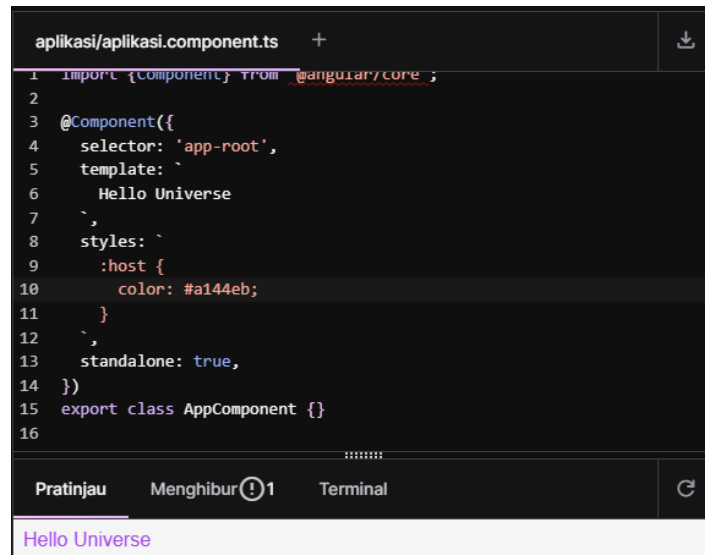
```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Hello Universe
7   `,
8   styles: `
9     :host {
10       color: blue;
11     }
12   `,
13   standalone: true,
14 })
15 export class AppComponent {}
16
```

Pratinjau Menghibur 1 Terminal

Hello Universe

2. Perbarui gaya komponen

Perbarui nilai gaya dan ubah color properti dari blue menjadi #a144eb.



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Hello Universe
7   `,
8   styles: `
9     :host {
10       color: #a144eb;
11     }
12   `,
13   standalone: true,
14 })
15 export class AppComponent {}
16
```

Pratinjau Menghibur 1 Terminal

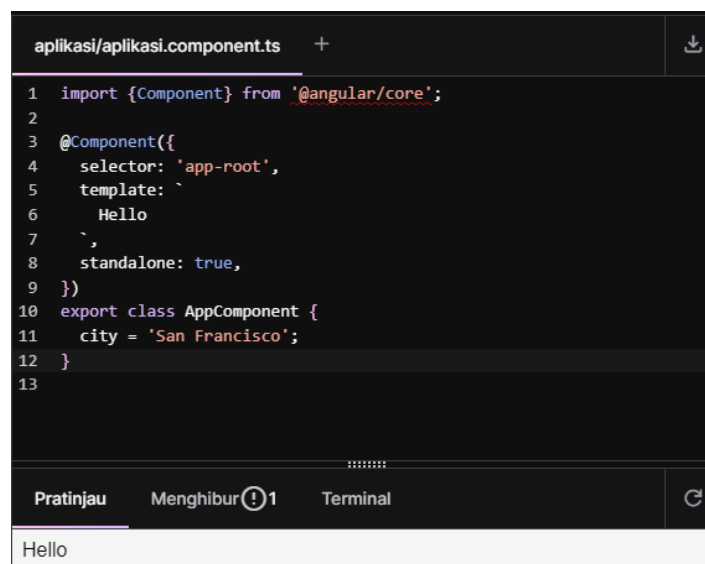
Hello Universe

Memperbarui Kelas Komponen

Di Angular, logika dan perilaku komponen ditentukan di kelas TypeScript komponen. Dalam aktivitas ini, Anda akan mempelajari cara memperbarui kelas komponen dan cara menggunakan interpolasi .

1. Tambahkan properti bernamacity

Perbarui kelas komponen dengan menambahkan properti yang dipanggil cityke AppComponentkelas tersebut.



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Hello
7   `,
8   standalone: true,
9 })
10 export class AppComponent {
11   city = 'San Francisco';
12 }
13
```

Pratinjau Menghibur 1 Terminal

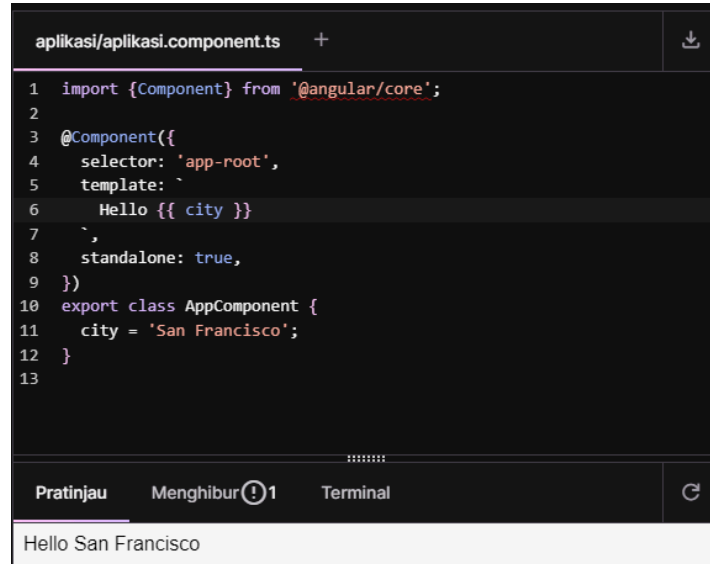
Hello

Properti ini citybertipe stringtetapi Anda dapat menghilangkan tipe tersebut karena inferensi tipe di TypeScript . Properti ini citydapat digunakan di AppComponentkelas dan dapat direferensikan dalam templat komponen.

Untuk menggunakan properti kelas dalam templat, Anda harus menggunakan sintaksis `{{ }}`.

2. Perbarui templat komponen

Perbarui `templateproperty` agar sesuai dengan HTML berikut:



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Hello {{ city }}
7   `,
8   standalone: true,
9 })
10 export class AppComponent {
11   city = 'San Francisco';
12 }
13
```

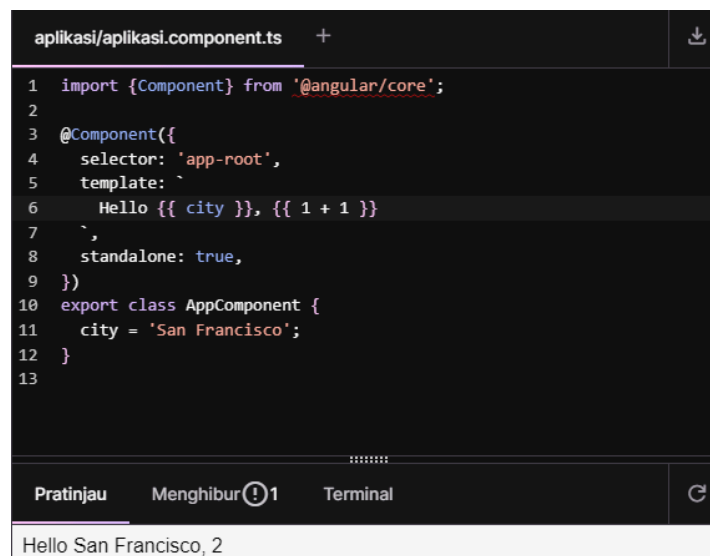
Pratinjau Menghibur 1 Terminal

Hello San Francisco

Ini adalah contoh interpolasi dan merupakan bagian dari sintaks templat Angular. Ini memungkinkan Anda melakukan lebih dari sekadar memasukkan teks dinamis ke dalam template. Anda juga dapat menggunakan sintaks ini untuk memanggil fungsi, menulis ekspresi, dan lainnya.

3. Lebih banyak latihan dengan interpolasi

Coba ini - tambahkan satu set lagi `{{ }}` yang isinya `1 + 1`:



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     Hello {{ city }}, {{ 1 + 1 }}
7   `,
8   standalone: true,
9 })
10 export class AppComponent {
11   city = 'San Francisco';
12 }
13
```

Pratinjau Menghibur 1 Terminal

Hello San Francisco, 2

Angular mengevaluasi konten `{{ }}` dan merender output dalam template.

Komponen Penyusun

Anda telah belajar memperbarui templat komponen, logika komponen, dan gaya komponen, namun bagaimana Anda menggunakan komponen dalam aplikasi Anda?

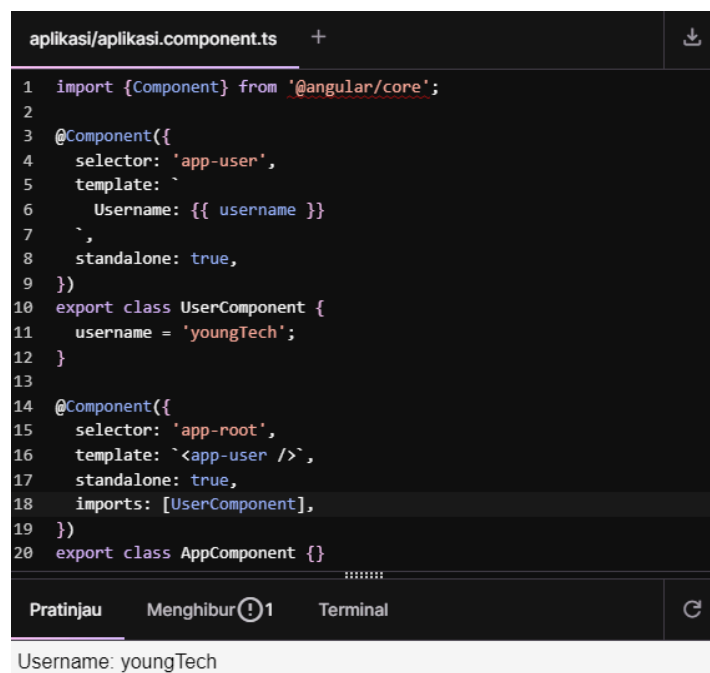
Properti `selector` konfigurasi komponen memberi Anda nama untuk digunakan saat mereferensikan komponen di templat lain. Anda menggunakan `selector` tag HTML seperti itu, misalnya `app-user` di `<app-user />` template.

Dalam aktivitas ini, Anda akan mempelajari cara menyusun komponen.

Dalam contoh ini, ada dua komponen `UserComponent` dan `AppComponent`.

1. Tambahkan referensi ke `UserComponent`

Perbarui `AppComponent` template untuk menyertakan referensi ke `UserComponent` yang menggunakan pemilih `app-user`. Pastikan untuk menambahkan `UserComponent` array `imports` `AppComponent`, ini membuatnya tersedia untuk digunakan dalam `AppComponent` template.



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-user',
5   template: `
6     Username: {{ username }}
7   `,
8   standalone: true,
9 })
10 export class UserComponent {
11   username = 'youngTech';
12 }
13
14 @Component({
15   selector: 'app-root',
16   template: `<app-user />`,
17   standalone: true,
18   imports: [UserComponent],
19 })
20 export class AppComponent {}
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2
```

```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-user',
5   template: `
6     Username: {{ username }}
7   `,
8   standalone: true,
9 })
10 export class UserComponent {
11   username = 'youngTech';
12 }
13
14 @Component({
15   selector: 'app-root',
16   template: `<section><app-user /></section>`,
17   standalone: true,
18   imports: [UserComponent],
19 })
20 export class AppComponent {}
.....
Pratinjau Menghibur 1 Terminal
Username: youngTech
```

Aliran Kontrol di Komponen -@if

Memutuskan apa yang akan ditampilkan di layar bagi pengguna adalah tugas umum dalam pengembangan aplikasi. Seringkali, keputusan dibuat secara terprogram menggunakan kondisi.

Untuk mengekspresikan tampilan bersyarat dalam templat, Angular menggunakan @ifsintaks templat.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan kondisional dalam templat. Sintaks yang memungkinkan tampilan bersyarat elemen dalam templat adalah @if. Berikut ini contoh cara menggunakan sintaks @if dalam sebuah komponen:

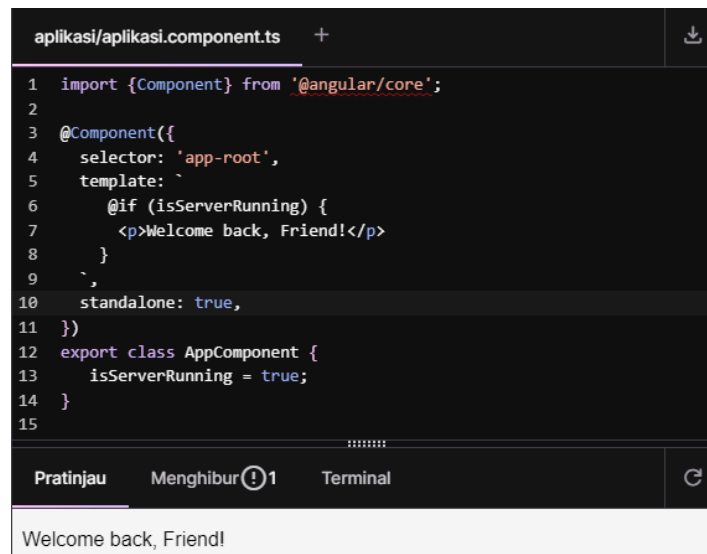
```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     @if (isLoggedIn) {
7       <p>Welcome back, Friend!</p>
8     }
9   `,
10   standalone: true,
11 })
12 export class AppComponent {
13   isLoggedIn = true;
14 }
15
.....
Pratinjau Menghibur 1 Terminal
Welcome back, Friend!
```

Dua hal yang perlu diperhatikan:

- Ada `@`awalan untuk if karena ini adalah tipe sintaks khusus yang disebut Sintaks templat sudut
- Untuk aplikasi yang menggunakan v16 dan yang lebih lama, silakan merujuk ke dokumentasi Angular untuk NgIf untuk informasi lebih lanjut.

1. Buat properti bernama `isServerRunning`

Di `AppComponent` kelas, tambahkan boolean properti bernama `isServerRunning`, atur nilai awal menjadi `true`.



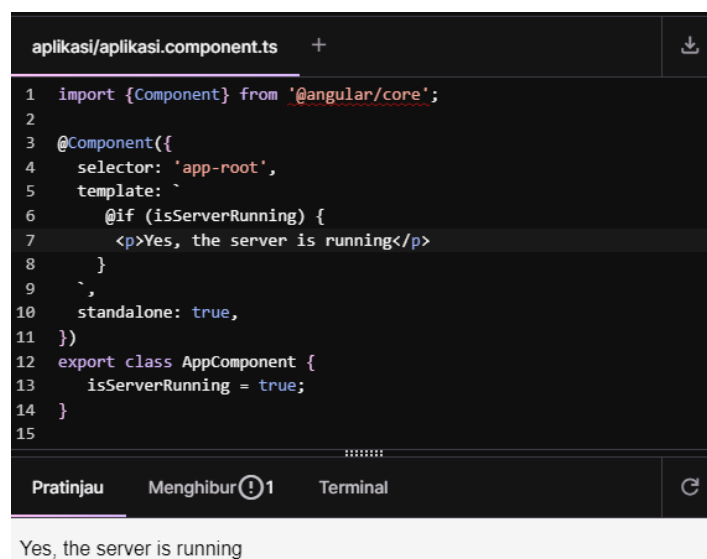
```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     @if (isServerRunning) {
7       <p>Welcome back, Friend!</p>
8     }
9   `,
10  standalone: true,
11 })
12 export class AppComponent {
13   isServerRunning = true;
14 }
15
```

Pratinjau Menghibur 1 Terminal

Welcome back, Friend!

2. Gunakan `@if` dalam templat

Perbarui templat untuk menampilkan pesan `Yes, the server is running` jika nilainya `isServerRunning` adalah `true`.



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     @if (isServerRunning) {
7       <p>Yes, the server is running</p>
8     }
9   `,
10  standalone: true,
11 })
12 export class AppComponent {
13   isServerRunning = true;
14 }
15
```

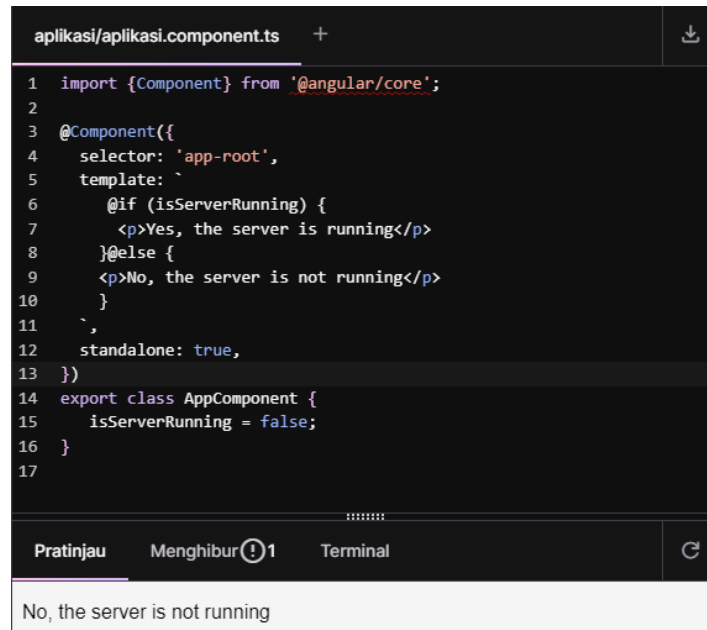
Pratinjau Menghibur 1 Terminal

Yes, the server is running

3. Gunakan `@else` dalam templat

Sekarang Angular mendukung sintaks templat asli untuk mendefinisikan kasus lain dengan `@else` sintaks tersebut. Perbarui templat untuk menampilkan pesan No, the server is not running seperti kasus lain.

Berikut ini contohnya:



```
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     @if (isServerRunning) {
7       <p>Yes, the server is running</p>
8     }@else {
9       <p>No, the server is not running</p>
10    }
11  `,
12   standalone: true,
13 })
14 export class AppComponent {
15   isServerRunning = false;
16 }
17
```

Pratinjau Menghibur 1 Terminal

No, the server is not running

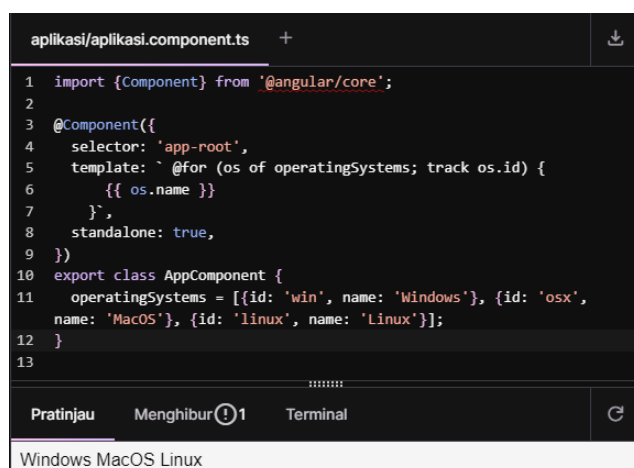
Aliran Kontrol di Komponen -@for

Seringkali ketika membangun aplikasi web, Anda perlu mengulangi beberapa kode beberapa kali - misalnya, jika diberi serangkaian nama, Anda mungkin ingin menampilkan setiap nama dalam sebuah `<p>` tag.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan `@for` elemen berulang dalam templat.

Sintaks yang memungkinkan elemen berulang dalam templat adalah `@for`.

Berikut ini contoh cara menggunakan sintaks `@for` dalam sebuah komponen:



```
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: ` @for (os of operatingSystems; track os.id) {
6     {{ os.name }}
7   }`,
8   standalone: true,
9 })
10 export class AppComponent {
11   operatingSystems = [{id: 'win', name: 'Windows'}, {id: 'osx',
12     name: 'MacOS'}, {id: 'linux', name: 'Linux'}];
13 }
14
```

Pratinjau Menghibur 1 Terminal

Windows MacOS Linux

1. Tambahkan usersproperty

Di AppComponentkelas, tambahkan properti bernama usersyang berisi pengguna dan nama mereka.

```
aplikasi/aplikasi.component.ts +
4 selector: 'app-root',
5 template: `@for (os of operatingSystems; track os.id) {
6   {{ os.name }}
7 }`,
8 standalone: true,
9 })
10 export class AppComponent {
11   operatingSystems = [{id: 'win', name: 'Windows'}, {id: 'osx',
12     name: 'MacOS'}, {id: 'linux', name: 'Linux'}];
13   users = [{id: 0, name: 'Sarah'}, {id: 1, name: 'Amy'}, {id: 2,
14     name: 'Rachel'}, {id: 3, name: 'Jessica'}, {id: 4, name:
15     'Poornima'}]
16 }
```

2. Perbarui templatnya

Perbarui templat untuk menampilkan setiap nama pengguna dalam pelemen menggunakan @forsintaks templat.

```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `@for (user of users; track user.id) {
6     <p>{{ user.name }}</p>
7   }`,
8   standalone: true,
9 })
10 export class AppComponent {
11   operatingSystems = [{id: 'win', name: 'Windows'}, {id: 'osx',
12     name: 'MacOS'}, {id: 'linux', name: 'Linux'}];
13   users = [{id: 0, name: 'Sarah'}, {id: 1, name: 'Amy'}, {id: 2,
14     name: 'Rachel'}, {id: 3, name: 'Jessica'}, {id: 4, name:
15     'Poornima'}]
16 }
```

Pratinjau Menghibur 1 Terminal

Sarah
Amy
Rachel
Jessica
Poornima

Catatan: penggunaan trackdiperlukan, Anda dapat menggunakan idatau pengenalan unik lainnya.

Pengikatan Properti di Angular

Pengikatan properti di Angular memungkinkan Anda menetapkan nilai untuk properti elemen HTML, komponen Angular, dan lainnya.

Gunakan pengikatan properti untuk menetapkan nilai properti dan atribut secara dinamis. Anda dapat melakukan hal-hal seperti fitur tombol alih, mengatur jalur gambar secara terprogram, dan berbagi nilai antar komponen.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan pengikatan properti di templat.

Untuk mengikat atribut elemen, bungkus nama atribut dalam tanda kurung siku.

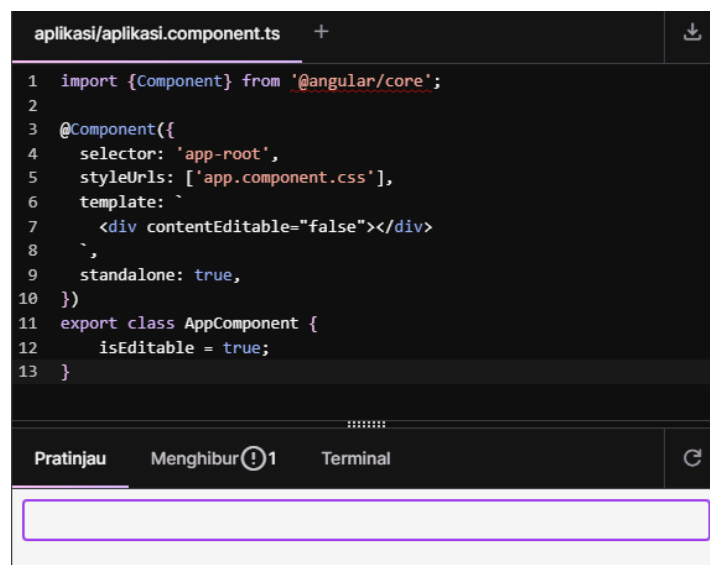
Berikut ini contohnya:

```
<img alt="photo" [src]="imageUrl">
```

Dalam contoh ini, nilai atribut src akan terikat pada properti kelas imageUrl. Nilai apa pun imageUrl yang dimilikinya akan ditetapkan sebagai src atribut tag img.

1. Tambahkan properti bernamaisEditable

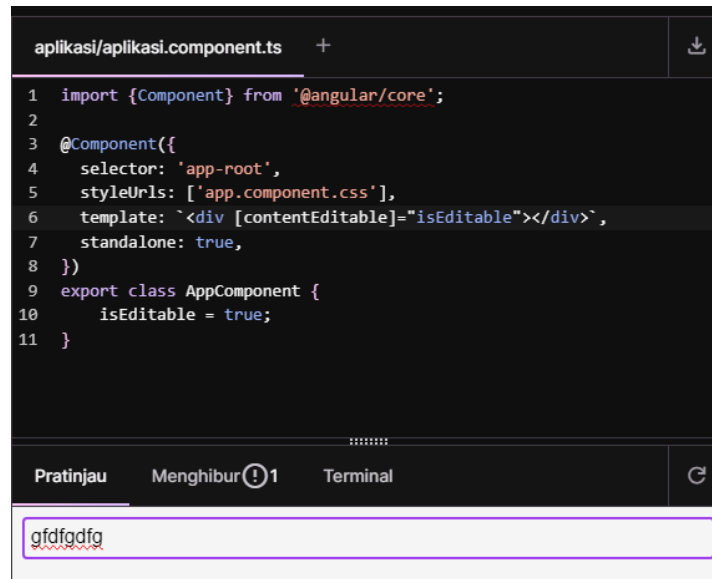
Perbarui kode app.component.ts dengan menambahkan properti ke AppComponent kelas yang dipanggil isEditable dengan nilai awal disetel ke true.



```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   styleUrls: ['app.component.css'],
6   template: `
7     <div contentEditable="false"></div>
8   `,
9   standalone: true,
10 })
11 export class AppComponent {
12   isEditable = true;
13 }
```

2. Mengikat contentEditable

Selanjutnya, ikat contentEditable atribut div ke isEditable properti dengan menggunakan [] sintaks.



```
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   styleUrls: ['app.component.css'],
6   template: `<div [contentEditable]="isEditable"></div>`,
7   standalone: true,
8 })
9 export class AppComponent {
10   isEditable = true;
11 }
```

Div sekarang dapat diedit. Kerja bagus

Event handling

Penanganan peristiwa mengaktifkan fitur interaktif pada aplikasi web. Ini memberi Anda kemampuan sebagai pengembang untuk merespons tindakan pengguna seperti penekanan tombol, pengiriman formulir, dan banyak lagi.

Dalam aktivitas ini, Anda akan mempelajari cara menambahkan event handler.

Di Angular Anda mengikat acara dengan sintaks tanda kurung (). Pada elemen tertentu, bungkus peristiwa yang ingin Anda ikat dengan tanda kurung dan atur pengendali peristiwa. Perhatikan buttoncontoh ini:



```
@Component({
  ...
  template: `<button (click)="greet()">`
})
class AppComponent {
  greet() {
    console.log('Hello, there 🐼');
  }
}
```

Dalam contoh ini, greet()fungsi tersebut akan berjalan setiap kali tombol diklik. Perhatikan bahwa greet()sintaksnya menyertakan tanda kurung tambahan.

1. Add an event handler

Tambahkan onMouseOverfungsi event handler di AppComponentkelas. Gunakan kode berikut sebagai implementasinya:

```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     <section>
7       There's a secret message for you, hover to reveal 🐞
8       {{ message }}
9     </section>
10  `,
11   standalone: true,
12 })
13 export class AppComponent {
14   message = '';
15
16   onMouseOver() {
17     this.message = 'Way to go 🎉';
18   }
19 }
20
```

Pratinjau Menghibur 1 Terminal

There's a secret message for you, hover to reveal 🐞

2. Bind to the template event

Perbarui kode templat app.component.ts untuk mengikat mouseover peristiwa elemen section.

```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2
3 @Component({
4   selector: 'app-root',
5   template: `
6     <section (mouseover)="onMouseOver()">
7       There's a secret message for you, hover to reveal 🐞
8       {{ message }}
9     </section>
10  `,
11   standalone: true,
12 })
13 export class AppComponent {
14   message = '';
15
16   onMouseOver() {
17     this.message = 'Way to go 🎉';
18   }
19 }
20
```

Pratinjau Menghibur 1 Terminal

There's a secret message for you, hover to reveal 🐞 Way to go 🎉

Component Communication with @Input

Terkadang pengembangan aplikasi mengharuskan Anda mengirim data ke dalam sebuah komponen. Data ini dapat digunakan untuk menyesuaikan komponen atau mungkin mengirim informasi dari komponen induk ke komponen anak.

Angular menggunakan konsep yang disebut Input. Hal ini mirip dengan propskerangka lainnya. Untuk membuat Inputproperty, gunakan @Inputdekorator. Dalam aktivitas ini, Anda akan mempelajari cara menggunakan @Inputdekorator untuk mengirimkan informasi ke komponen.

Untuk membuat Inputproperty, tambahkan @Inputdekorator ke properti kelas komponen:

```
class UserComponent {  
  @Input() occupation = "";  
}
```

Saat Anda siap meneruskan nilai melalui Input, nilai dapat diatur dalam templat menggunakan sintaks atribut. Berikut ini contohnya:

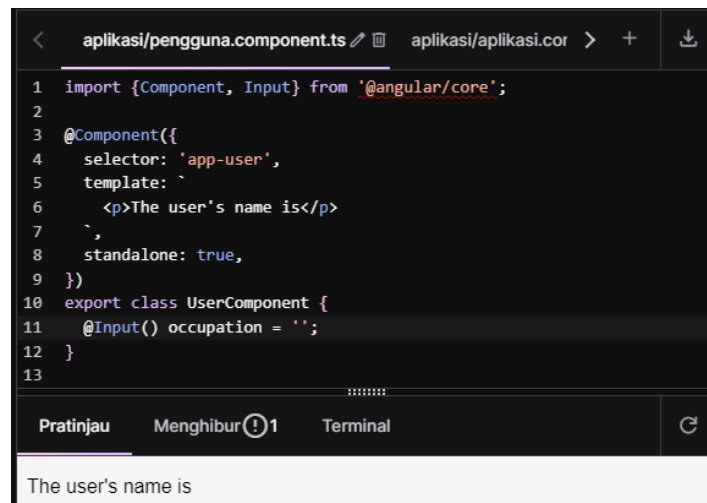
```
@Component({  
  ...  
  template: `<app-user occupation="Angular Developer"><app-user/>`  
})  
class AppComponent {}
```

Pastikan Anda mengikat properti occupation di file UserComponent.

```
@Component({  
  ...  
  template: `<p>The user's name is {{occupation}}</p>`  
})
```

1. Tentukan @Inputproperty

Perbarui kode user.component.ts untuk mendefinisikan Inputproperty di UserComponent file name. Untuk saat ini, tetapkan nilai awal menjadi empty string. Pastikan untuk memperbarui template untuk menginterpolasi nameproperty di akhir kalimat.



```
1 import {Component, Input} from '@angular/core';
2
3 @Component({
4   selector: 'app-user',
5   template: `
6     <p>The user's name is</p>
7   `,
8   standalone: true,
9 })
10 export class UserComponent {
11   @Input() occupation = '';
12 }
13
```

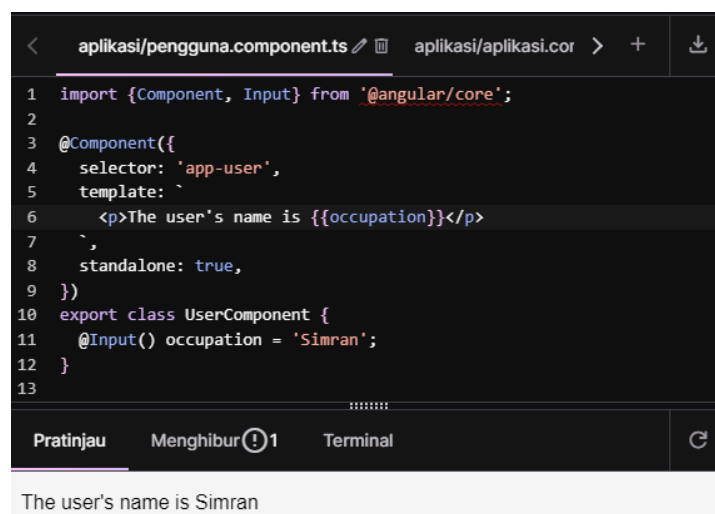
Pratinjau Menghibur 1 Terminal

The user's name is

2. Berikan nilai pada @Inputproperty

Perbarui kode app.component.ts untuk mengirimkan nameproperty dengan nilai "Simran".

Ketika kode telah berhasil diperbarui, aplikasi akan menampilkan The user's name is Simran.



```
1 import {Component, Input} from '@angular/core';
2
3 @Component({
4   selector: 'app-user',
5   template: `
6     <p>The user's name is {{occupation}}</p>
7   `,
8   standalone: true,
9 })
10 export class UserComponent {
11   @Input() occupation = 'Simran';
12 }
13
```

Pratinjau Menghibur 1 Terminal

The user's name is Simran

Komunikasi Komponen dengan @Output

Saat bekerja dengan komponen, mungkin diperlukan untuk memberi tahu komponen lain bahwa sesuatu telah terjadi. Mungkin tombol telah diklik, item telah ditambahkan/dihapus dari daftar, atau pembaruan penting lainnya telah terjadi. Dalam skenario ini komponen perlu berkomunikasi dengan komponen induk.

Angular menggunakan @Outputdekorator untuk mengaktifkan jenis perilaku ini.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan @Outputdekorator dan EventEmitterberkomunikasi dengan komponen.

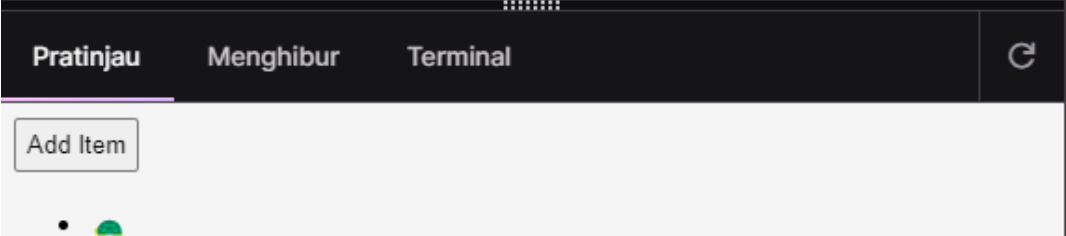
1. Tambahkan @Outputproperty

Perbarui child.component.ts dengan menambahkan properti keluaran yang disebut addItemEvent, pastikan untuk menyetel tipe EventEmitter menjadi string.

2. addItemMetode lengkap

Dalam child.component.ts memperbarui addItemmetode; gunakan kode berikut sebagai logikanya:

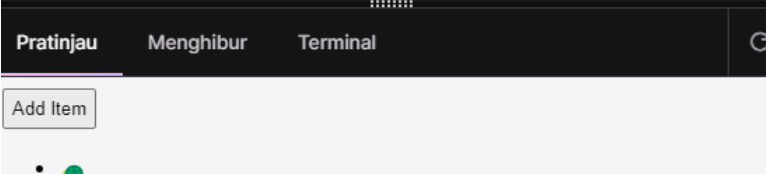
```
14 addItem() {
15   console.log('Button clicked. Emitting event with 🌈');
16   this.addItemEvent.emit('🌈');
17 }
18 }
19
```



3. Perbarui AppComponenttemplatnya

Dalam app.component.ts memperbarui templat untuk mendengarkan acara yang dipancarkan dengan menambahkan kode berikut:

```
5 @Component({
6   selector: 'app-root',
7   template: `
8     <app-child (addItemEvent)="addItem($event)"></app-child>
9     <ul>
10      <li *ngFor="let item of items">{{ item }}</li>
11    </ul>
12   `,
13   standalone: true,
14   imports: [CommonModule, ChildComponent]
15 })
16 export class AppComponent {
17   items: string[] = [];
18
19   addItem(item: string) {
20     console.log('Event received with item:', item);
21     this.items.push(item);
22   }
23 }
```



Penayangan yang Dapat Ditunda

Terkadang dalam pengembangan aplikasi, Anda mendapatkan banyak komponen yang perlu dijadikan referensi dalam aplikasi Anda, namun beberapa di antaranya tidak perlu langsung dimuat karena berbagai alasan.

Mungkin mereka berada di lipatan bawah yang terlihat atau merupakan komponen berat yang tidak berinteraksi sampai nanti. Dalam hal ini, kita dapat memuat beberapa sumber daya tersebut nanti dengan tampilan yang dapat ditangguhkan.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan tampilan yang dapat ditangguhkan untuk menunda pemuatan bagian templat komponen Anda.

1. Tambahkan `@deferblok` di sekitar komponen komentar

Di aplikasi Anda, halaman postingan blog memiliki komponen komentar setelah detail postingan.

Bungkus komponen komentar dengan `@deferblok` untuk menunda pemuatannya.

```
@defer {  
  <comments />  
}
```

Kode di atas adalah contoh cara menggunakan `@deferblok` dasar. Secara default `@defer` akan memuat `comments` komponen ketika browser dalam keadaan idle.

2. Tambahkan pengganti

Tambahkan satu `@placeholderblok` ke `@deferblok` tersebut. Blok tersebut `@placeholder` adalah tempat Anda meletakkan html yang akan ditampilkan sebelum pemuatan yang ditangguhkan dimulai. Konten dalam `@placeholderblok` dimuat dengan penuh semangat.

```
@defer {  
  <comments />  
} @placeholder {  
  <p>Future comments</p>  
}
```

3. Tambahkan blok pemuatan

Tambahkan satu `@loadingblok` ke `@deferblok` tersebut. Blok tersebut `@loading` adalah tempat Anda meletakkan html yang akan ditampilkan saat konten yang ditangguhkan sedang diambil secara aktif, tetapi belum selesai. Konten dalam `@loadingblok` dimuat dengan penuh semangat.

```
@defer {  
  <comments />  
} @placeholder {  
  <p>Future comments</p>  
} @loading {  
  <p>Loading comments...</p>  
}
```

4. Tambahkan pemicu area pandang

Tampilan yang dapat ditangguhkan memiliki sejumlah opsi pemicu. Tambahkan pemicu area pandang sehingga konten akan menunda pemuatan setelah memasuki area pandang.

5. Tambahkan durasi minimum

Keduanya `@placeholder` dan `@loading` bagian memiliki parameter opsional untuk mencegah terjadinya kedipan saat pemuatan terjadi dengan cepat. `@placeholder` memiliki `minimum` dan `@loading` memiliki `minimum` dan `after`. Tambahkan `minimum` durasi ke `@loading` blok sehingga akan dirender setidaknya selama 2 detik.

```
@defer {  
  <comments />  
} @placeholder {  
  <p>Future comments</p>  
} @loading (minimum 2s) {  
  <p>Loading comments...</p>  
}
```

6. Tambah isi

Pemicu area pandang paling baik digunakan saat Anda menunda konten yang berada cukup jauh di bawah halaman sehingga perlu di-scroll untuk melihatnya. Jadi mari tambahkan beberapa konten ke postingan blog kita. Anda dapat menulis sendiri, atau menyalin konten di bawah dan memasukkannya ke dalam `<article>` elemen.

```
10 <article>  
11   <p>Angular is my favorite framework, and this is why. Angular has the coolest  
    deferrable view feature that makes defer loading content the easiest and most ergonomic it  
    could possibly be. The Angular community is also filled with amazing contributors and  
    experts that create excellent content. The community is welcoming and friendly, and it  
    really is the best community out there.</p>  
12   <p>I can't express enough how much I enjoy working with Angular. It offers the best  
    developer experience I've ever had. I love that the Angular team puts their developers  
    first and takes care to make us very happy. They genuinely want Angular to be the best  
    framework it can be, and they're doing such an amazing job at it, too. This statement comes  
    from my heart and is not at all copied and pasted. In fact, I think I'll say these exact  
    same things again a few times.</p>  
13   <p>Angular is my favorite framework, and this is why. Angular has the coolest  
    deferrable view feature that makes defer loading content the easiest and most ergonomic it  
    could possibly be. The Angular community is also filled with amazing contributors and  
    experts that create excellent content. The community is welcoming and friendly, and it  
    really is the best community out there.</p>  
14   <p>I can't express enough how much I enjoy working with Angular. It offers the best  
    developer experience I've ever had. I love that the Angular team puts their developers  
    .....</p>  
Pratinjau  Menghibur  Terminal
```

How I feel about Angular

Angular is my favorite framework, and this is why. Angular has the coolest deferrable view feature that makes defer loading content the easiest and most ergonomic it could possibly be. The Angular community is also filled with amazing contributors and experts that create excellent content. The community is welcoming and friendly, and it really is the best community out there.

Setelah Anda menambahkan kode ini, sekarang gulir ke bawah untuk melihat pemuatan konten yang ditangguhkan setelah Anda menggulirnya ke area pandang.

Mengoptimalkan gambar

Gambar merupakan bagian besar dari banyak aplikasi, dan dapat menjadi kontributor utama terhadap masalah kinerja aplikasi, termasuk rendahnya skor Core Web Vitals .

Pengoptimalan gambar bisa menjadi topik yang kompleks, namun Angular menangani sebagian besarnya untuk Anda, dengan arahnya NgOptimizedImage. Dalam aktivitas ini, Anda akan mempelajari cara menggunakannya NgOptimizedImage untuk memastikan gambar Anda dimuat secara efisien.

1. Impor arahan NgOptimizedImage

Untuk memanfaatkan NgOptimizedImage arahan, impor terlebih dahulu dari @angular/common perpustakaan dan tambahkan ke imports array komponen.

```
@Component({
  standalone: true,

  imports: [NgOptimizedImage],
})
```

2. Perbarui atribut src menjadi ngSrc

Untuk mengaktifkan NgOptimizedImage direktif, tukar src atribut dengan ngSrc. Hal ini berlaku untuk sumber gambar statis (yaitu, src) dan sumber gambar dinamis (yaitu, [src]).

```
import { NgOptimizedImage } from '@angular/common';

@Component({
  standalone: true,
  selector: 'app-user',
  template: `
    <p>Username: {{ username }}</p>
    <p>Preferred Framework:</p>
    <ul>
      <li>
        Static Image:
        <img ngSrc="/assets/logo.svg" alt="Angular logo" width="32" height="32" />
      </li>
      <li>
        Dynamic Image:
        <img [ngSrc]="logoUrl" [alt]="logoAlt" width="32" height="32" />
      </li>
    </ul>
  `,
  imports: [NgOptimizedImage],
})
```

3. Tambahkan atribut lebar dan tinggi

Perhatikan bahwa dalam contoh kode di atas, setiap gambar memiliki atribut width dan height atribut. Untuk mencegah pergeseran tata letak , NgOptimizedImage arahan tersebut memerlukan kedua atribut ukuran pada setiap gambar.

Dalam situasi di mana Anda tidak bisa atau tidak ingin menentukan gambar statis height dan width gambar, Anda dapat menggunakan atribut untuk fill memberi tahu gambar agar bertindak seperti "gambar latar belakang", mengisi elemen yang memuatnya:

```
<div class="image-container">
  <img ngSrc="www.example.com/image.png" fill />
</div>
```

Catatan: Agar fill gambar dapat dirender dengan benar, elemen induknya harus diberi gaya dengan

position: "relative",

position: "fixed", atau

position: "absolute".

4. Prioritaskan gambar-gambar penting

Salah satu pengoptimalan terpenting untuk kinerja pemuatan adalah memprioritaskan gambar apa pun yang mungkin merupakan "elemen LCP", yang merupakan elemen grafis terbesar di layar saat laman dimuat. Untuk mengoptimalkan waktu pemuatan, pastikan untuk menambahkan priority atribut ke "gambar pahlawan" atau gambar lain yang menurut Anda dapat menjadi elemen LCP.

```
<img ngSrc="www.example.com/image.png" height="600" width="800" priority />
```

5. Opsional: Gunakan pemuat gambar

NgOptimizedImage memungkinkan Anda menentukan pemuat gambar, yang memberi tahu arahan tentang cara memformat URL untuk gambar Anda. Menggunakan pemuat memungkinkan Anda menentukan gambar Anda dengan URL relatif yang pendek:

```
1 import { Component } from '@angular/core';
2 import { NgOptimizedImage } from '@angular/common';
3
4 @Component({
5   standalone: true,
6   selector: 'app-user',
7   template: `
8     <p>Username: {{ username }}</p>
9     <p>Preferred Framework:</p>
10    <ul>
11      <div class="image-container">
12        <img ngSrc="www.example.com/image.png" height="600" width="800" priority />
13      </div>
14      <li>
15        Static Image:
16        <img ngSrc="/assets/logo.svg" alt="Angular logo" width="32" height="32" />
17      </li>
18    </ul>
19  `
20 })
21 export class UserComponent {
22   username = 'youngTech';
23   preferredFramework = 'Angular';
24 }
```

Pratinjau Menghibur Terminal

Username: youngTech

Preferred Framework:



Pemuat gambar lebih dari sekadar kenyamanan--mereka memungkinkan Anda menggunakan kemampuan penuh NgOptimizedImage. Pelajari lebih lanjut tentang pengoptimalan ini dan pemuat bawaan untuk CDN populer di sini .

Mengaktifkan Perutean

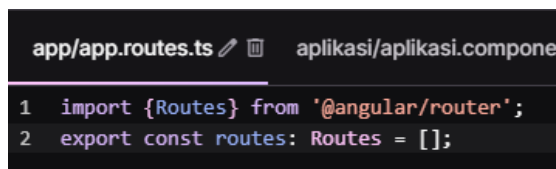
Untuk sebagian besar aplikasi, ada saatnya aplikasi memerlukan lebih dari satu halaman. Ketika saatnya tiba, perutean menjadi bagian besar dari kisah kinerja bagi pengguna.

Dalam aktivitas ini, Anda akan mempelajari cara menyiapkan dan mengonfigurasi aplikasi Anda untuk menggunakan Angular Router.

1. Buat file app.routes.ts

Di dalam app.routes.ts, buat perubahan berikut:

- a. Impor Routes dari @angular/router paket.
- b. Ekspor konstanta yang disebut routes sebagai Routes, tetapkan [] sebagai nilainya.

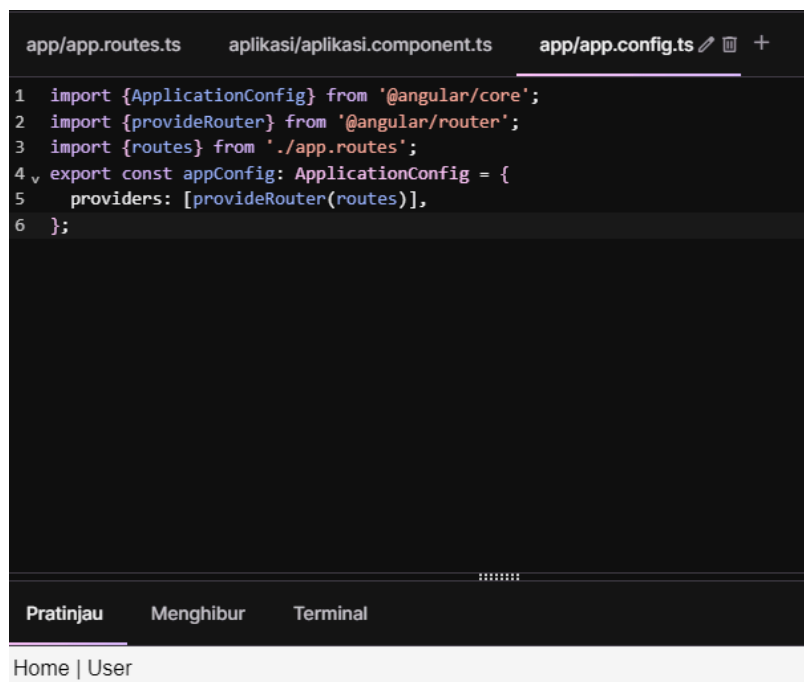


```
app/app.routes.ts  aplikasi/aplikasi.component.ts
1 import {Routes} from '@angular/router';
2 export const routes: Routes = [];
```

2. Tambahkan perutean ke penyedia

Di app.config.ts, konfigurasi aplikasi ke Angular Router dengan langkah-langkah berikut:

- a. Impor provideRouter fungsi dari @angular/router.
- b. Impor routes dari ./app.routes.ts.
- c. Panggil provideRouter fungsi yang routes diteruskan sebagai argumen dalam providers array.



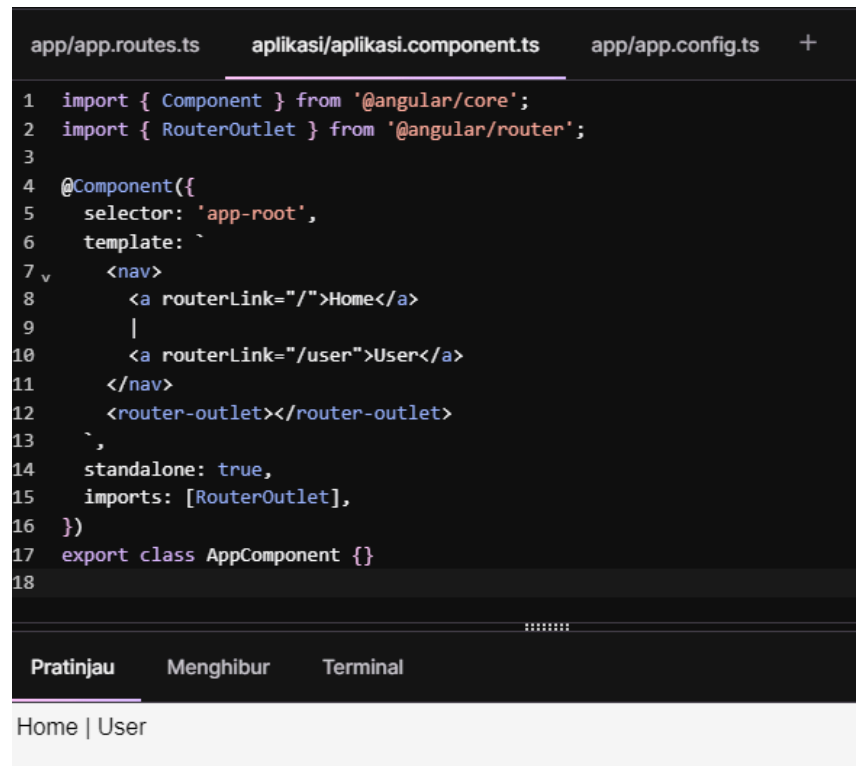
```
app/app.routes.ts  aplikasi/aplikasi.component.ts  app/app.config.ts
1 import {ApplicationConfig} from '@angular/core';
2 import {provideRouter} from '@angular/router';
3 import {routes} from './app.routes';
4 export const appConfig: ApplicationConfig = {
5   providers: [provideRouter(routes)],
6 };

Pratinjau  Menghibur  Terminal
Home | User
```

3. Impor RouterOutletkomponen

Terakhir, untuk memastikan aplikasi Anda siap menggunakan Angular Router, Anda perlu memberi tahu aplikasi di mana Anda mengharapkan router menampilkan konten yang diinginkan. Selesaikan itu dengan menggunakan RouterOutletarah dari @angular/router.

Perbarui templat AppComponentdengan menambahkan<router-outlet />



```
app/app.routes.ts  aplikasi/aplikasi.component.ts  app/app.config.ts  +
1  import { Component } from '@angular/core';
2  import { RouterOutlet } from '@angular/router';
3
4  @Component({
5    selector: 'app-root',
6    template: `
7    <nav>
8      <a routerLink="/">Home</a>
9      |
10     <a routerLink="/user">User</a>
11   </nav>
12   <router-outlet></router-outlet>
13 `
14   standalone: true,
15   imports: [RouterOutlet],
16 })
17 export class AppComponent {}
18
```

Pratinjau Menghibur Terminal

Home | User

Tentukan Rut

Sekarang setelah Anda menyiapkan aplikasi untuk menggunakan Angular Router, Anda perlu menentukan rutanya.

Dalam aktivitas ini, Anda akan mempelajari cara menambahkan dan mengonfigurasi rute dengan aplikasi Anda.

1. Tentukan rute masukapp.routes.ts

Di aplikasi Anda, ada dua halaman untuk ditampilkan: (1) Halaman Beranda dan (2) Halaman Pengguna.

Untuk menentukan rute, tambahkan objek rute ke routesarray yang app.routes.tsberisi:

Rute path(yang secara otomatis dimulai di jalur root (yaitu, /))

Rute componentyang ingin Anda tampilkan

```
app/app.routes.ts  app/home/home.component.ts  +
1 import { Routes } from '@angular/router';
2 import { HomeComponent } from './home/home.component';
3 import { UserComponent } from './user/user.component';
4
5 export const routes: Routes = [
6   {
7     path: '',
8     title: 'App Home Page',
9     component: HomeComponent,
10  },
11  {
12    path: 'user',
13    title: 'User Page',
14    component: UserComponent,
15  },
16 ];
17
```

Pratinjau Menghibur 1 Terminal

[Home](#) | [User](#)
Home Page

Kode di atas adalah contoh bagaimana cara HomeComponentmenambahkannya sebagai rute. Sekarang lanjutkan dan terapkan ini bersama dengan UserComponentdi taman bermain.

Gunakan 'user'untuk jalur UserComponent.

2. Tambahkan judul ke definisi rute

Selain menentukan rute dengan benar, Angular Router juga memungkinkan Anda mengatur judul halaman setiap kali pengguna bernavigasi dengan menambahkan properti titleke setiap rute.

```
app/app.routes.ts  app/home/home.component.ts  +
1 import { Component } from '@angular/core';
2
3 @Component({
4   standalone: true,
5   selector: 'app-home',
6   template: `
7     <div>Home Page</div>
8   `,
9 })
10 export class HomeComponent {}
```

Pratinjau Menghibur Terminal

[Home](#) | [User](#)
Home Page

Di `app.routes.ts`, tambahkan `title` properti ke rute default (`path: ''`) dan `userRoute`.
Berikut ini contohnya:

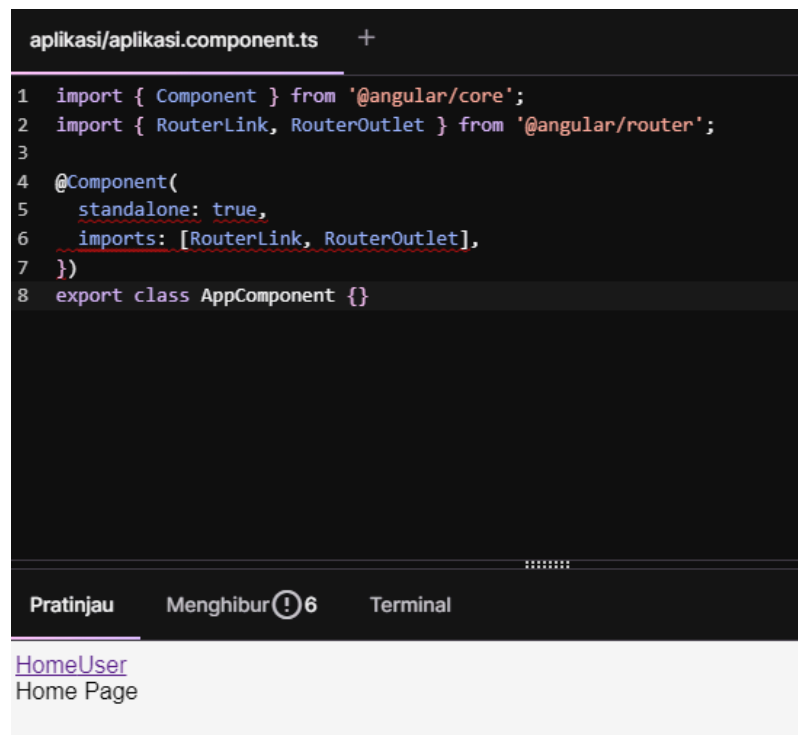
Gunakan RouterLink untuk Navigasi

Dalam keadaan aplikasi saat ini, seluruh halaman disegarkan ketika kita mengeklik tautan internal yang ada di dalam aplikasi. Meskipun hal ini mungkin tidak tampak signifikan pada aplikasi kecil, hal ini dapat menimbulkan implikasi kinerja pada halaman yang lebih besar dengan lebih banyak konten sehingga pengguna harus mendownload ulang aset dan menjalankan perhitungan lagi.

Dalam aktivitas ini, Anda akan mempelajari cara memanfaatkan RouterLink arahan untuk memanfaatkan Angular Router secara maksimal

1. Impor RouterLink arahan

Tambahkan `app.component.ts` arahan RouterLink import ke pernyataan import yang ada dari `@angular/router` dan tambahkan ke `imports` array dekorator komponen Anda.



```
aplikasi/aplikasi.component.ts  +
1  import { Component } from '@angular/core';
2  import { RouterLink, RouterOutlet } from '@angular/router';
3
4  @Component({
5    standalone: true,
6    imports: [RouterLink, RouterOutlet],
7  })
8  export class AppComponent {}
```

Pratinjau Menghibur 6 Terminal

[HomeUser](#)
Home Page

2. Tambahkan routerLink ke templat

Untuk menggunakan RouterLink direktif, ganti `href` atribut dengan `routerLink`.
Perbarui templat dengan perubahan ini.


```
aplikasi/aplikasi.component.ts +
1 import { Component } from '@angular/core';
2 import { RouterLink, RouterOutlet } from '@angular/router';
3
4 @Component({
5   selector: 'app-root',
6   template: `
7     <nav>
8       <a routerLink="/">Home</a>
9       <a routerLink="/user">User</a>
10     </nav>
11     <router-outlet></router-outlet>
12   `,
13   standalone: true,
14   imports: [RouterLink, RouterOutlet],
15 })
16 export class AppComponent {}
17 |
```

Pratinjau Menghibur Terminal

[HomeUser](#)
Home Page

Formulir

Formulir adalah bagian penting dari banyak aplikasi karena memungkinkan aplikasi Anda menerima masukan pengguna. Mari pelajari cara penanganan formulir di Angular.

Di Angular, ada dua jenis formulir: berbasis template dan reaktif. Anda akan mempelajari keduanya dalam beberapa aktivitas berikutnya.

Dalam aktivitas ini, Anda akan mempelajari cara menyiapkan formulir menggunakan pendekatan berbasis templat.

1. Buat kolom masukan

Di `user.component.ts`, perbarui template dengan menambahkan input teks dengan `idset to framework`, ketik `set to text`.

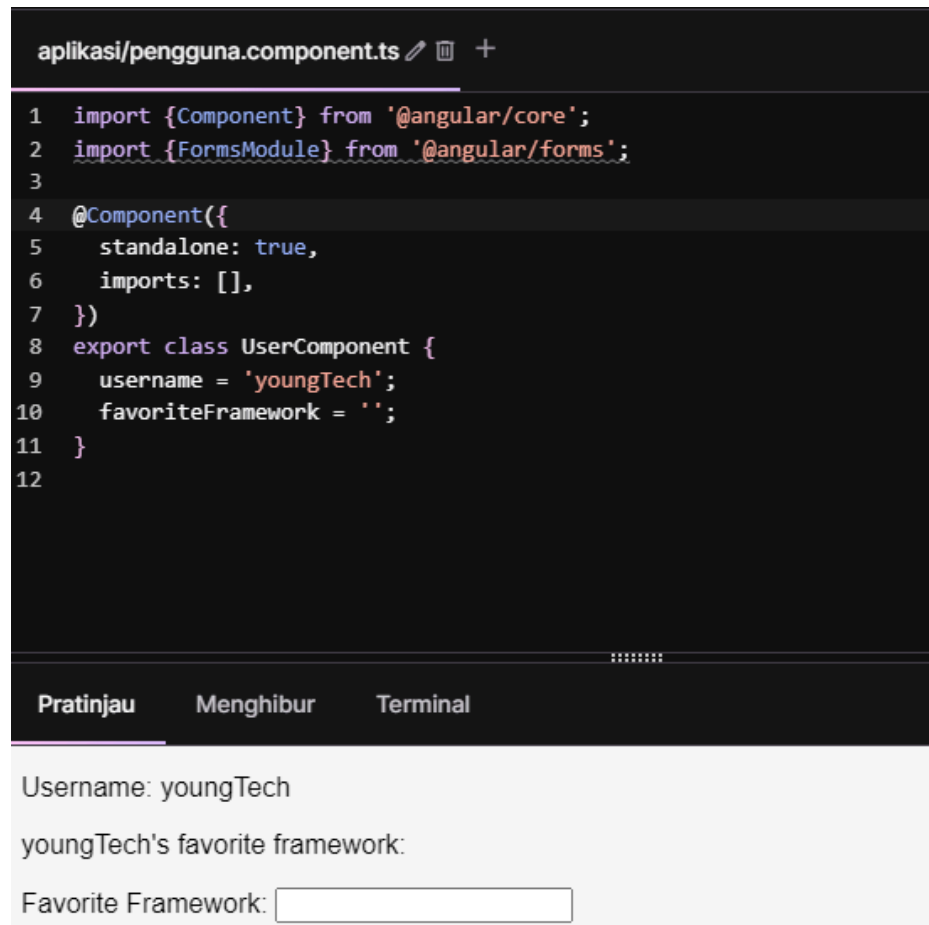
```
9 <label for="framework">
10   Favorite Framework:
11   <input id="framework" type="text" />
12 </label>
13 `
14   standalone: true,
15   imports: [],
16 })
17 export class UserComponent {
```

Pratinjau Menghibur Terminal

Username: youngTech
youngTech's favorite framework:
Favorite Framework:

2. ImportFormsModule

Agar formulir ini dapat menggunakan fitur Angular yang mengaktifkan pengikatan data ke formulir, Anda harus mengimpor file FormsModule.



```
aplikasi/pengguna.component.ts  ✎  🗑  +
1  import {Component} from '@angular/core';
2  import {FormsModule} from '@angular/forms';
3
4  @Component({
5    standalone: true,
6    imports: [],
7  })
8  export class UserComponent {
9    username = 'youngTech';
10    favoriteFramework = '';
11  }
12
```

Pratinjau Menghibur Terminal

Username: youngTech
youngTech's favorite framework:
Favorite Framework:

Impor `FormsModule` dari `@angular/forms` dan tambahkan ke `imports` array `UserComponent`.

3. Tambahkan pengikatan ke nilai input

Ini `FormsModule` memiliki arahan yang disebut `ngModel` yang mengikat nilai input ke properti di kelas Anda.

Perbarui masukan untuk menggunakan `ngModel` arahan, khususnya dengan sintaks berikut `[(ngModel)]="favoriteFramework"` untuk mengikat ke `favoriteFramework` properti.

```
9  <label for="framework">
10    Favorite Framework:
11    <input id="framework" type="text" [(ngModel)]="favoriteFramework" />
12  </label>
13  `,
14  standalone: true,
15  imports: [FormsModule],
16  })
17  export class UserComponent {
18    username = 'youngTech';
19    favoriteFramework = '';
20  }
```

Pratinjau Menghibur Terminal

Username: youngTech
youngTech's favorite framework:
Favorite Framework:

Setelah Anda melakukan perubahan, coba masukkan nilai di kolom input. Perhatikan bagaimana pembaruan di layar (ya, sangat keren).

Catatan: Sintaksnya `[()]` dikenal sebagai "pisang dalam kotak" namun mewakili pengikatan dua arah: pengikatan properti dan pengikatan peristiwa. Pelajari lebih lanjut di dokumen Angular tentang pengikatan data dua arah .

Mendapatkan nilai kontrol formulir

Sekarang formulir Anda sudah disiapkan dengan Angular, langkah selanjutnya adalah mengakses nilai dari kontrol formulir.

Dalam aktivitas ini, Anda akan mempelajari cara mendapatkan nilai dari input formulir Anda.

1. Tampilkan nilai kolom input di templat

Untuk menampilkan nilai masukan dalam templat, Anda dapat menggunakan sintaks interpolasi `{{ }}` sama seperti properti kelas komponen lainnya:

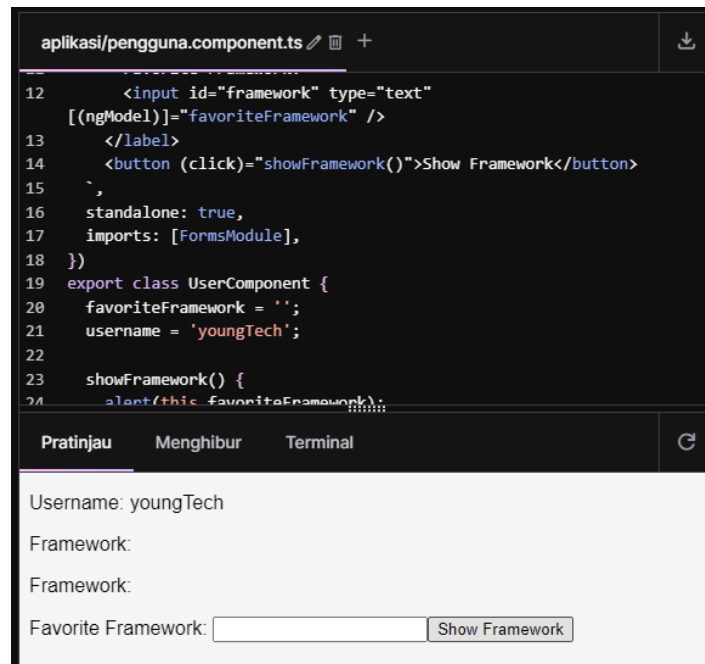
```
aplikasi/pengguna.component.ts  +
4  @Component({
5    selector: 'app-user',
6    template: `
7      <p>Username: {{ username }}</p>
8      <p>Framework: {{ favoriteFramework }}</p>
9      <p>Framework:</p>
10     <label for="framework">
11       Favorite Framework:
12       <input id="framework" type="text"
13       [(ngModel)]="favoriteFramework" />
14     </label>
15     <button (click)="showFramework()">Show Framework</button>
16   `
17 })
18 export class UserComponent {
19   username = 'youngTech';
20   favoriteFramework = '';
21   showFramework() {
22     this.favoriteFramework = this.favoriteFramework + ' ' + this.username;
23   }
24 }
```

Pratinjau Menghibur Terminal

Username: youngTech
Framework:
Framework:
Favorite Framework: Show Framework

2. Ambil nilai kolom input

Saat Anda perlu mereferensikan nilai kolom input di kelas komponen, Anda dapat melakukannya dengan mengakses properti kelas menggunakan sintaksis `this`.



The screenshot shows a code editor with the file `aplikasi/pengguna.component.ts`. The code defines a `UserComponent` class with a `showFramework()` method that uses `this.favoriteFramework` to access a property. Below the code, the 'Pratinjau' (Preview) tab shows the rendered UI: 'Username: youngTech', 'Framework:' labels, and a 'Favorite Framework' input field with a 'Show Framework' button.

```
12 <input id="framework" type="text"
    [(ngModel)]="favoriteFramework" />
13 </label>
14 <button (click)="showFramework()">Show Framework</button>
15 `
16 standalone: true,
17 imports: [FormsModule],
18 })
19 export class UserComponent {
20   favoriteFramework = '';
21   username = 'youngTech';
22
23   showFramework() {
24     alert(this.favoriteFramework);
25   }
26 }
```

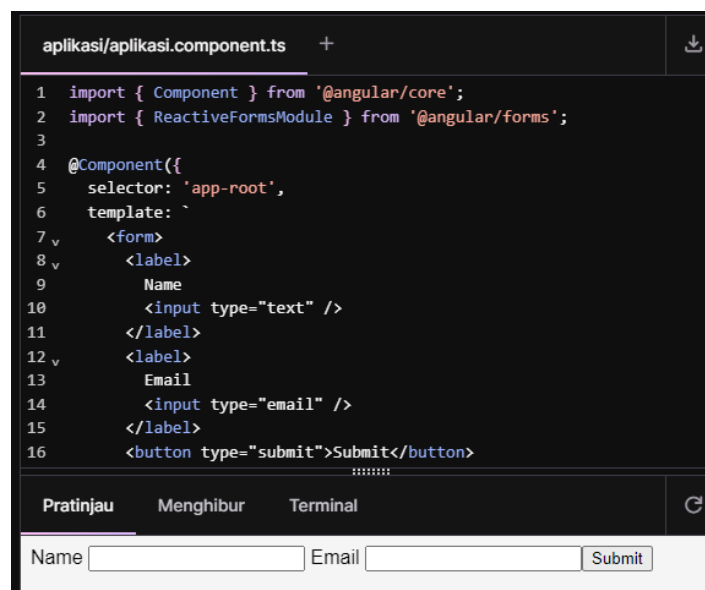
Bentuk Reaktif

Saat Anda ingin mengelola formulir secara terprogram dan bukan hanya mengandalkan template, formulir reaktif adalah jawabannya.

Dalam aktivitas ini, Anda akan mempelajari cara menyiapkan formulir reaktif.

1. Impor ReactiveFormsmodul

Di `app.component.ts`, impor `ReactiveFormsModule` dari `@angular/forms` dan tambahkan ke `imports` array komponen.



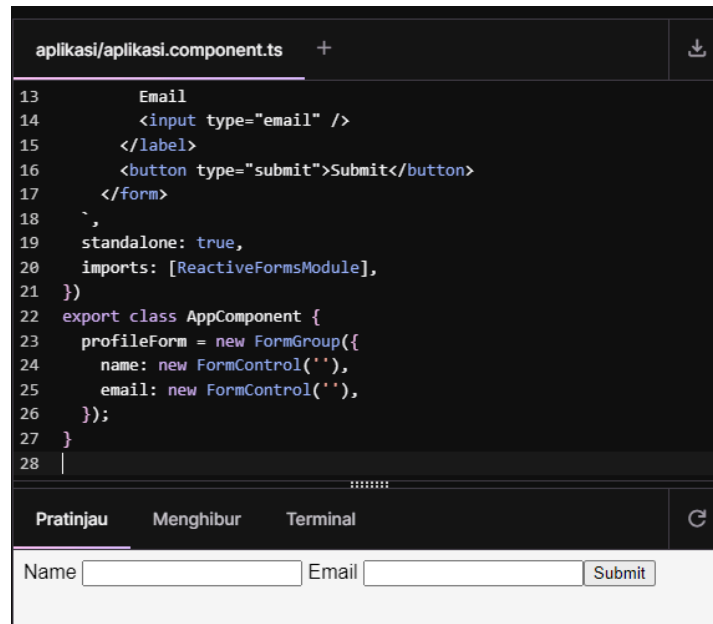
The screenshot shows the `aplikasi/aplikasi.component.ts` file. It imports `Component` from `@angular/core` and `ReactiveFormsModule` from `@angular/forms`. The `@Component` decorator includes a `template` with form fields for 'Name' and 'Email', and a 'Submit' button. The 'Pratinjau' (Preview) tab shows the rendered form with input fields for 'Name' and 'Email' and a 'Submit' button.

```
1 import { Component } from '@angular/core';
2 import { ReactiveFormsModule } from '@angular/forms';
3
4 @Component({
5   selector: 'app-root',
6   template: `
7     <form>
8       <label>
9         Name
10        <input type="text" />
11      </label>
12     <label>
13       Email
14       <input type="email" />
15     </label>
16     <button type="submit">Submit</button>
17   `
18 })
19 export class AppComponent {
20 }
21 
```

2. Buat FormGroup objek dengan FormControl

Formulir reaktif menggunakan FormControl kelas untuk mewakili kontrol formulir (misalnya input). Angular menyediakan FormGroup kelas yang berfungsi sebagai pengelompokan kontrol formulir menjadi objek bermanfaat yang membuat penanganan formulir besar lebih nyaman bagi pengembang.

Tambahkan FormControl dan FormGroup ke impor dari @angular/forms sehingga Anda bisa membuat FormGroup untuk setiap formulir, dengan properti bernama email sebagai FormControl.



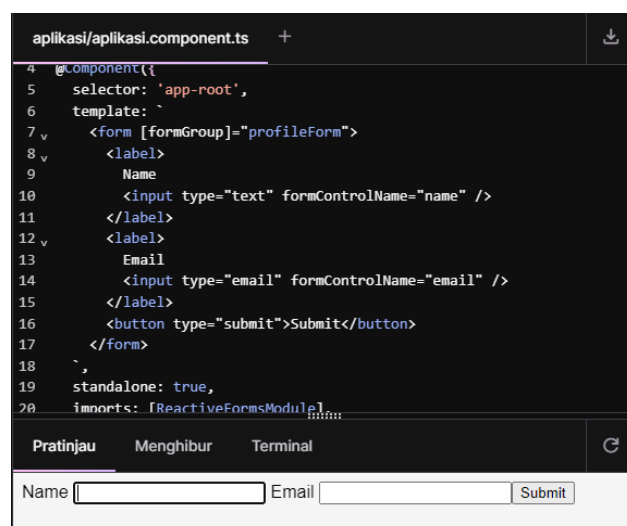
```
aplikasi/aplikasi.component.ts +
13     Email
14     <input type="email" />
15   </label>
16   <button type="submit">Submit</button>
17 </form>
18 `},
19 standalone: true,
20 imports: [ReactiveFormsModule],
21 })
22 export class AppComponent {
23   profileForm = new FormGroup({
24     name: new FormControl(''),
25     email: new FormControl(''),
26   });
27 }
28 |
```

Pratinjau Menghibur Terminal

Name Email

3. Tautkan FormGroup dan FormControl ke formulir

Masing-masing FormGroup harus dilampirkan ke formulir menggunakan [formGroup] arahan. Selain itu, masing-masing FormControl dapat dilampirkan dengan FormControlName arahan dan ditugaskan ke properti terkait. Perbarui templat dengan kode formulir berikut:



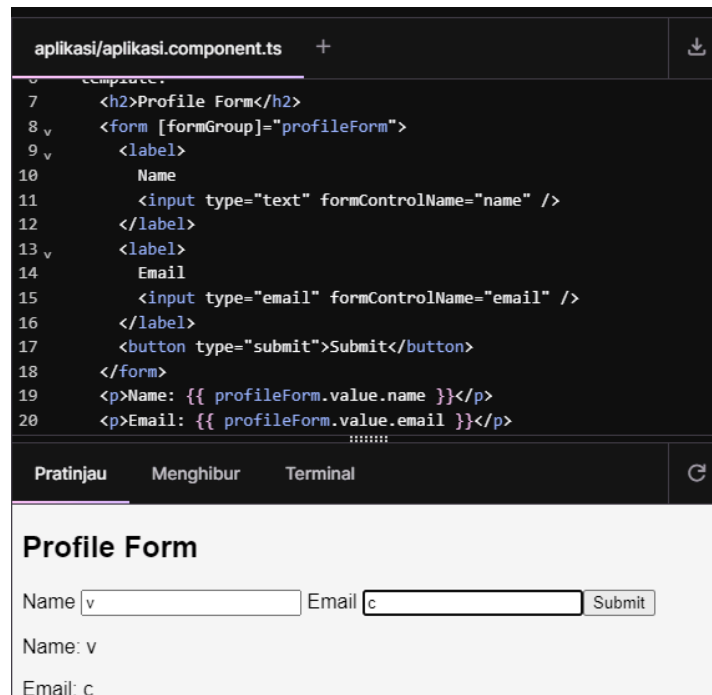
```
aplikasi/aplikasi.component.ts +
4 @Component({
5   selector: 'app-root',
6   template: `
7     <form [formGroup]="profileForm">
8       <label>
9         Name
10        <input type="text" FormControlName="name" />
11      </label>
12       <label>
13         Email
14        <input type="email" FormControlName="email" />
15      </label>
16      <button type="submit">Submit</button>
17    </form>
18  `},
19  standalone: true,
20  imports: [ReactiveFormsModule]
21 })
22 |
```

Pratinjau Menghibur Terminal

Name Email

4. Tangani pembaruan pada formulir

Bila ingin mengakses data dari FormGroup, dapat dilakukan dengan mengakses nilai dari FormGroup. Perbarui template untuk menampilkan nilai formulir:

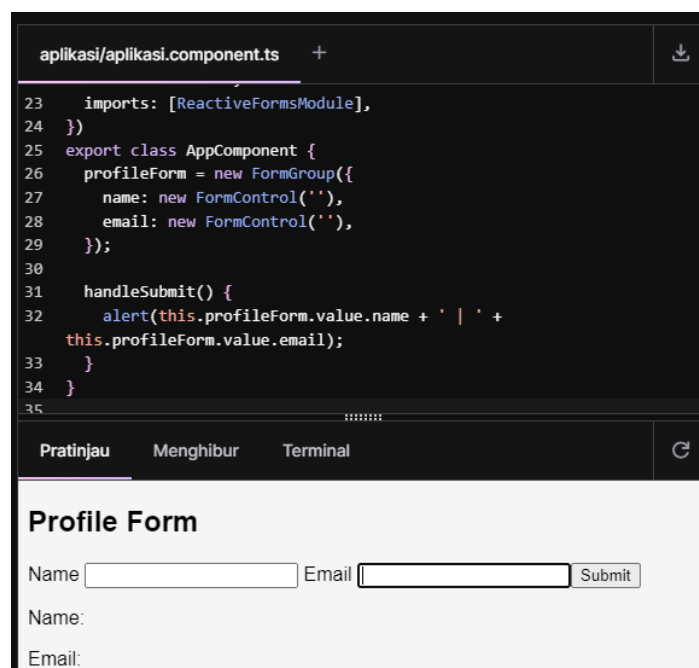


```
aplikasi/aplikasi.component.ts +
7 <h2>Profile Form</h2>
8 <form [formGroup]="profileForm">
9   <label>
10     Name
11     <input type="text" formControlName="name" />
12   </label>
13   <label>
14     Email
15     <input type="email" formControlName="email" />
16   </label>
17   <button type="submit">Submit</button>
18 </form>
19 <p>Name: {{ profileForm.value.name }}</p>
20 <p>Email: {{ profileForm.value.email }}</p>
.....
Pratinjau Menghibur Terminal
Profile Form
Name v Email c Submit
Name: v
Email: c
```

5. Akses nilai FormGroup

Tambahkan metode baru ke kelas komponen yang handleSubmitnantinya akan Anda gunakan untuk menangani pengiriman formulir. Metode ini akan menampilkan nilai dari form, Anda dapat mengakses nilai dari FormGroup.

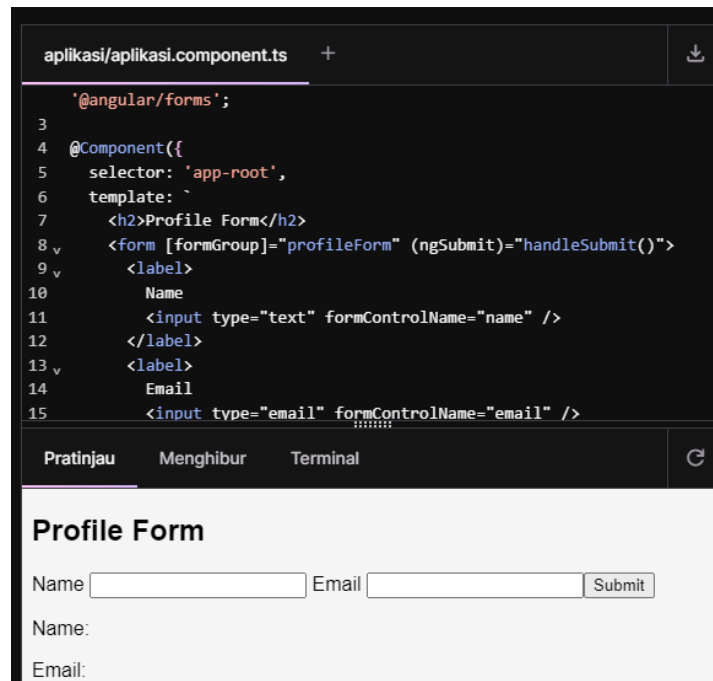
Di kelas komponen, tambahkan handleSubmit()metode untuk menangani pengiriman formulir.



```
aplikasi/aplikasi.component.ts +
23 imports: [ReactiveFormsModule],
24 })
25 export class AppComponent {
26   profileForm = new FormGroup({
27     name: new FormControl(''),
28     email: new FormControl(''),
29   });
30
31   handleSubmit() {
32     alert(this.profileForm.value.name + ' | ' +
33           this.profileForm.value.email);
34   }
35
36   .....
Pratinjau Menghibur Terminal
Profile Form
Name Email Submit
Name:
Email:
```

6. Tambahkan ngSubmit ke formulir

Anda memiliki akses ke nilai formulir, sekarang saatnya menangani acara pengiriman dan menggunakan `handleSubmit` metode ini. Angular memiliki event handler untuk tujuan khusus ini yang disebut `ngSubmit`. Perbarui elemen formulir untuk memanggil `handleSubmit` metode saat formulir dikirimkan.



Memvalidasi formulir

Skenario umum lainnya ketika bekerja dengan formulir adalah kebutuhan untuk memvalidasi masukan untuk memastikan data yang dikirimkan benar.

Dalam aktivitas ini, Anda akan mempelajari cara memvalidasi formulir dengan formulir reaktif.

1. Impor Validator

Angular menyediakan seperangkat alat validasi. Untuk menggunakannya, pertama-tama perbarui komponen yang akan diimpor Validators dari `@angular/forms`.

```
aplikasi/aplikasi.component.ts +
1 import {Component} from '@angular/core';
2 import {FormGroup, FormControl} from '@angular/forms';
3 import {ReactiveFormsModule, Validators} from '@angular/forms';
4
5 @Component({
6   selector: 'app-root',
7   template: `
8     <form [formGroup]="profileForm">
9       <input type="text" formControlName="name" name="name" />
10      <input type="email" formControlName="email" name="email" />
11      <button type="submit">Submit</button>
12    </form>
13  `,
14   standalone: true,
15   imports: [ReactiveFormsModule],
16 })
17 export class AppComponent {
18   profileForm = new FormGroup({
19     name: new FormControl(''),
20     email: new FormControl('')
21   });
22 }
23
Pratinjau Menghibur Terminal
Submit
```

2. Tambahkan validasi ke formulir

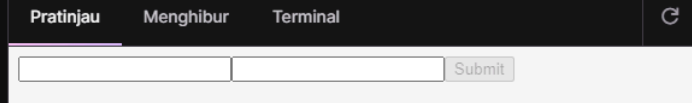
Setiap FormControl dapat diteruskan ke yang Validators ingin Anda gunakan untuk memvalidasi FormControl nilai. Misalnya, jika Anda ingin membuat name bidang tersebut profileForm wajib diisi, gunakan Validators.required. Untuk email bidang dalam formulir Angular, kami ingin memastikan bidang tersebut tidak dibiarkan kosong dan mengikuti struktur alamat email yang valid. Kita dapat mencapainya dengan menggabungkan validator Validators.required dan Validators.email dalam sebuah array. Perbarui name dan email FormControl:

```
aplikasi/aplikasi.component.ts +
9      <input type="text" formControlName="name" name="name" />
10     <input type="email" formControlName="email" name="email" />
11     <button type="submit">Submit</button>
12   </form>
13   `,
14   standalone: true,
15   imports: [ReactiveFormsModule],
16 })
17 export class AppComponent {
18   profileForm = new FormGroup({
19     name: new FormControl('', Validators.required),
20     email: new FormControl('', [Validators.required,
21                               Validators.email]),
22   });
23 }
24
Pratinjau Menghibur Terminal
Submit
```

3. Periksa validasi formulir di templat

Untuk menentukan apakah suatu formulir valid, FormGroup kelas memiliki valid property. Anda dapat menggunakan property ini untuk mengikat atribut secara dinamis. Perbarui pengiriman button agar diaktifkan berdasarkan validitas formulir.

```
3 import {ReactiveFormsModule, Validators} from '@angular/forms';
4
5 @Component({
6   selector: 'app-root',
7   template: `
8     <form [formGroup]="profileForm">
9       <input type="text" formControlName="name" name="name" />
10      <input type="email" formControlName="email" name="email" />
11      <button type="submit"
12        [disabled]="!profileForm.valid">Submit</button>
13    </form>
14  `,
15   standalone: true,
16   imports: [ReactiveFormsModule],
17 })
18 export class AppComponent {
19   .....
20 }
```



Membuat layanan injeksi

Injeksi ketergantungan (DI) di Angular adalah salah satu fitur kerangka kerja yang paling kuat. Pertimbangkan injeksi ketergantungan sebagai kemampuan Angular untuk menyediakan sumber daya yang Anda perlukan untuk aplikasi Anda saat runtime. Ketergantungan dapat berupa layanan atau sumber daya lainnya.

Anda dapat mempelajari lebih lanjut tentang injeksi ketergantungan di dokumen Angular . Untuk saat ini, Anda akan berlatih membuat injectablesumber daya.

Dalam aktivitas ini, Anda akan mempelajari cara membuat layanan injeksi.

Salah satu cara menggunakan layanan adalah bertindak sebagai cara untuk berinteraksi dengan data dan API. Untuk membuat layanan dapat digunakan kembali, Anda harus menyimpan logika dalam layanan dan membagikannya ke seluruh aplikasi saat diperlukan.

Untuk membuat layanan memenuhi syarat untuk disuntikkan oleh sistem DI gunakan @Injectabledekorator. Misalnya:

```
@Injectable({
  providedIn: 'root'
})
class UserService {
  // methods to retrieve and return data
}
```

Dekorator @Injectablememberi tahu sistem DI bahwa UserServicetersedia untuk diminta di kelas. providedInmenetapkan cakupan di mana sumber daya ini tersedia. Untuk saat ini,

cukup dipahami bahwa ini providedIn: 'root' berarti UserService tersedia untuk seluruh aplikasi.

1. Tambahkan @Injectable dekorator

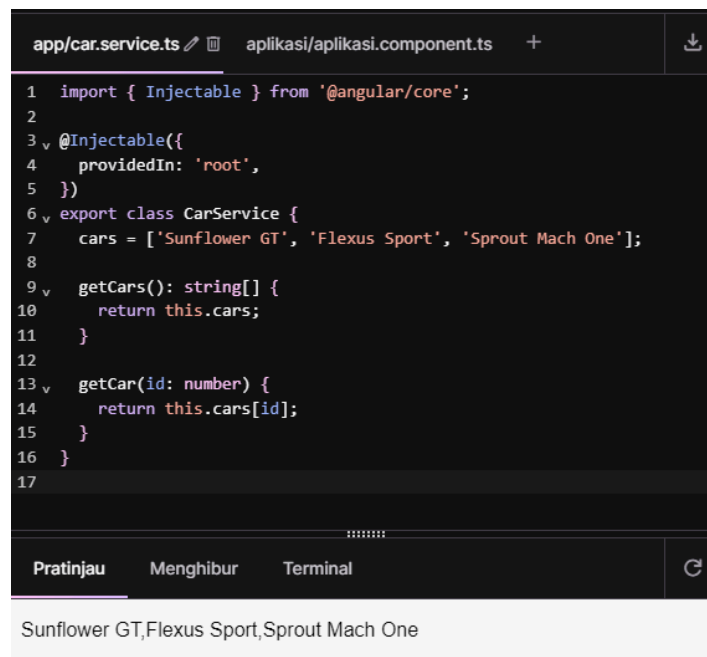
Perbarui kode car.service.ts dengan menambahkan @Injectable dekorator.

2. Konfigurasi dekorator

Nilai-nilai dalam objek yang diteruskan ke dekorator dianggap sebagai konfigurasi untuk dekorator.

Perbarui @Injectable dekorator untuk car.service.ts menyertakan konfigurasi untuk providedIn: 'root'.

Tip: Gunakan contoh di atas untuk menemukan sintaks yang benar.



```
app/car.service.ts  aplikasi/aplikasi.component.ts  +  ⬇
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root',
5  })
6  export class CarService {
7    cars = ['Sunflower GT', 'Flexus Sport', 'Sprout Mach One'];
8
9    getCars(): string[] {
10     return this.cars;
11   }
12
13   getCar(id: number) {
14     return this.cars[id];
15   }
16 }
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595

```

```

app/car.service.ts  aplikasi/aplikasi.component.ts  +
1 import {Component, inject} from '@angular/core';
2 import {CarService} from './car.service';
3
4 @Component({
5   selector: 'app-root',
6   template: ``,
7   standalone: true,
8 })
9 export class AppComponent {
10   display = '';
11   private carService = inject(CarService);
12
13
14   constructor() {}
15 }
16

```

2. Gunakan carServicecontohnya

Memanggil inject(CarService)memberi Anda contoh yang CarServicecepat Anda gunakan dalam aplikasi Anda, disimpan di carServiceproperti.

Pada constructorfungsi AppComponent, tambahkan implementasi berikut:

```

app/car.service.ts  aplikasi/aplikasi.component.ts  +
1 import {Component, inject} from '@angular/core';
2 import {CarService} from './car.service';
3
4 @Component({
5   selector: 'app-root',
6   template: ``,
7   standalone: true,
8 })
9 export class AppComponent {
10   display = '';
11   private carService = inject(CarService);
12
13
14   constructor() {
15     this.display = this.carService.getCars().join(' ★ ');
16   }
17 }
18

```

3. Perbarui AppComponenttemplatnya

Perbarui templat komponen app.component.tsdengan kode berikut:

```
1 import {Component, inject} from '@angular/core';
2 import {CarService} from './car.service';
3
4 @Component({
5   selector: 'app-root',
6   template: `<p>Car Listing: {{ display }}</p>`,
7   standalone: true,
8 })
9 export class AppComponent {
10   display = '';
11   private carService = inject(CarService);
12
13
14   constructor() {
15     this.display = this.carService.getCars().join(' ★ ');
16   }
17 }
18
```

Pratinjau Menghibur Terminal

Car Listing: Sunflower GT ★ Flexus Sport ★ Sprout Mach One

Injeksi ketergantungan berbasis konstruktor

Dalam aktivitas sebelumnya Anda menggunakan inject()fungsi untuk menyediakan sumber daya, "menyediakan" sumber daya tersebut ke komponen Anda. Fungsinya inject()adalah satu pola dan penting untuk mengetahui bahwa ada pola lain untuk menyuntikkan sumber daya yang disebut injeksi ketergantungan berbasis konstruktor.

Anda menentukan sumber daya sebagai parameter fungsi constructorkomponen. Angular akan menyediakan sumber daya tersebut untuk komponen Anda.

Dalam aktivitas ini, Anda akan mempelajari cara menggunakan injeksi ketergantungan berbasis konstruktor.

Untuk menyuntikkan layanan atau sumber daya lain yang dapat disuntikkan ke dalam komponen Anda, gunakan sintaks berikut:

```
@Component({...})
class PetCarDashboardComponent {
  constructor(private petCareService: PetCareService) {
    ...
  }
}
```

1. Perbarui kode untuk menggunakan DI berbasis konstruktor

Di app.component.ts, perbarui kode konstruktor agar sesuai dengan kode di bawah ini:

```
app/car.service.ts  aplikasi/aplikasi.component.ts  +  ⬇

1 import {Injectable} from '@angular/core';
2
3 @Injectable({
4   providedIn: 'root',
5 })
6 export class CarService {
7   cars = ['Sunflower GT', 'Flexus Sport', 'Sprout Mach One'];
8
9   getCars(): string[] {
10    return this.cars;
11  }
12
13   getCar(id: number) {
14    return this.cars[id];
15  }
16 }
17
18
```

Pratinjau Menghibur Terminal ↻

Car Listing: Sunflower GT ★ Flexus Sport ★ Sprout Mach One

```
app/car.service.ts  aplikasi/aplikasi.component.ts  +  ⬇

1 import {Component, inject} from '@angular/core';
2 import {CarService} from './car.service';
3
4 @Component({
5   selector: 'app-root',
6   template: `
7     <p>Car Listing: {{ display }}</p>
8   `,
9   standalone: true,
10 })
11 export class AppComponent {
12   display = '';
13
14   constructor(private carService: CarService) {
15     this.display = this.carService.getCars().join(' ★ ');
16   }
17 }
18
```

Pratinjau Menghibur Terminal ↻

Car Listing: Sunflower GT ★ Flexus Sport ★ Sprout Mach One

#Pipa

Pipa adalah fungsi yang digunakan untuk mengubah data dalam templat. Secara umum, pipa merupakan fungsi “murni” yang tidak menimbulkan efek samping. Angular memiliki sejumlah pipa bawaan berguna yang dapat Anda impor dan gunakan di komponen Anda. Anda juga dapat membuat pipa khusus.

*Impor LowerCasePipanya

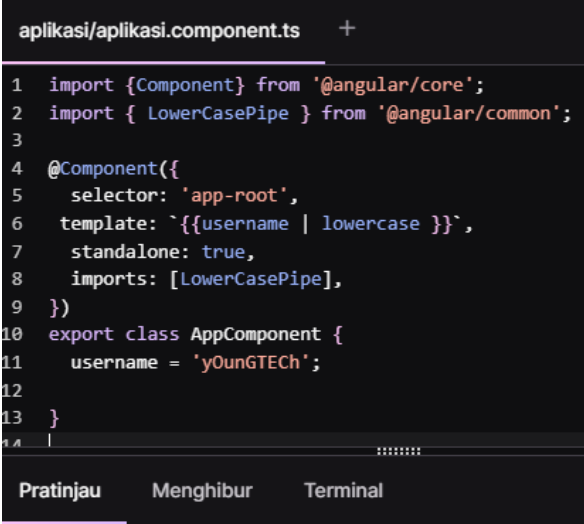
Pertama, perbarui app.component.ts dengan menambahkan impor tingkat file LowerCasePipedari @angular/common.

*Tambahkan pipa ke impor templat

Selanjutnya, perbarui @Component()dekorator importsuntuk menyertakan referensiLowerCasePipe

*Tambahkan pipa ke templat

Terakhir, app.component.tsperbarui templat untuk menyertakan lowercasepipa:



```
1 import {Component} from '@angular/core';
2 import { LowerCasePipe } from '@angular/common';
3
4 @Component({
5   selector: 'app-root',
6   template: `{{username | lowercase }}`,
7   standalone: true,
8   imports: [LowerCasePipe],
9 })
10 export class AppComponent {
11   username = 'yOunGTECh';
12 }
13
14 |
```

Pratinjau Menghibur Terminal

youngtech

#Memformat data dengan pipa

Anda dapat memanfaatkan pipa lebih jauh lagi dengan mengonfigurasinya. Pipa dapat dikonfigurasi dengan memberikan opsi kepada pipa tersebut.

Dalam aktivitas ini, Anda akan mengerjakan beberapa pipa dan parameter pipa.

Format nomor denganDecimalPipe

Di app.component.ts, perbarui templat untuk menyertakan parameter pipa decimal.

template: `

Number with "decimal" {{ num | number:"3.2-2" }}

Catatan: Apa formatnya? Parameter untuk DecimalPipedisebut digitsInfo, parameter ini menggunakan

format: {minIntegerDigits}. {minFractionDigits} - {maxFractionDigits}

Format tanggal denganDatePipe

Sekarang, perbarui templat untuk menggunakan datepipa.

template: `

Date with "date" {{ birthday | date: 'medium' }}

Untuk kesenangan ekstra, cobalah beberapa parameter berbeda untuk date. Informasi lebih lanjut dapat ditemukan di dokumen Angular .

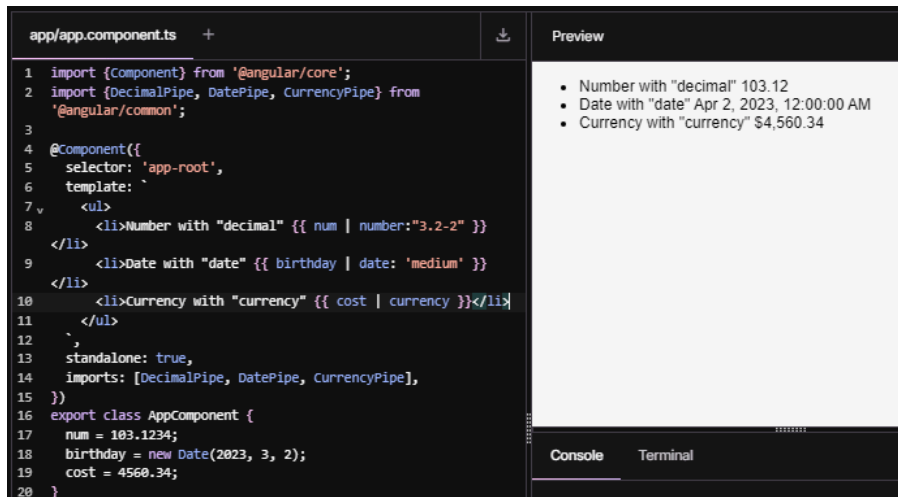
Format mata uang denganCurrencyPipe

Untuk tugas terakhir Anda, perbarui templat untuk menggunakan currencypipe.

template: `

```
<li>Currency with "currency" {{ cost | currency }}</li>
```

Anda juga dapat mencoba parameter berbeda untuk currency. Informasi lebih lanjut dapat ditemukan di dokumen Angular .



```
app/app.component.ts +
1 import {Component} from '@angular/core';
2 import {DecimalPipe, DatePipe, CurrencyPipe} from
  '@angular/common';
3
4 @Component({
5   selector: 'app-root',
6   template: `
7     <ul>
8       <li>Number with "decimal" {{ num | number:"3.2-2" }}
9     </li>
10      <li>Date with "date" {{ birthday | date: 'medium' }}
11    </li>
12      <li>Currency with "currency" {{ cost | currency }}</li>
13    </ul>
14  `
15  ,
16  standalone: true,
17  imports: [DecimalPipe, DatePipe, CurrencyPipe],
18 })
19 export class AppComponent {
20   num = 103.1234;
21   birthday = new Date(2023, 3, 2);
22   cost = 4560.34;
23 }
```

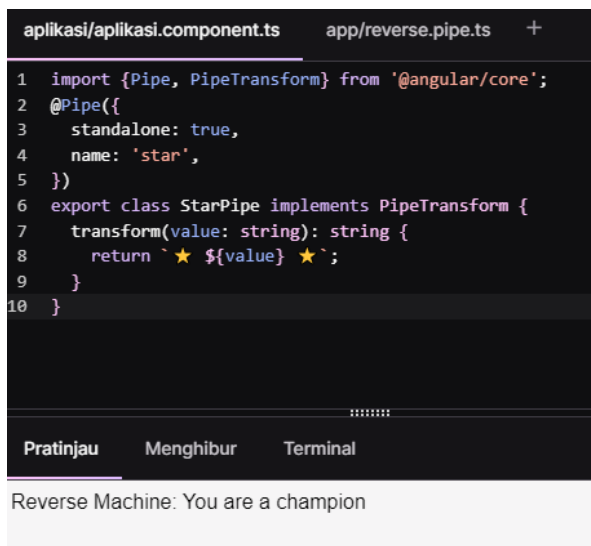
Preview

- Number with "decimal" 103.12
- Date with "date" Apr 2, 2023, 12:00:00 AM
- Currency with "currency" \$4,560.34

Console Terminal

#Create Pipe

Anda dapat membuat pipa khusus di Angular agar sesuai dengan kebutuhan transformasi data Anda. Dalam aktivitas ini, Anda akan membuat pipa kustom dan menggunakannya di template Anda.



```
aplikasi/aplikasi.component.ts app/reverse.pipe.ts +
1 import {Pipe, PipeTransform} from '@angular/core';
2 @Pipe({
3   standalone: true,
4   name: 'star',
5 })
6 export class StarPipe implements PipeTransform {
7   transform(value: string): string {
8     return `★ ${value} ★`;
9   }
10 }

Pratinjau Menghibur Terminal
Reverse Machine: You are a champion
```

*BuatReversePipe

Tambahkan reverse.pipe.tsdekorator @Pipeke ReversePipekelas dan berikan konfigurasi berikut:

```
@Pipe({
```

```

    standalone: true,
    name: 'reverse'
  })

```

*Terapkan transformfungsinya

Sekarang ReversePipekelasnya adalah sebuah pipa. Perbarui transformfungsi untuk menambahkan logika pembalikan:

```

export class ReversePipe implements PipeTransform {
  transform(value: string): string {
    let reverse = '';
    for (let i = value.length - 1; i >= 0; i--) {
      reverse += value[i];
    }
    return reverse;
  }
}

```

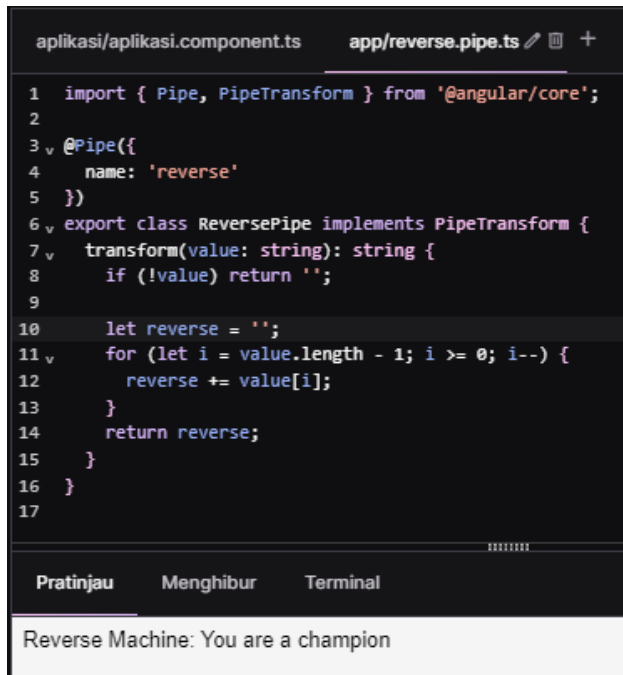
*Gunakan ReversePipedi templat

Dengan menerapkan logika pipa, langkah terakhir adalah menggunakannya dalam templat. Masukkan app.component.ts pipa ke dalam templat dan tambahkan ke impor komponen:

```

@Component({
  ...
  template: `Reverse Machine: {{ word | reverse }}`
  imports: [ReversePipe]
})

```

```
aplikasi/aplikasi.component.ts  app/reverse.pipe.ts  +
1  import { Pipe, PipeTransform } from '@angular/core';
2
3  @Pipe({
4    name: 'reverse'
5  })
6  export class ReversePipe implements PipeTransform {
7    transform(value: string): string {
8      if (!value) return '';
9
10     let reverse = '';
11     for (let i = value.length - 1; i >= 0; i--) {
12       reverse += value[i];
13     }
14     return reverse;
15   }
16 }
17
```

Pratinjau Menghibur Terminal

Reverse Machine: You are a champion

E. TUGAS

-

F. KESIMPULAN

Praktikum ini memberikan gambaran tentang bagaimana AngularJS digunakan dalam pengembangan aplikasi web, dengan fokus pada pengikatan data yang dinamis, penggunaan direktif untuk mengatur tampilan, serta kemampuan untuk mengurutkan data secara interaktif. Contoh yang diberikan dalam praktikum mengenalkan konsep dasar AngularJS seperti pembuatan modul, kontroler, penggunaan ekspresi dan filter, serta pengulangan data menggunakan ng-repeat. Keseluruhan materi praktikum ini bertujuan untuk memberikan pemahaman yang mendalam kepada mahasiswa dalam menerapkan AngularJS untuk mengembangkan aplikasi web yang modern.