

# **Projeto e Desenvolvimento de Software II**

**Prof. Carlos Eduardo de Carvalho Dantas**

**carloseduardodantas@iftm.edu.br**

# **Parte I – Técnicas e Ferramentas para Desenvolvimento de Sistemas**

# AGENDA

---

## 1. Desenvolvimento Front-End

- Exemplos Angular

# Atividades e Exercícios Angular

---

- 1) Criar um novo projeto Angular chamado ExerciciosAngular

```
PS C:\Users\carlo> ng new ExerciciosAngular  
As a forewarning, we are moving the CLI npm p
```

# Atividades e Exercícios Angular

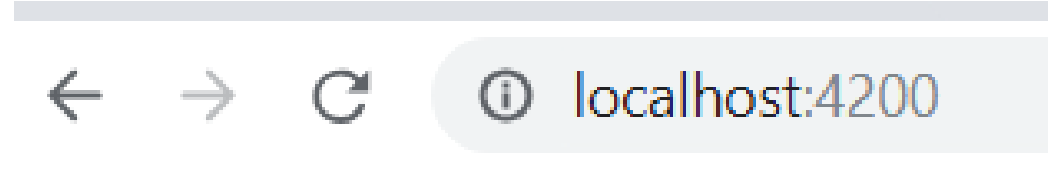
- Exemplo 1) A variável **nomes** se trata de um atributo da classe AppComponent, que é um componente. Essa variável pode ser usada no template html para que os dados sejam mostrados

```
TS app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css']
7  })
8  export class AppComponent {
9
10     nomes: string[] = ['joão', 'maria', 'josé', 'pedro', 'felipe', 'carlos'];
11
12  }
13
```

```
TS app.component.ts  <> app.component.html x
1  <ul>
2    <li *ngFor="let nome of nomes">
3      {{nome}}
4    </li>
5  </ul>
```

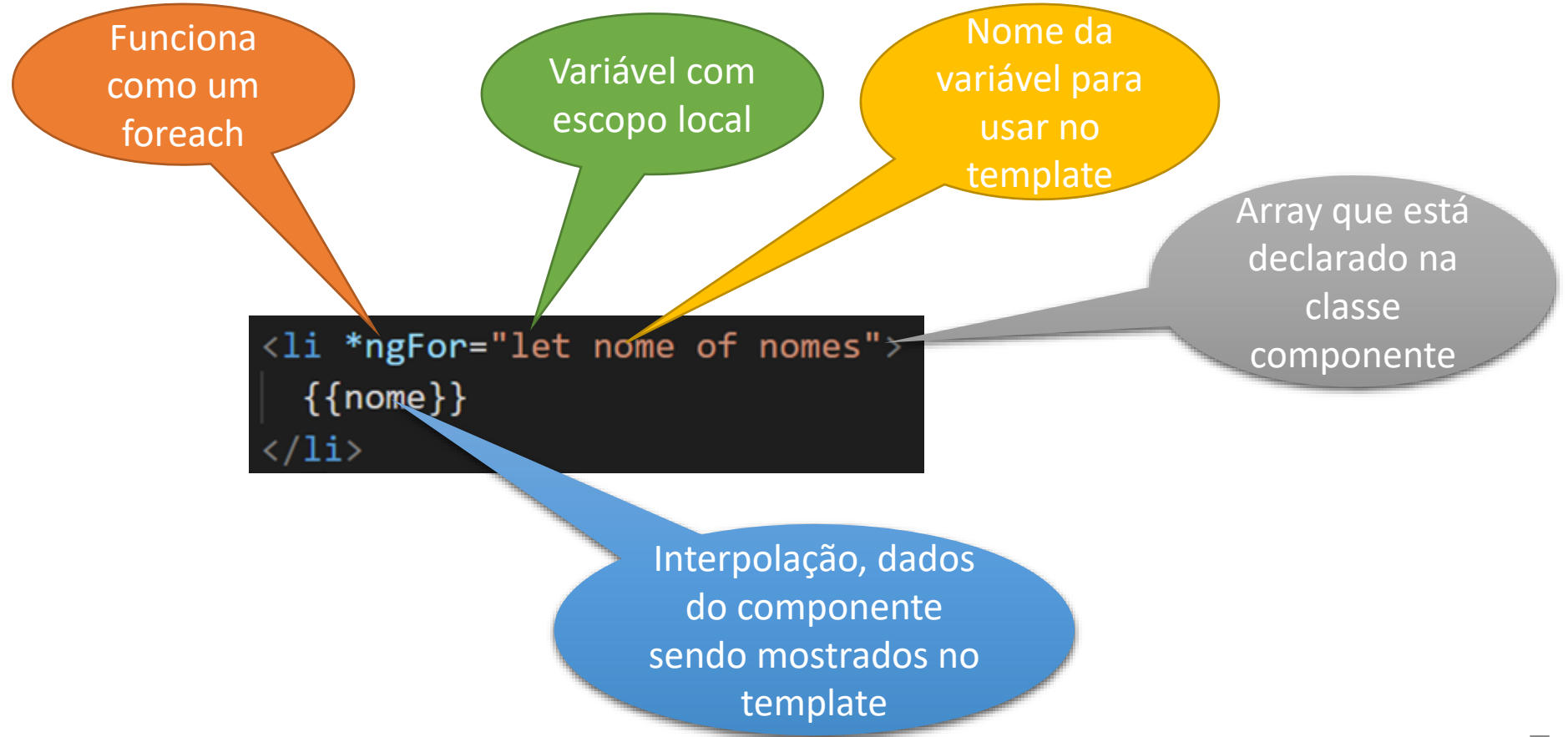
# Atividades e Exercícios Angular

## - Exemplo 1



- joão
- maria
- josé
- pedro
- felipe
- carlos

# Atividades e Exercícios Angular



# Atividades e Exercícios Angular

- Exemplo 2) É possível mostrar também a posição de cada elemento

```
ts app.component.ts  <> app.component.html ●  
1  <ul *ngFor="let nome of nomes, let i= index">  
2    <li>nome da pessoa é: {{nome}}, está na posição {{i + 1}} </li>  
3  </ul>
```

← → ↻ ⓘ localhost:4200

- nome da pessoa é: joão, está na posição 1
- nome da pessoa é: maria, está na posição 2
- nome da pessoa é: josé, está na posição 3
- nome da pessoa é: pedro, está na posição 4
- nome da pessoa é: felipe, está na posição 5
- nome da pessoa é: carlos, está na posição 6



# Atividades e Exercícios Angular

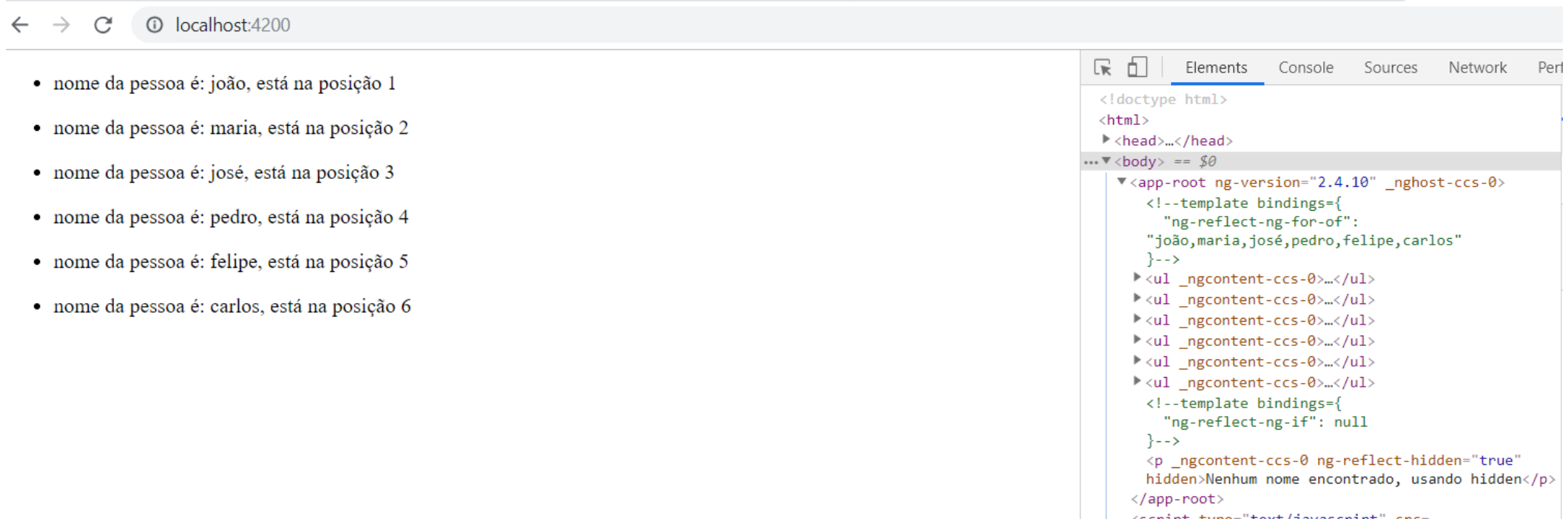
- Exemplo 3) Usando `*ngIf` e `[hidden]`. Ambos são usados para esconder determinado elemento (poderia ser uma div inteira) de acordo com uma condição booleana.

```
TS app.component.ts  <> app.component.html x
```

```
1  <ul *ngFor="let nome of nomes, let i= index">
2  |   <li>nome da pessoa é: {{nome}}, está na posição {{i + 1}} </li>
3  | </ul>
4  | <p *ngIf="nomes.length == 0">Nenhum nome encontrado, usando *ngIf</p>
5  |
6  | <p [hidden]="nomes.length > 0">Nenhum nome encontrado, usando hidden</p>
```

# Atividades e Exercícios Angular

- Exemplo 3) \*ngIf adiciona ou retira do DOM, o que é melhor para segurança, mas pior para desempenho.



The screenshot shows a web browser at localhost:4200 displaying a list of names and their positions. The list contains six items:

- nome da pessoa é: joão, está na posição 1
- nome da pessoa é: maria, está na posição 2
- nome da pessoa é: josé, está na posição 3
- nome da pessoa é: pedro, está na posição 4
- nome da pessoa é: felipe, está na posição 5
- nome da pessoa é: carlos, está na posição 6

The developer console shows the Angular template bindings for the list. The bindings are:

```
<!--template bindings={
  "ng-reflect-ng-for-of":
    "joão,maria,josé,pedro,felipe,carlos"
}-->
<ul _ngcontent-ccs-0>...</ul>
<ul _ngcontent-ccs-0>...</ul>
<ul _ngcontent-ccs-0>...</ul>
<ul _ngcontent-ccs-0>...</ul>
<ul _ngcontent-ccs-0>...</ul>
<ul _ngcontent-ccs-0>...</ul>
<!--template bindings={
  "ng-reflect-ng-if": null
}-->
<p _ngcontent-ccs-0 ng-reflect-hidden="true"
  hidden>Nenhum nome encontrado, usando hidden</p>
</app-root>
```

# Atividades e Exercícios Angular

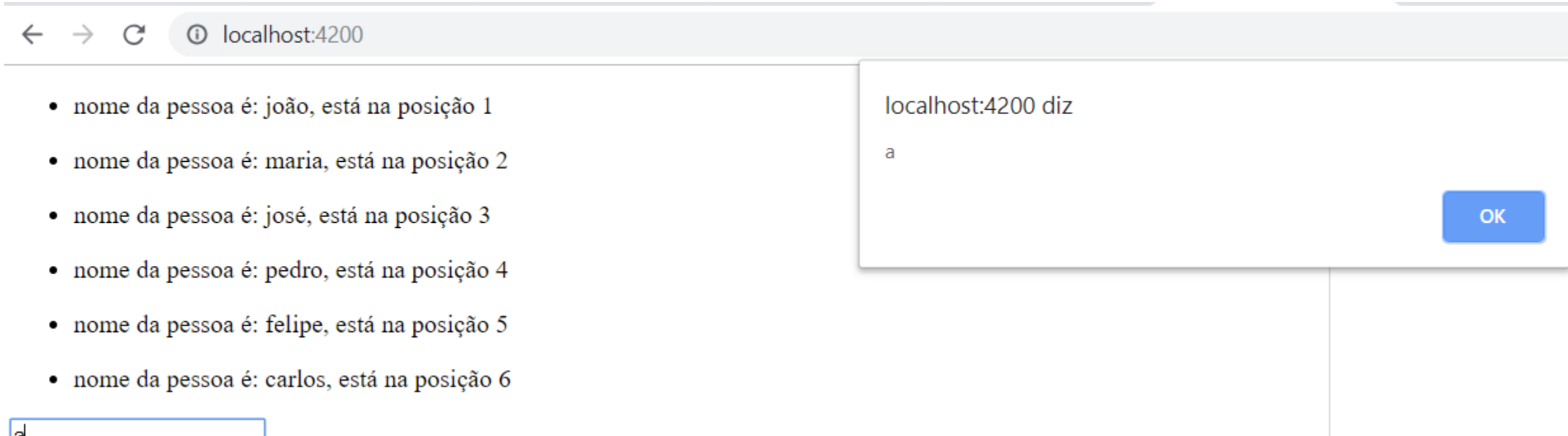
- Exemplo 4) Buscar por algum nome em específico

```
<input #box placeholder="digite um nome" (keyup)="buscar(box.value)"/>
```

```
export class AppComponent {  
  nomes: string[] = ['joão', 'maria', 'josé', 'pedro', 'felipe', 'carlos'];  
  
  buscar(valor: string) {  
    alert(valor)  
  }  
}
```

# Atividades e Exercícios Angular

## - Exemplo 4) Buscar por algum nome em específico



The screenshot shows a web browser window with the address bar displaying 'localhost:4200'. The main content area contains a list of six items, each representing a person and their position. Below the list is an input field. A modal dialog box is open on the right side of the browser window, displaying the text 'localhost:4200 diz' followed by a line break and the letter 'a'. An 'OK' button is located in the bottom right corner of the dialog box.

- nome da pessoa é: joão, está na posição 1
- nome da pessoa é: maria, está na posição 2
- nome da pessoa é: josé, está na posição 3
- nome da pessoa é: pedro, está na posição 4
- nome da pessoa é: felipe, está na posição 5
- nome da pessoa é: carlos, está na posição 6

localhost:4200 diz  
a

OK

# Atividades e Exercícios Angular

- Exemplo 4) Buscar por algum nome em específico

Variável de  
referência do  
template

Evento  
keyup, tecla  
pressionada

```
<input #box placeholder="digite um nome" (keyup)="buscar(box.value)"/>
```

Chamando o método na classe  
componente, usando a variável  
de referência para obter o valor  
do input

# Atividades e Exercícios Angular

- Exemplo 5) Implementando a busca
  - **Método 1** – iteração básica e verbosa.

```
<ul *ngFor="let nomeFiltro of nomesFiltro">  
  <li>{{nomeFiltro}}</li>  
</ul>
```

```
nomesFiltro: string[];  
  
buscar(valor: string) {  
  this.nomesFiltro = [];  
  
  //método 1  
  for (var i = 0; i < this.nomes.length; i++) {  
    if (this.nomes[i].toLowerCase().includes(valor.toLowerCase())) {  
      this.nomesFiltro.push(this.nomes[i]);  
    }  
  }  
}
```

Função para  
verificar se uma  
string é substring  
de outra

Passar a  
string para  
minúsculo

Adiciona elemento  
no final do array

# Atividades e Exercícios Angular

- Exemplo 5) Implementando a busca
  - **Método 2** – usando forEach, método interno do array.

```
//método 2
let temp = [];
this.nomes.forEach(buscarItem);
function buscarItem(nome) {
  if (nome.toLowerCase().includes(valor.toLowerCase())) {
    temp.push(nome);
  }
}
this.nomesFiltro = temp;
```

A própria função forEach já irá injetar cada nome como argumento para a função buscarItem

O argumento da função forEach é outra função que irá processar cada item do array

# Atividades e Exercícios Angular

- Exemplo 5) Implementando a busca
  - **Método 2** – a função que é chamada para processar cada elemento da lista também é chamada de função de callback.

```
var nomes = ['maria', 'josé', 'joão'];
```

```
nomes.forEach(function(nome){  
    console.log(nome);  
});
```



1º Iteração: nome = 'maria'  
2º Iteração: nome = 'josé'  
3º Iteração: nome = 'joão'



# Atividades e Exercícios Angular

- Exemplo 5) Implementando a busca
  - **Método 3** – usando o método filter.

```
//método 3  
this.nomesFiltro = this.nomes.filter(function (nome) {  
    return nome.toLowerCase().includes(valor.toLowerCase());  
});
```

O método filter é usado quando precisa filtrar os elementos da lista de acordo com algum critério

O argumento da função filter é outra função que irá processar cada item do array. No método 2 a função foi declarada explicitamente. No método 3, se trata de uma função anônima

# Atividades e Exercícios Angular

- Exemplo 5) Implementando a busca
  - **Método 4** – usando o método filter com arrow functions.

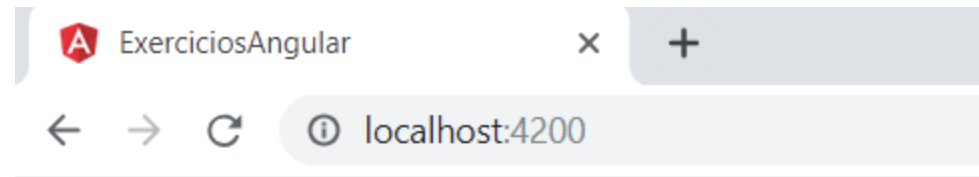
```
//método 4  
this.nomesFiltro = this.nomes.filter(  
  (nome) => nome.toLowerCase().includes(valor.toLowerCase()));
```

Possui exatamente a mesma semântica do método 3. Só diminui a verbosidade do código.

Já se sabe que o método filter irá receber cada nome da lista, e retornar booleano. Logo, a arrow function dispensa a formalidade das declarações

# Atividades e Exercícios Angular

## - Exemplo 5) Implementando a busca



- nome da pessoa é: joão, está na posição 1
- nome da pessoa é: maria, está na posição 2
- nome da pessoa é: josé, está na posição 3
- nome da pessoa é: pedro, está na posição 4
- nome da pessoa é: felipe, está na posição 5
- nome da pessoa é: carlos, está na posição 6

- carlos

# Atividades e Exercícios Angular

---

- É muito comum usar laços de repetição nos algoritmos. O problema é que com a abordagem dos métodos 1 e 2, é impossível saber qual o objetivo do corpo da iteração sem ver a sua implementação.
  - Buscar elemento?
  - listar todos?
  - Excluir?
  - Ordenar?
- Os métodos auxiliares visam explicitar qual operação se deseja realizar com base na sua sintaxe.

# Atividades e Exercícios Angular

---

- **Filter** – filtrar a lista de acordo com algum critério
- **Map** – modificar os elementos do array.
- **Find** – encontrar algum item específico
- **Every** – retorna verdadeiro se todos os itens do array respeitam alguma condição
- **Some** – retorna verdadeiro se pelo menos 1 item do array respeita alguma condição
- **Reduce** – Pegar todos os valores do array e condensar em um único elemento.

# Atividades e Exercícios Angular

## - Exemplo 6) Somando valores com reduce

Cria uma lista de "struct". Não é bem um objeto typescript, são apenas dados

```
14 | pessoas: any = [  
15 |   {id: 1, nome: 'joao',salario: 5000},  
16 |   {id: 2, nome: 'maria',salario: 1000},  
17 |   {id: 3, nome: 'jose',salario: 2000},  
18 |   {id: 4, nome: 'pedro',salario: 3000},  
19 |   {id: 5, nome: 'felipe',salario: 10000},  
20 |   {id: 6, nome: 'carlos',salario: 800},  
21 | ]  
22 |  
23 | getValorTotal(): Number {  
24 |   return this.pessoas.reduce(  
25 |     (soma,pessoa) => soma + pessoa.salario,0);  
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 |  

```

# Atividades e Exercícios Angular

- Exemplo 7) Buscando um elemento com find

```
buscarId(id) {  
  return this.pessoas.find(pessoa => pessoa.id == id);  
}
```

Cada elemento da lista é uma pessoa

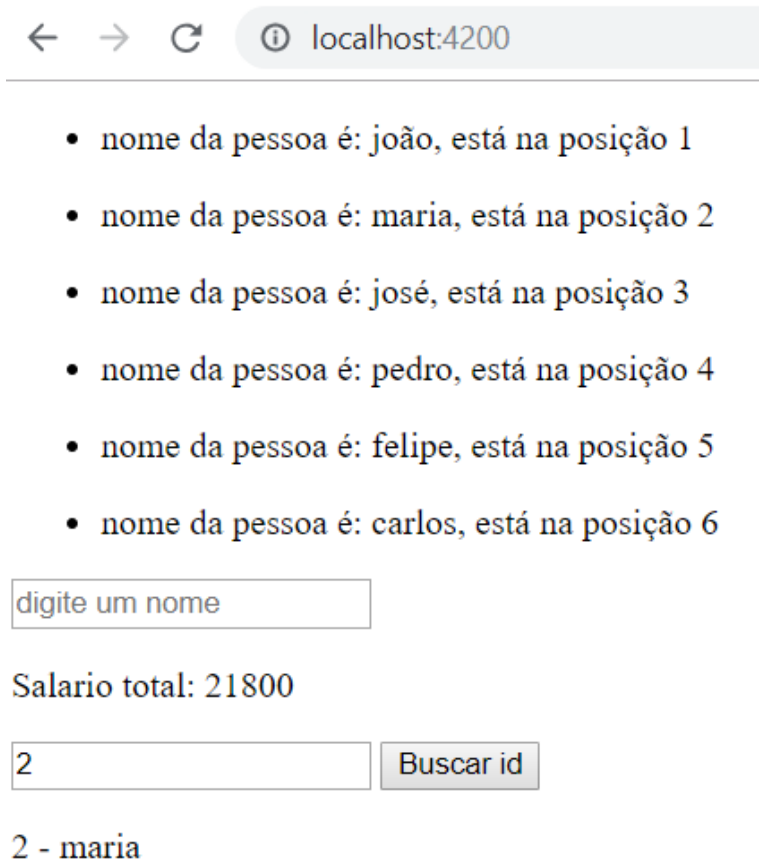
Só pode retornar uma pessoa, já que id é chave.

Usa variáveis de referência para o input e para o retorno da função

```
16 <input #parametroid placeholder="digite um id"/>  
17 <button (click)="retorno = buscarId(parametroid.value)">Buscar id</button>  
18 <p *ngIf="retorno != undefined">{{retorno.id}} - {{retorno.nome}}</p>
```

# Atividades e Exercícios Angular

## - Exemplo 7) Buscando um elemento com find



← → ↻ ⓘ localhost:4200

- nome da pessoa é: joão, está na posição 1
- nome da pessoa é: maria, está na posição 2
- nome da pessoa é: josé, está na posição 3
- nome da pessoa é: pedro, está na posição 4
- nome da pessoa é: felipe, está na posição 5
- nome da pessoa é: carlos, está na posição 6

digite um nome

Salario total: 21800

2

2 - maria



# Atividades e Exercícios Angular

## Exemplo 8) Aplicando map

Não existe  
retorno  
em map  
pois a  
alteração  
será  
aplicada  
na própria  
lista

```
28     aumentarSalario(percentual) {  
29         this.pessoas.map(pessoa =>  
30             pessoa.salario += pessoa.salario * percentual/100)  
31     }
```

```
20     <br>  
21     <input #percentual placeholder="digite o percentual"/>  
22     <button (click)="aumentarSalario(percentual.value)">Aumentar salario</button>  
23
```

# Atividades e Exercícios Angular

## - Exemplo 9) Aplicando every

```
verificaSalario(valor: number) {  
  return this.pessoas.every(pessoa => pessoa.salario > valor);  
}
```

Retorna booleano. Verifica se todo mundo ganha mais que o valor informado.

```
<p>{{verificaSalario(500)?'Todo mundo ganha mais que 500':  
'nem todo mundo ganha mais que 500'}}</p>
```

Usando operador ternário. Não se pode imprimir true ou false para o usuário

# Atividades e Exercícios Angular

## - Exemplo 10) Aplicando some

```
23 buscaCampos(criterio: string) {  
24     return this.pessoas.filter((pessoa) =>  
25         Object.keys(pessoa).some  
26         (chave => pessoa[chave].toString().includes(criterio)));  
}
```

Usa a combinação filter + some. Filter para filtrar por pessoas, e some para verificar os campos de cada pessoa

Retorna todas as chaves de pessoa: [id, nome, salario]

Verifica se o valor de algum dos campos de pessoa possui o critério como substring

# Atividades e Exercícios Angular

## - Exemplo 10) Aplicando some

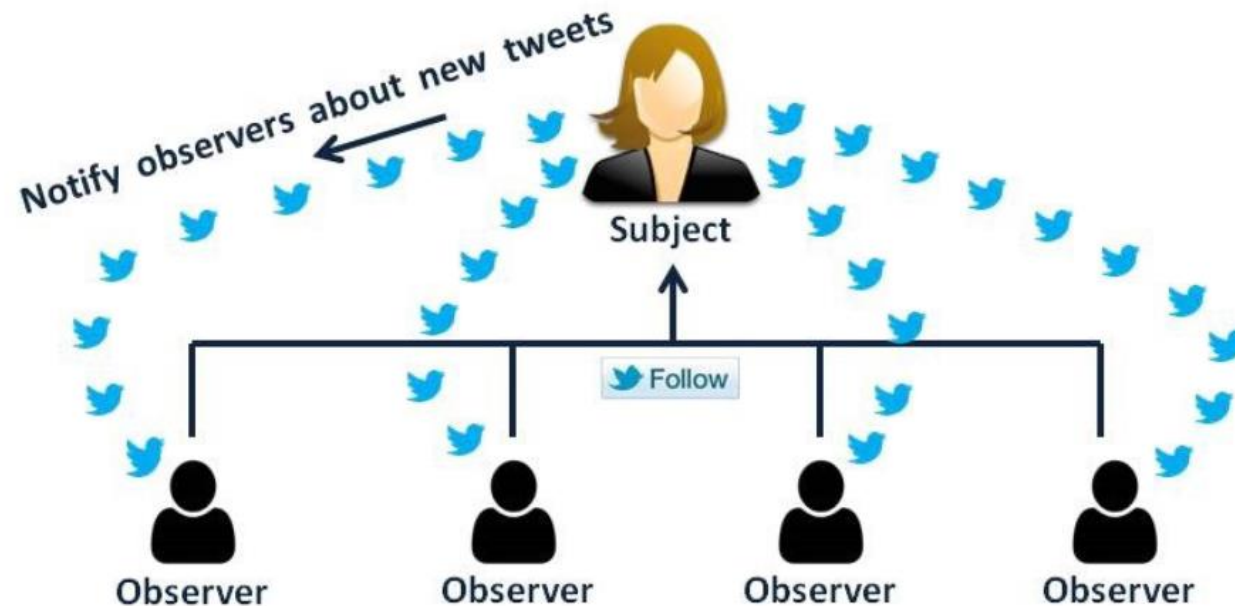
```
24 <br>
25 <input #criterio placeholder="digite um criterio de busca"
26 (keyup)="resposta = buscaCampos(criterio.value)"/>
27
28 <ul *ngFor="let r of resposta">
29   <li>{{r.id}} - {{r.nome}} - {{r.salario}}</li>
30 </ul>
```

- 1 - joao - 5000
- 2 - maria - 1000
- 5 - felipe - 10000

# Atividades e Exercícios Angular

- Observables - é uma coleção que funciona de forma unidirecional, ou seja, ele emite notificações sempre que ocorre uma mudança em um de seus itens e a partir disto pode-se executar uma ação.

## Observer Design Pattern



# Atividades e Exercícios Angular

---

- Observables estimulam a programação reativa (está incluído no pacote RxJS (Reactive Extensions), usado no Angular.
  - <https://github.com/Reactive-Extensions/RxJS>
  - <http://reactivex.io/>
- Programação reativa é um paradigma de programação orientado a fluxo de dados e propagação de mudança.
  - O Observer estimula a propagação de mudança, já que toda vez em que ocorre a mudança de estado, observadores são notificados.
  - Fluxo de dados ocorre porque os dados enviados podem ser controlados automaticamente, sem controle de estado.

# Atividades e Exercícios Angular

- As atualizações ocorrem automaticamente na medida em que novos dados surgem – programação assíncrona.

## Observables

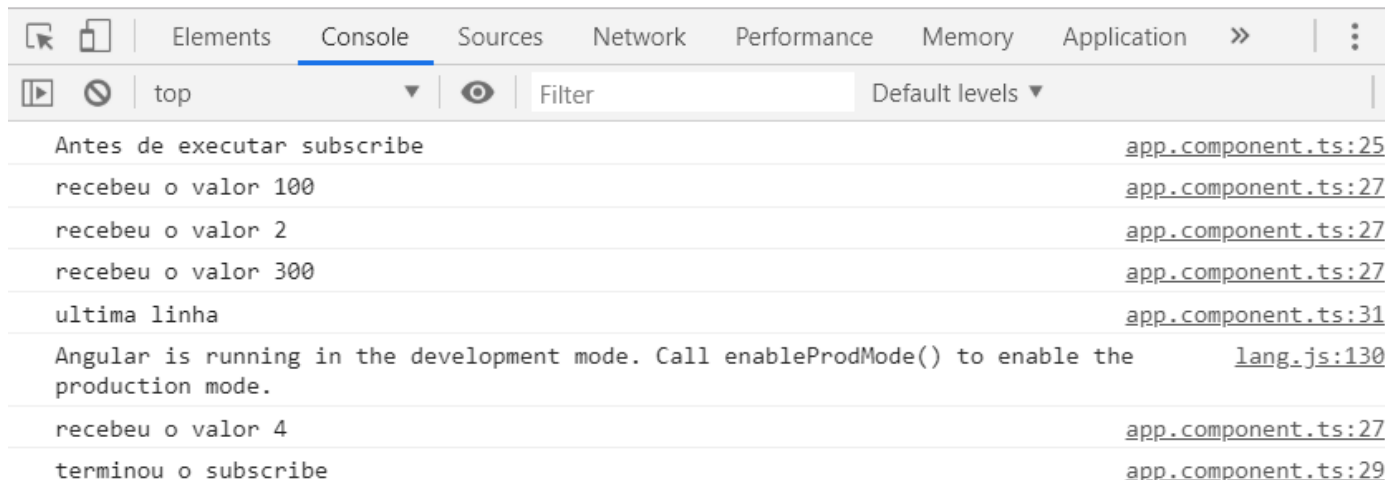


# Atividades e Exercícios Angular

## - Exemplo 11) Usando observables

```
ngOnInit() {  
  const observable = new Observable(subscriber => {  
    subscriber.next(100);  
    subscriber.next(2);  
    subscriber.next(300);  
    setTimeout(() => {  
      subscriber.next(4);  
      subscriber.complete();  
    }, 1000);  
  });  
}
```

```
console.log('Antes de executar subscribe');  
observable.subscribe({  
  next(x) { console.log('recebeu o valor ' + x); },  
  error(err) { console.error('Erro: ' + err); },  
  complete() { console.log('terminou o subscribe'); }  
});  
console.log('ultima linha');  
}
```





# Atividades e Exercícios Angular

## - Exemplo 11) Usando observables

```
ngOnInit() {  
  const observable = new Observable(subscriber => {  
    subscriber.next(100);  
    subscriber.next(2);  
    subscriber.next(300);  
    setTimeout(() => {  
      subscriber.next(4);  
      subscriber.complete();  
    }, 1000);  
  });  
}
```

setTimeout  
enviará a  
mensagem  
depois do  
tempo  
determinado  
, isto é, 1000  
milissegundo  
s.

A função subscriber  
define como obter  
valores e  
mensagens  
publicados no  
observer. Essa  
função é executada  
apenas quando o  
método subscribe  
for executado.

A cada chamada next,  
um novo valor é  
colocado no fluxo de  
dados

# Atividades e Exercícios Angular

## - Exemplo 11) Usando observables

```
console.log('Antes de executar subscribe');  
observable.subscribe({  
  next(x) { console.log('recebeu o valor ' + x); },  
  error(err) { console.error('Erro: ' + err); },  
  complete() { console.log('terminou o subscribe'); }  
});  
console.log('ultima linha');  
}
```

Quando o método subscribe é executado, recebe um conjunto de valores do observer, seja síncrono ou assíncrono.

A última linha executa antes do valor 4 exatamente pelo fato de subscribe ser assíncrono.

# Atividades e Exercícios Angular

## - Exemplo 12) Manipulando lista com observables

Cada vez que o observable produzir algum conteúdo, a lista chamada nomes irá receber este conteúdo

```
TS app.component.ts • app.component.html
8  })
9  export class AppComponent implements OnInit {
10     observable: Observable<string>;
11
12     nomes: Array<string> = [];
13
14     ngOnInit() {
15         this.observable = new Observable(subscriber => {
16             setInterval(() => {
17                 subscriber.next(this.makeid(5));
18             }, 10000);
19         });
20         let lista: Array<string> = [];
21         this.observable.subscribe({
22             next(x) { lista.push(x); },
23             error(err) { alert('ocorreu um erro '+err); }
24         });
25         this.nomes = lista;
```

A cada 10 segundos irá produzir uma string, usando o método makeid.

# Atividades e Exercícios Angular

## - Exemplo 12) Manipulando lista com observables

Para ficar mais interessante, a lista de nomes também receberá input do usuário

```
29 enviar(valor: string) {  
30     this.nomes.push(valor);  
31 }  
32  
33 makeid(length) {  
34     var text = "";  
35     var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";  
36     for (var i = 0; i < length; i++)  
37         text += possible.charAt(Math.floor(Math.random() * possible.length));  
38     return text;  
39 }  
40  
41 }  
42
```

Método construído só para gerar caracteres aleatoriamente

# Atividades e Exercícios Angular

## - Exemplo 12) Manipulando lista com observables

```
TS app.component.ts • app.component.html x
1 <input placeholder="digite um nome" #campo>
2 <button (click)="enviar(campo.value)">Enviar</button>
3 <ul *ngFor="let r of nomes">
4   <li>{{r}}</li>
5 </ul>
```

Teste x

localhost:4200

fernando Enviar

- carlos
- p2FR9
- fernando
- 5KhjO
- jIRwx
- xa8Uk
- Xw4wu
- BTvkP

# Atividades e Exercícios Angular

## - Exemplo 13) EventEmitter

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

1: powershell ▼



```
PS D:\Magistério\IFTM\Aulas\2019\2019-1\Projeto e Desenvolvimento de Software 2\projetos> ng  
new event
```

```
PS D:\Magistério\IFTM\Aulas\2019\2019-1\Projeto e Desenvolvimento de Software 2\projetos> cd  
.\event\  
PS D:\Magistério\IFTM\Aulas\2019\2019-1\Projeto e Desenvolvimento de Software 2\projetos\event  
> ng g c PaiComponent
```

# Atividades e Exercícios Angular

## - Exemplo 13) EventEmitter

```
TS pai-component.component.ts x
6 | styleUrls: [ './pai-component.component.css' ]
7 | })
8 | export class PaiComponentComponent implements OnInit {
9 |
10 |   familia: Object[];
11 |
12 |   constructor() {
13 |     this.familia = [
14 |       {
15 |         nome: 'Vitor',
16 |         sobrenome: 'Borges'
17 |       },
18 |       {
19 |         nome: 'Carlos',
20 |         sobrenome: 'Dantas'
21 |       }
22 |     ]
23 |   }
24 |
25 |   ngOnInit() {
26 |     console.log(this.familia);
27 |   }
28 | }
```

```
<> app.component.html x
1 | <h1>
2 |   {{title}}
3 | </h1>
4 | <app-pai-component></app-pai-component>
5 |
```

Elements Console Sources Network Performance

top Filter Defa

▼ (2) [{...}, {...}] ⓘ

- ▶ 0: {nome: "Vitor", sobrenome: "Borges"}
- ▶ 1: {nome: "Carlos", sobrenome: "Dantas"}
- length: 2
- ▶ \_\_proto\_\_: Array(0)

Angular is running in the development mode. Call enableProdMode production mode.

>

# Atividades e Exercícios Angular

## - Exemplo 13) EventEmitter

```
PS D:\Magistério\IFTM\Aulas\2019\2019-1\Projeto e Desenvolvimento de Software 2\projetos\event  
t> ng g c filho
```

```
TS filho.component.ts x pai-component.component.html  
1 import { Component, OnInit, Input } from '@angular/core';  
2  
3 @Component({  
4   selector: 'app-filho',  
5   templateUrl: './filho.component.html',  
6   styleUrls: ['./filho.component.css']  
7 })  
8 export class FilhoComponent implements OnInit {  
9  
10  @Input() recebeFamilia;  
11  
12  constructor() { }  
13  
14  ngOnInit() {  
15    console.log(this.recebeFamilia)  
16  }  
17  
18 }
```

```
S filho.component.ts x pai-component.component.html x TS pai-co  
1 <p>  
2   pai-component works!  
3 </p>  
4 <app-filho [recebeFamilia]="familia"></app-filho>
```



# Atividades e Exercícios Angular

---

- Exemplo 13) EventEmitter

- Exercício: Implemente um loop no html do componente filho, imprimindo os valores do objeto recebido.

# Atividades e Exercícios Angular

## - Exemplo 13) EventEmitter

```
TS filho.component.ts x pai-component.component.html TS pai-component.component.ts
10 @Input() recebeFamilia;
11
12 @Output() respostaFamilia = new EventEmitter();
13
14 constructor() { }
15
16 ngOnInit() {
17   console.log(this.recebeFamilia)
18   console.log("Objeto familia recebido do componente pai via input: ",
19   this.recebeFamilia)
20 }
21
22 feedback() {
23   console.log("Resposta para o componente pai",
24   this.respostaFamilia.emit({"nome":"Raimundo","sobrenome":"nonato"}));
25 }
26
```

```
<> filho.component.html x
1 <p>
2   filho works!
3 </p>
4
5 <button (click)="feedback()">enviar para o pai</button>
6
7
```

# Atividades e Exercícios Angular

## - Exemplo 13) EventEmitter

pai-component.component.ts

```
18     {
19       nome: 'Carlos',
20       sobrenome: 'Dantas'
21     }
22   ]
23 }
24
25 ngOnInit() {
26   console.log(this.familia);
27 }
28
29 receberFeedback(respostaFilho) {
30   console.log('Foi emitido o evento e chegou no pai >>> ', respostaFilho);
31 }
32
33 }
```

pai-component.component.html

```
1  Pai!!!
2  <ul>
3    <li *ngFor="let f of familia">
4      {{f.nome}} {{f.sobreNome}}
5    </li>
6  </ul>
7
8  <app-filho [recebeFamilia]="familia"
9    (respostaFamilia)="receberFeedback($event)"></app-filho>
10
```

# Atividades e Exercícios Angular

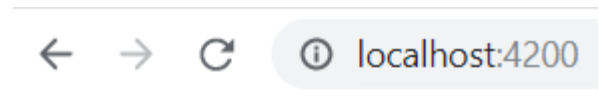
## - Exemplo 14) Estoque produtos

```
TS app.component.ts x  app.component.html
7  })
8  export class AppComponent {
9      products = [];
10     title = 'Products';
11     ngOnInit(): void {
12         this.products = this.getProducts();
13     }
14     getProducts() {
15         return [
16             { 'id': '1', 'title': 'Screw Driver', 'price': 400, 'stock': 11 },
17             { 'id': '2', 'title': 'Nut Volt', 'price': 200, 'stock': 5 },
18             { 'id': '3', 'title': 'Resistor', 'price': 78, 'stock': 45 },
19             { 'id': '4', 'title': 'Tractor', 'price': 20000, 'stock': 1 },
20             { 'id': '5', 'title': 'Roller', 'price': 62, 'stock': 15 },
21         ];
22     }
23 }
```

# Atividades e Exercícios Angular

## - Exemplo 14) Estoque produtos

```
TS app.component.ts    app.component.html x
1  <div class="container">
2    <br />
3    <h1 class="text-center">{{title}}</h1>
4    <table class="table">
5      <thead>
6        <th>Id</th>
7        <th>Title</th>
8        <th>Price</th>
9        <th>Stock</th>
10     </thead>
11     <tbody>
12       <tr *ngFor="let p of products">
13         <td>{{p.id}}</td>
14         <td>{{p.title}}</td>
15         <td>{{p.price}}</td>
16         <td>{{p.stock}}</td>
17       </tr>
18     </tbody>
```



## Products

Id	Title	Price	Stock
1	Screw Driver	400	11
2	Nut Volt	200	5
3	Resistor	78	45
4	Tractor	20000	1
5	Roller	62	15

# Atividades e Exercícios Angular

## - Exemplo 14) Estoque produtos

```
create src\app\filho\filho.component.html
create src\app\filho\filho.component.spec.ts
create src\app\filho\filho.component.ts
update src\app\app.module.ts
PS D:\Magistério\IFTM\Aulas\2019\2019-1\Projeto e Desenvolvimento de Software 2\event> ng g c
StockStatus
```

```
TS stock-status.component.ts x stock-status.component.html
7  })
8  export class StockStatusComponent {
9    @Input() stock: number;
10   @Input() productId: number;
11   @Output() stockValueChange = new EventEmitter();
12   color = '';
13   updatedstockvalue: number;
14
15   stockValueChanged() {
16     this.stockValueChange.emit(
17       { id: this.productId, updatdstockvalue: this.updatedstockvalue });
18     this.updatedstockvalue = null;
19   }
```

```
20
21   ngOnChanges() {
22     if (this.stock > 10) {
23       this.color = 'green';
24     } else {
25       this.color = 'red';
26     }
27   }
28 }
```

# Atividades e Exercícios Angular

## - Exemplo 14) Estoque produtos

<> stock-status.component.html x

```
1 <input type='number' [(ngModel)]='updatedstockvalue' />
2 <button class='btn btn-primary'
3   [style.background]='color'
4   (click)="stockValueChanged()">Change Stock Value</button>
```

<> app.component.html x

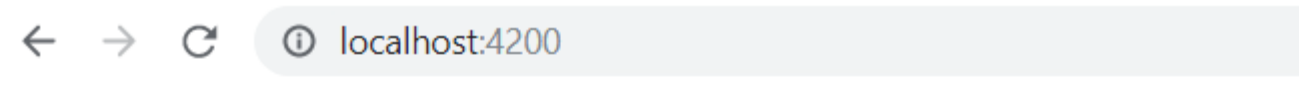
```
10 </thead>
11 <tbody>
12 <tr *ngFor="let p of products">
13   <td>{{p.id}}</td>
14   <td>{{p.title}}</td>
15   <td>{{p.price}}</td>
16   <td>{{p.stock}}</td>
17   <td><app-stock-status
18     [productId]='p.id'
19     [stock]='p.stock'
20     (stockValueChange)="changeStockValue($event)">
21   </app-stock-status>
22 </td>
```

TS app.component.ts x

```
17 { 'id': '2', 'title': 'Nut Volt', 'price': 200, 'stock': 5 },
18 { 'id': '3', 'title': 'Resistor', 'price': 78, 'stock': 45 },
19 { 'id': '4', 'title': 'Tractor', 'price': 20000, 'stock': 1 },
20 { 'id': '5', 'title': 'Roller', 'price': 62, 'stock': 15 },
21 ];
22 }
23
24 productToUpdate: any;
25 changeStockValue(p) {
26   this.productToUpdate = this.products.find(this.findProducts, [p.id]);
27   this.productToUpdate.stock = this.productToUpdate.stock + p.updatedstockva
28 }
29 findProducts(p) {
30   return p.id === this[0];
31 }
32 }
```

# Atividades e Exercícios Angular

## - Exemplo 14) Estoque produtos



### Products

Id	Title	Price	Stock		
1	Screw Driver	400	8	<input type="text"/>	Change Stock Value
2	Nut Volt	200	12	<input type="text"/>	Change Stock Value
3	Resistor	78	45	<input type="text"/>	Change Stock Value
4	Tractor	20000	1	<input type="text"/>	Change Stock Value
5	Roller	62	15	<input type="text"/>	Change Stock Value



# REFERÊNCIAS

---

- GUEDES, Thiago. Crie aplicações com Angular. Casa do Código, 2018.
- GRONER, Loiane. Curso Angular. Disponível em: <https://www.youtube.com/watch?v=tPOMG0D57S0&list=PLGxZ4Rq3BOBoSRcKWEdQACbUCNWLczg2G>. Acesso em 02/04/2019.
- <https://medium.com/tableless/entendendo-rxjs-observable-com-angular-6f607a9a6a00>
- <https://dzone.com/articles/understanding-output-and-eventemitter-in-angular>