UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

3416-003 Web Development Using Angular 8 and TypeScript

Module 5: Components and Databinding DeepDived

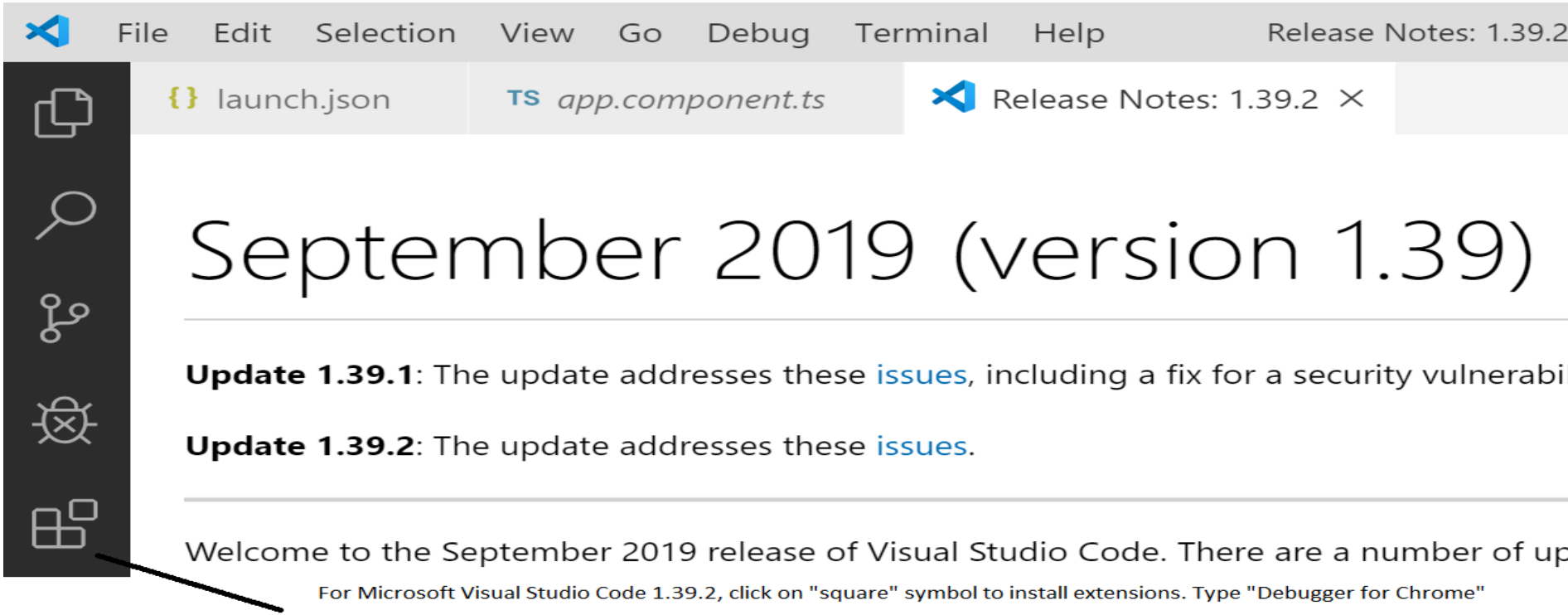
UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 1: Section 1

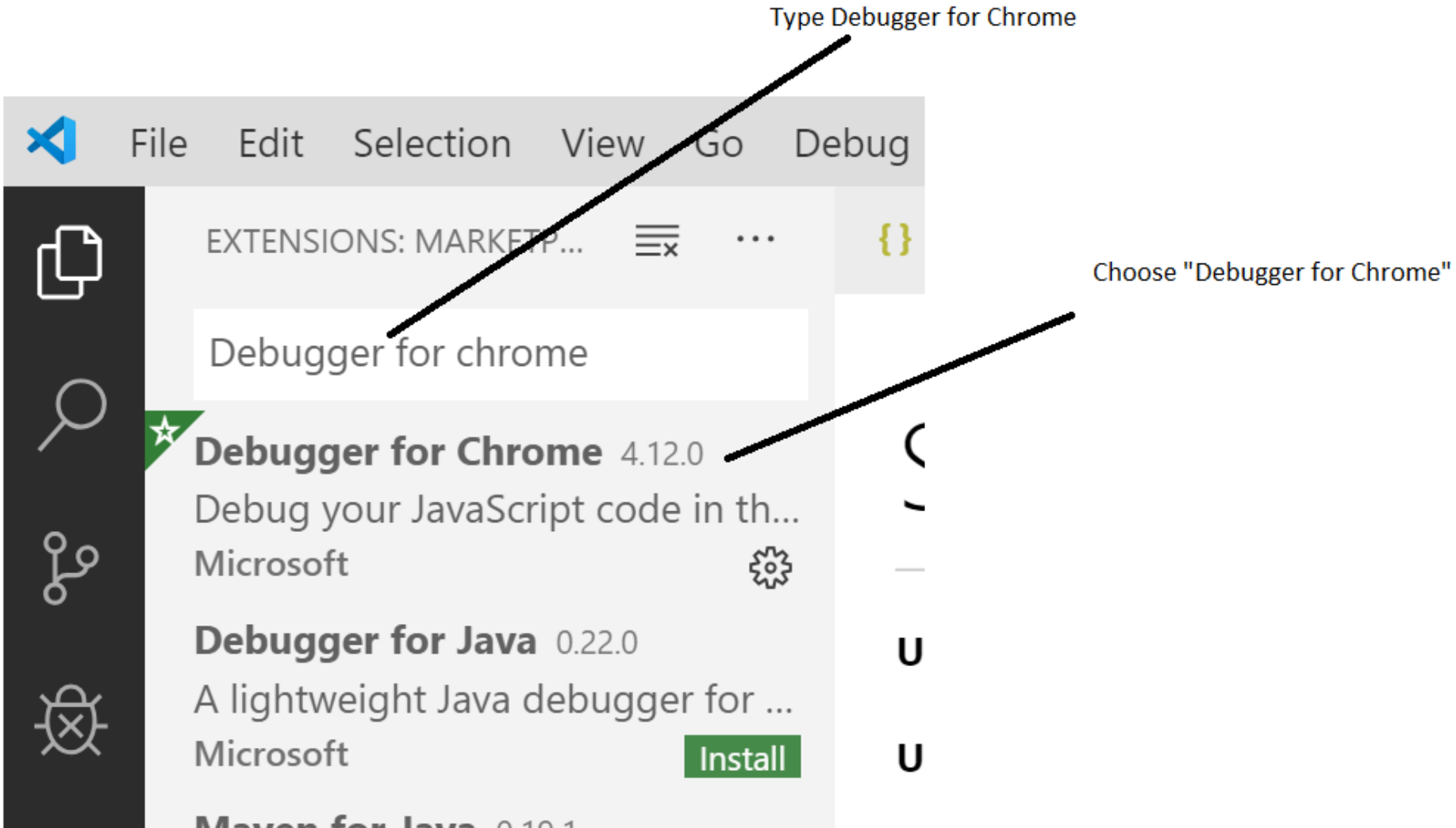
What Is This Course About

Debugger for Chrome

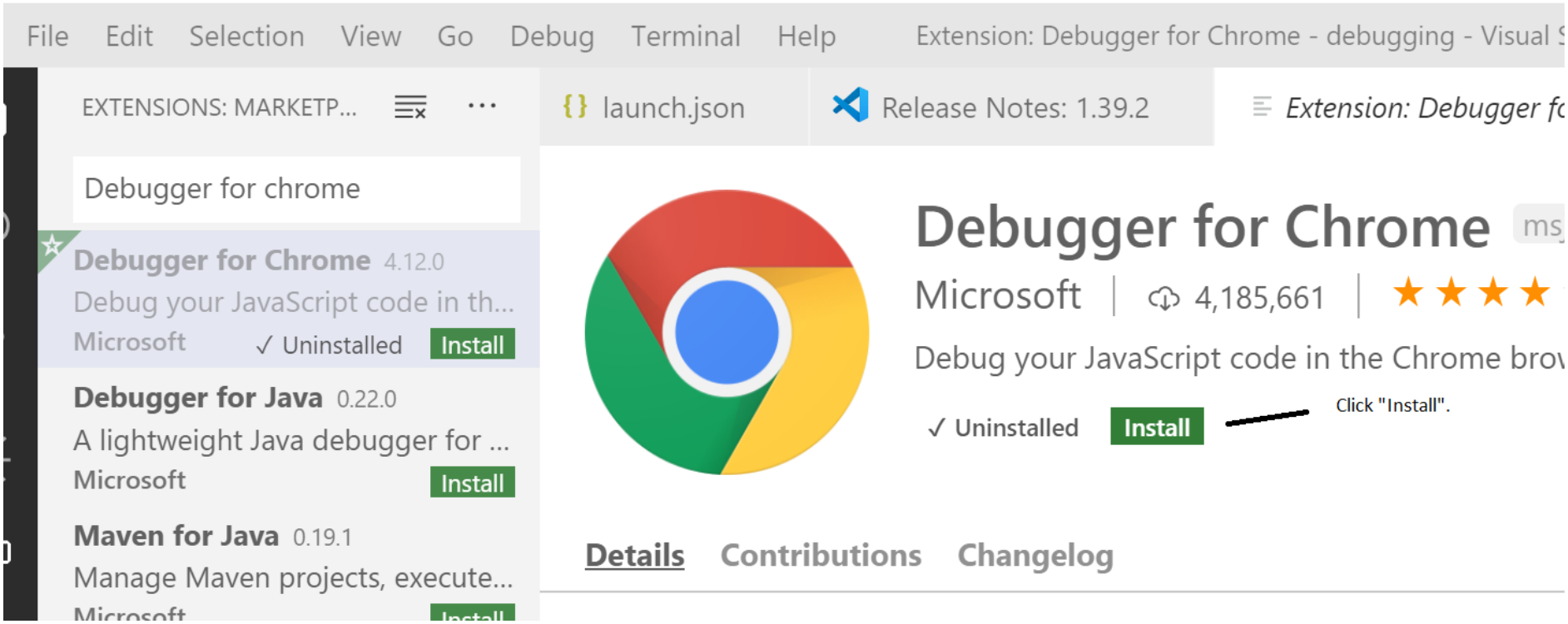
- **First Way to debug Angular Code**
- Install debugger for chrome for Visual Studio Code



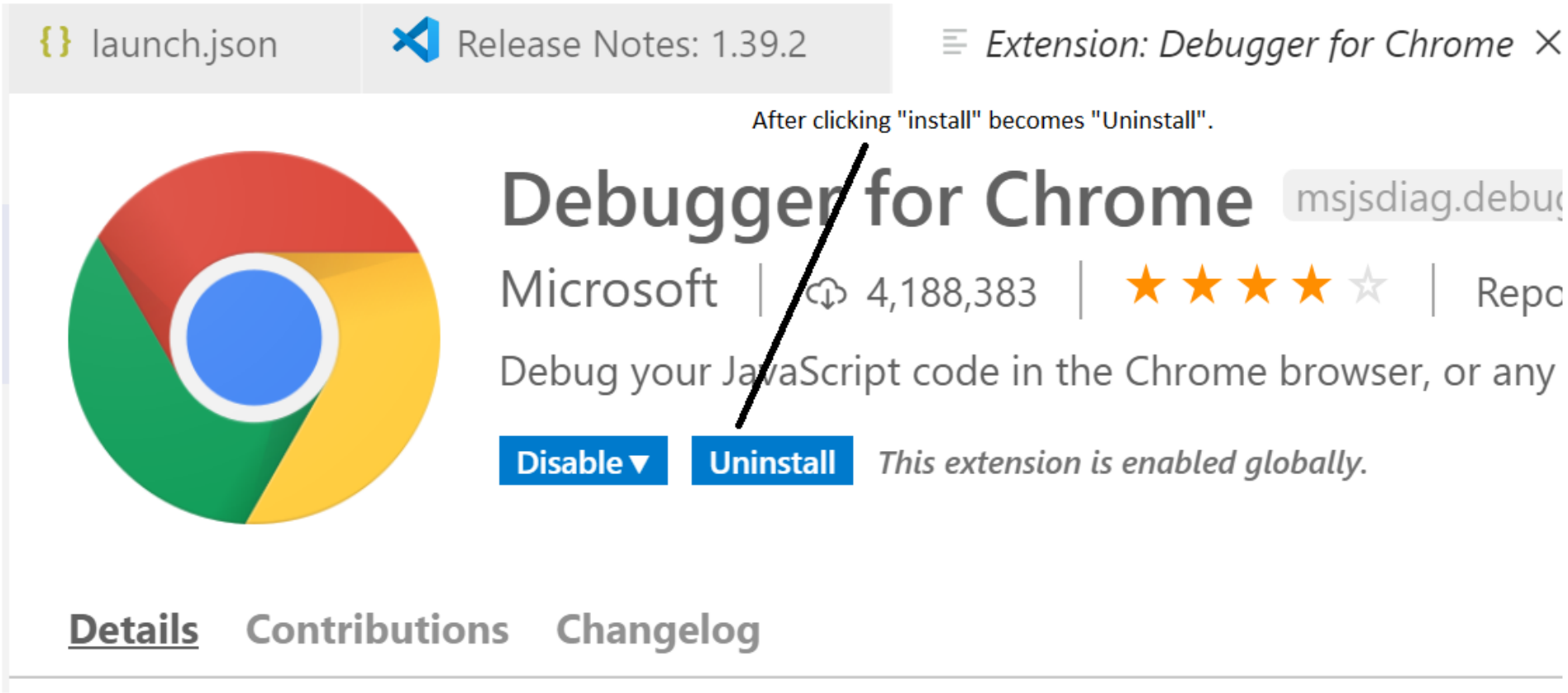
Debugger for Chrome



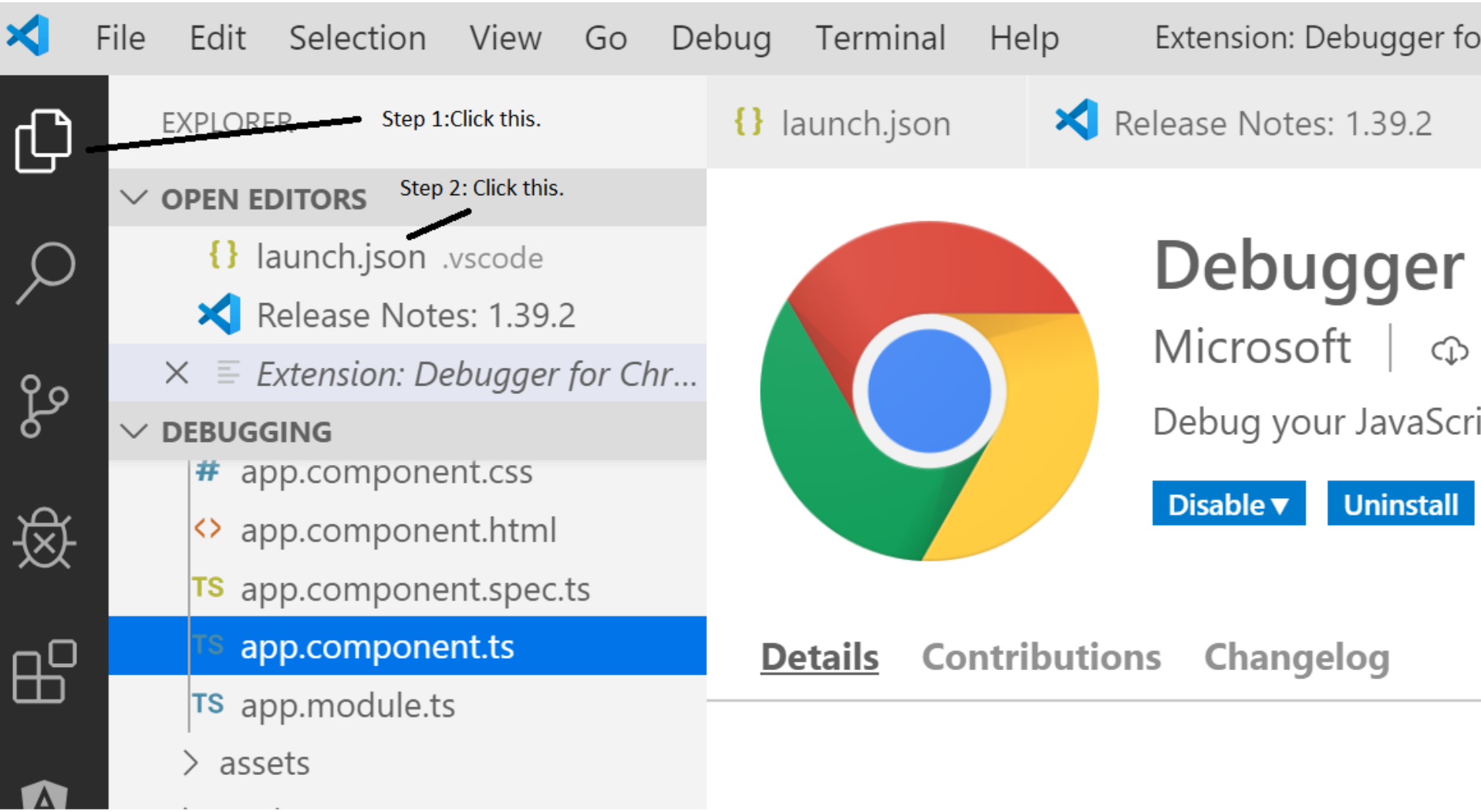
Debugger for Chrome



Debugger for Chrome

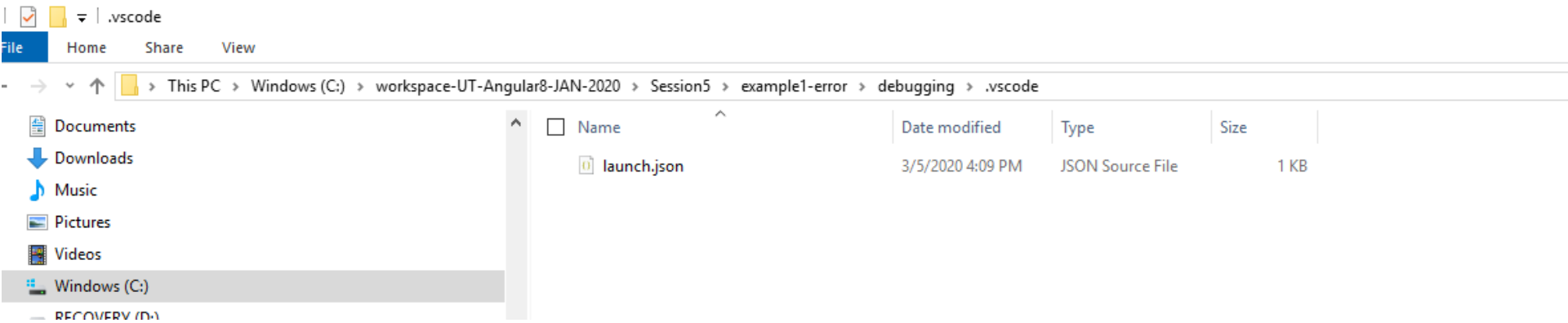


Debugger for Chrome



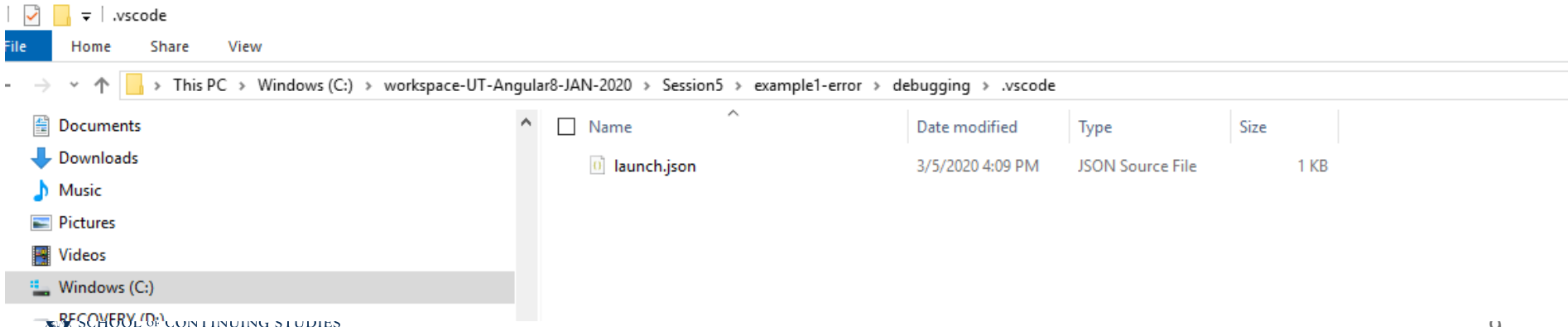
Debugger for Chrome

- **This is for Visual Studio Code 1.42.1**
- Inside ".vscode/launch.json".
- **C:\workspace-UT-Angular8-SEPT-2020\Session5\example1-error\debugging\.vscode**



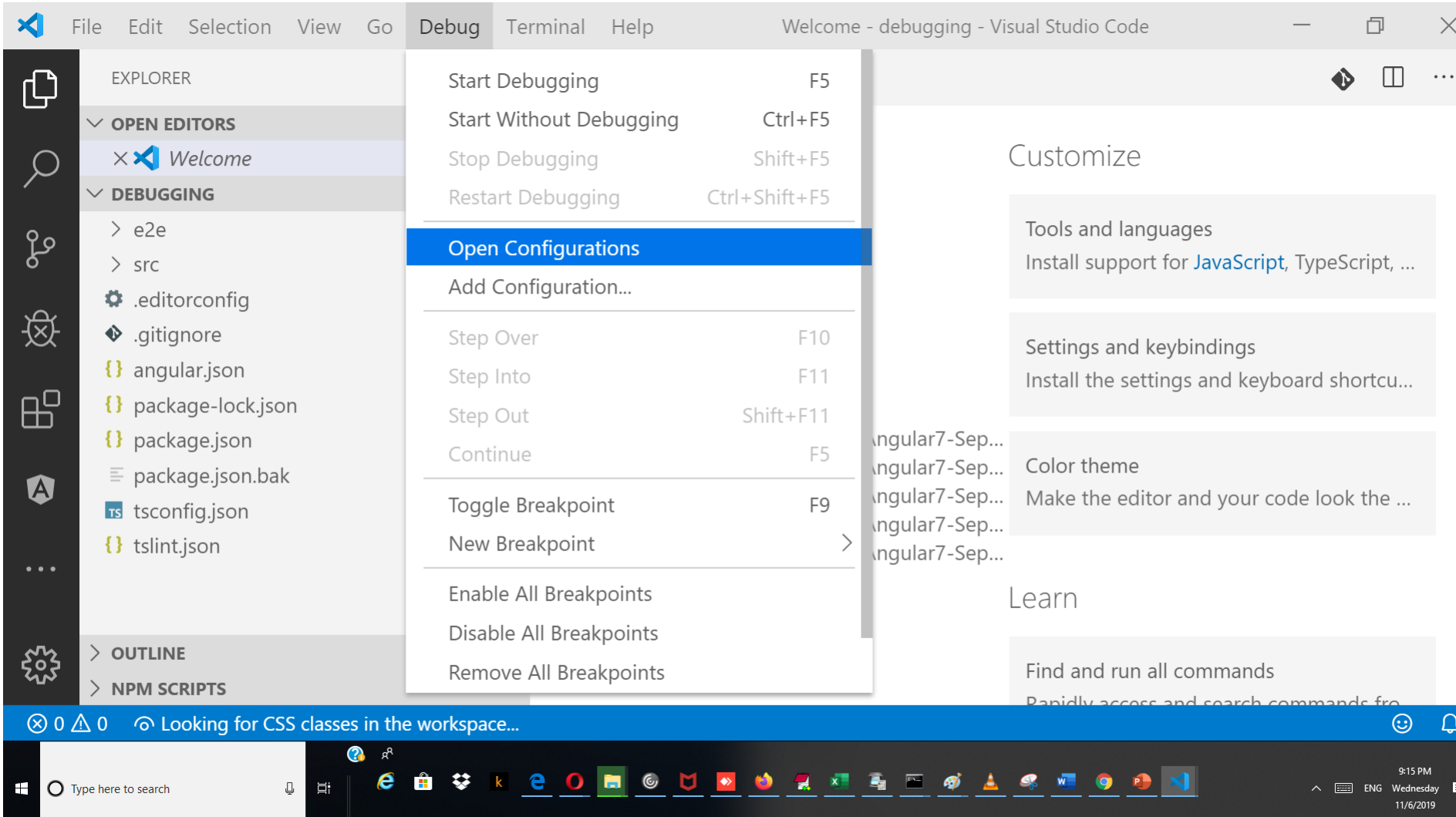
Debugger for Chrome

- **This is for Visual Studio Code 1.42.1**
- **Copy C:\AlbertLam\Angular8-UT-SEPT-2020 \Session5\code-before-npm-install\example1-error\debugging to C:\workspace-UT-Angular8-SEPT-2020 \Session5**
- Inside ".vscode/launch.json".
- C:\ workspace-UT-Angular8-SEPT-2020 \Session5\example1-error**debugging\node_modules\.vscode**



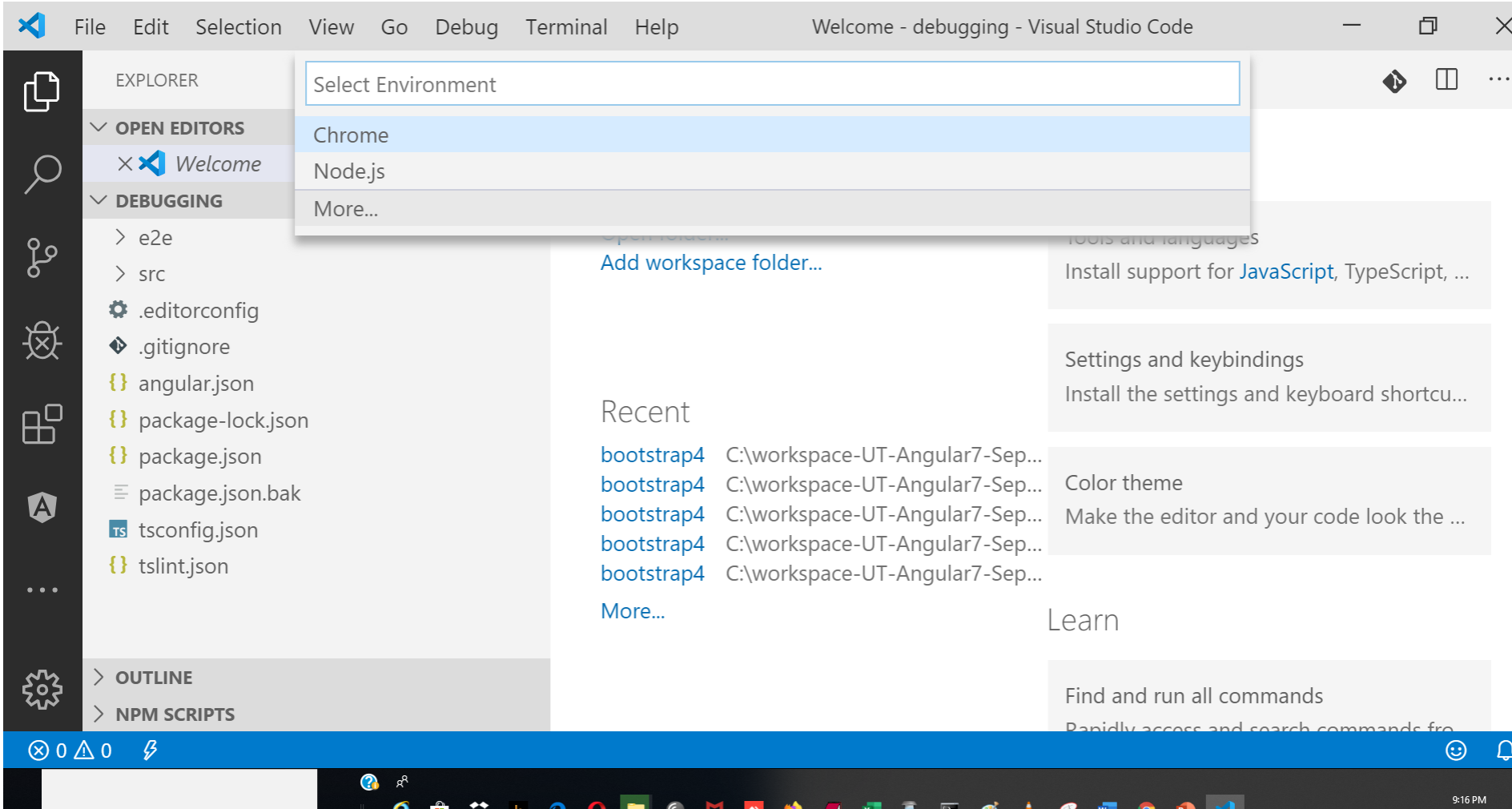
Debugger for Chrome

- **Click “Debug/Open Configurations”:**



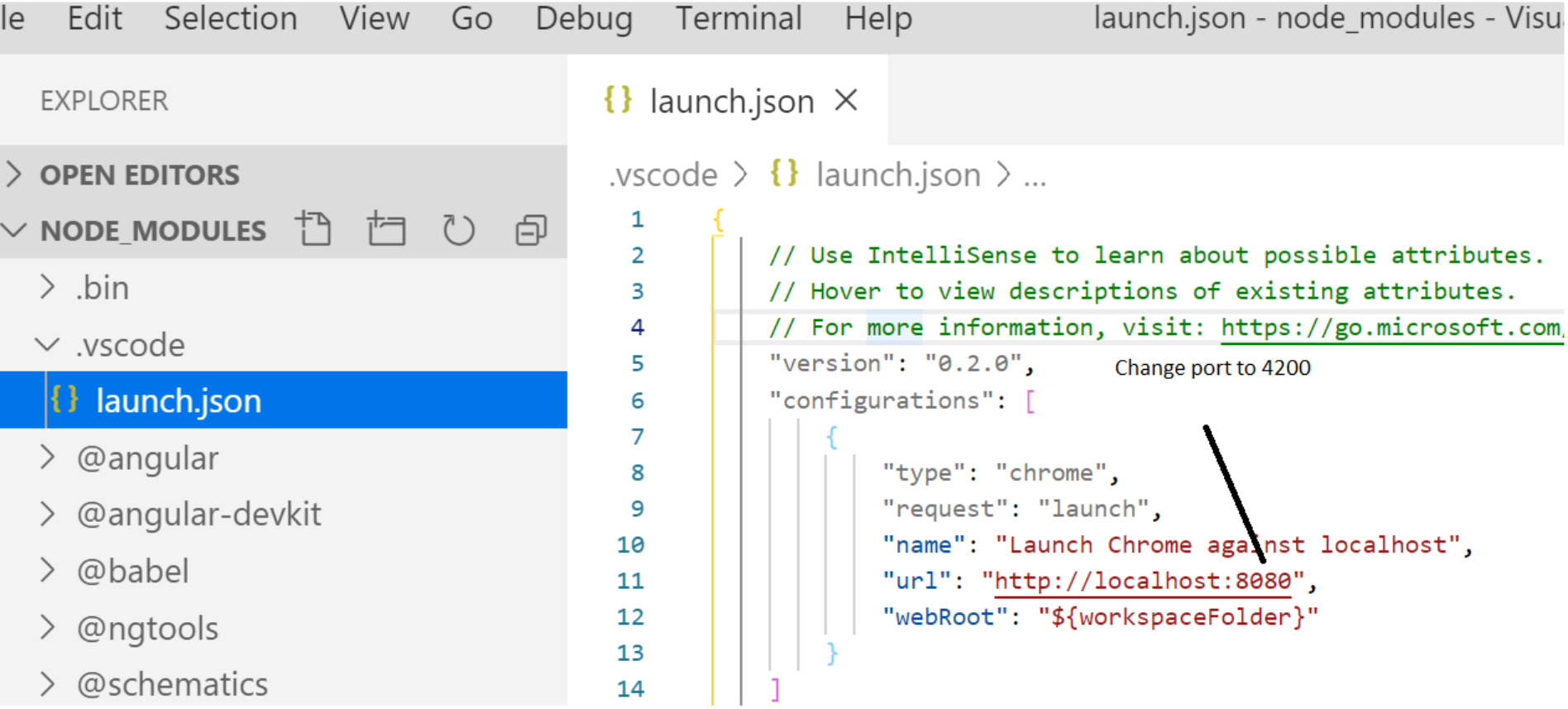
Debugger for Chrome

- **Click “Debug/Open Configurations” and choose “chrome”:**



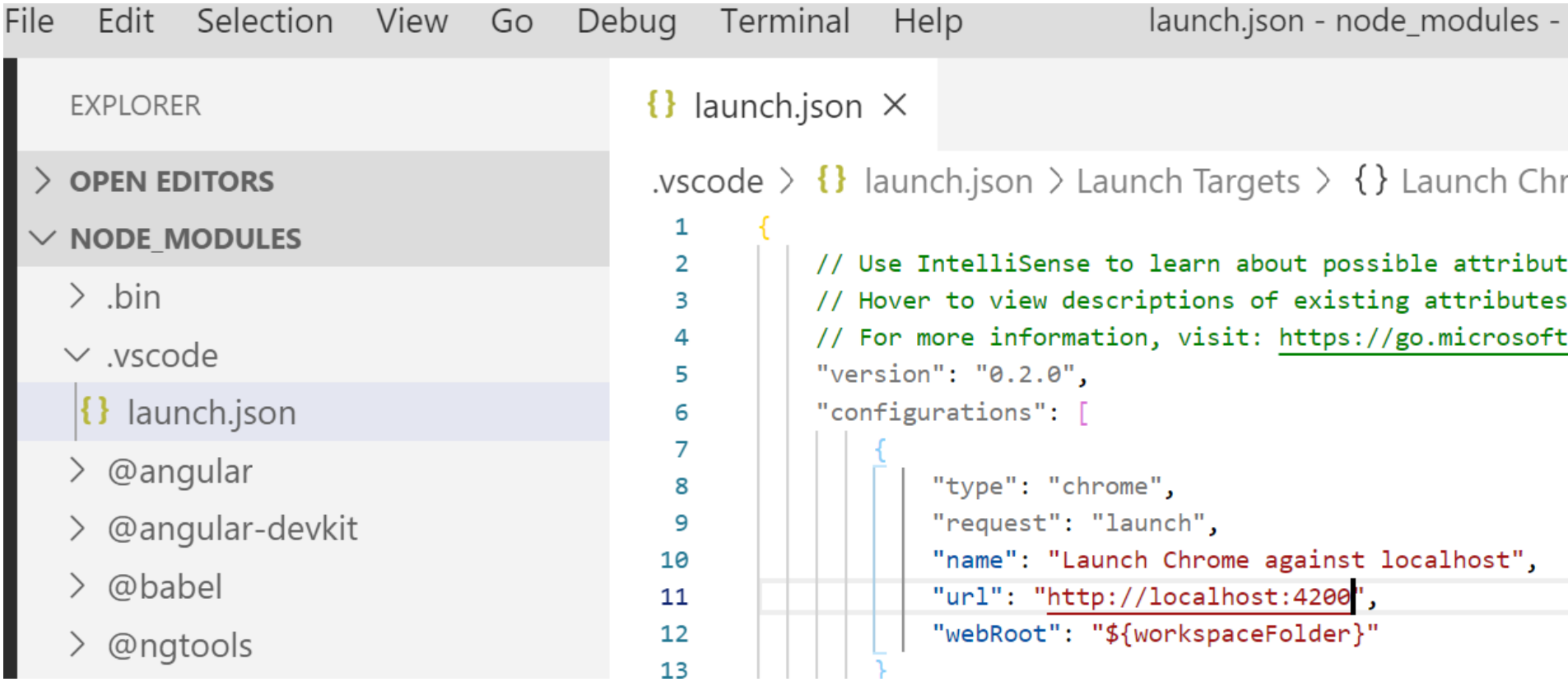
Debugger for Chrome

- **This is for Visual Studio Code 1.42.1**
- Edit launch.json



Debugger for Chrome

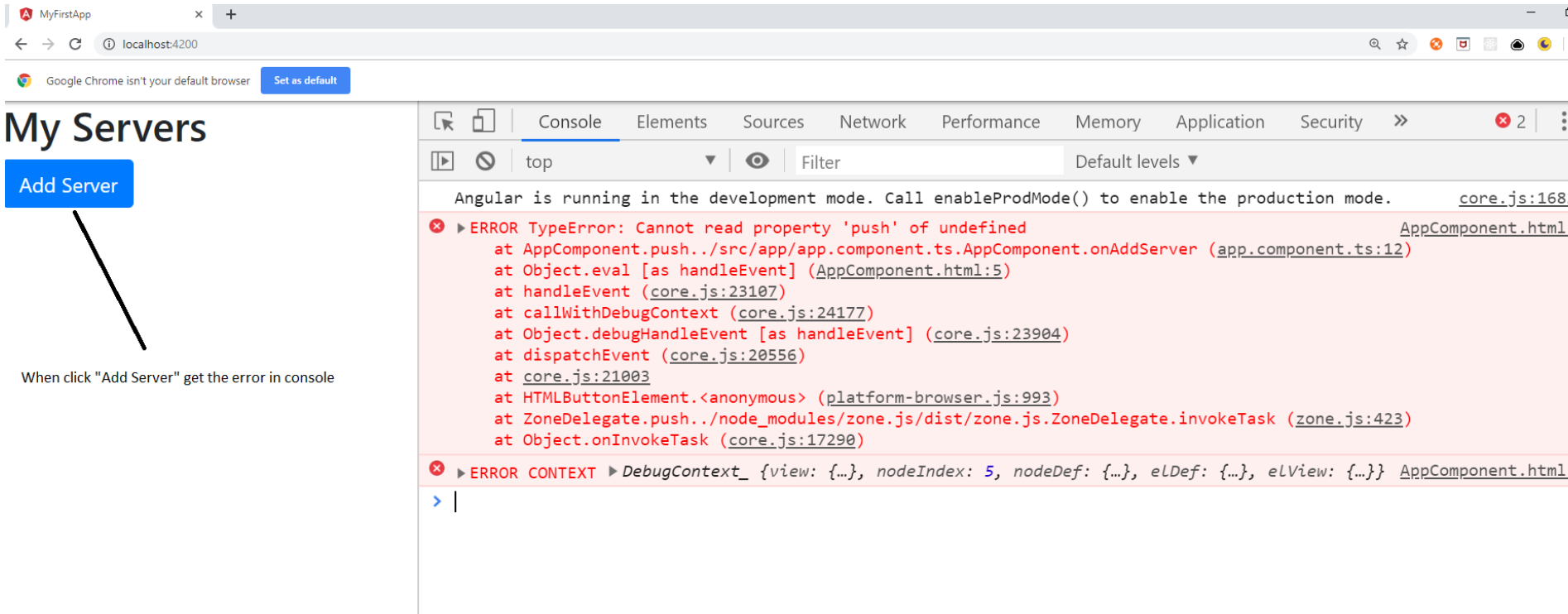
- **This is for Visual Studio Code 1.42.1**
- Edit launch.json



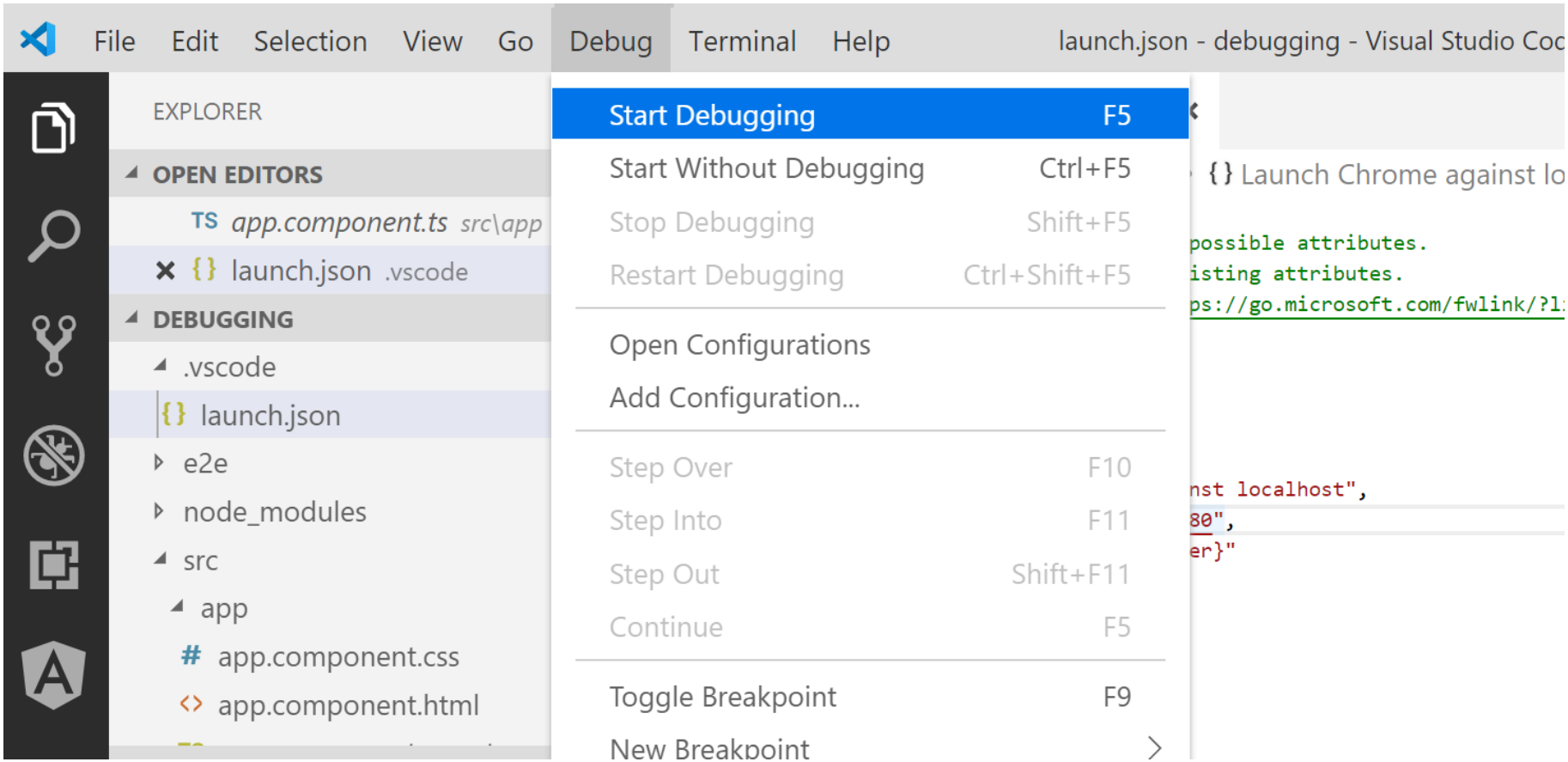
Debugger for Chrome

- Download example1-error from google drive under session5\code.
- Run “**npm install**” and “**ng serve –o**”

```
C:\workspace-UT-Angular7-Sept-2019\Session5\example1-error\debugging>npm install
```

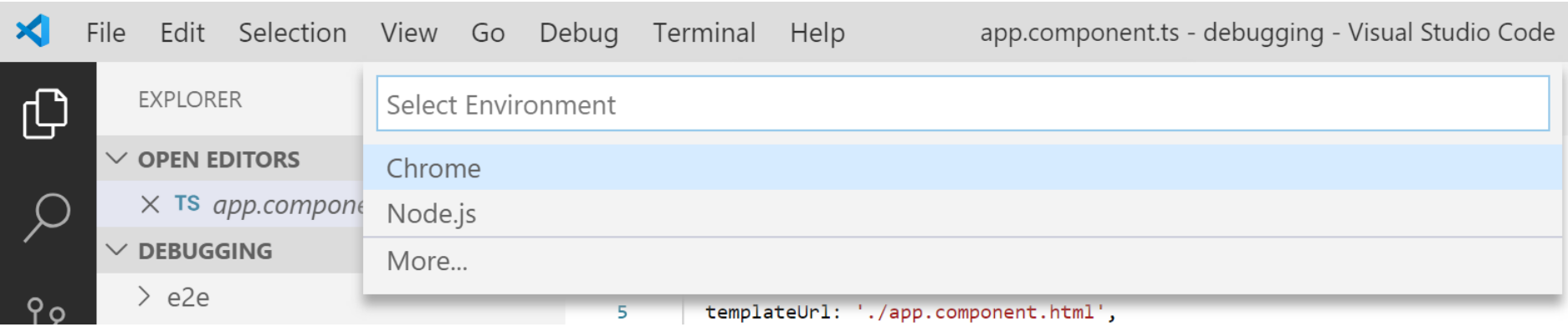


Debugger for Chrome

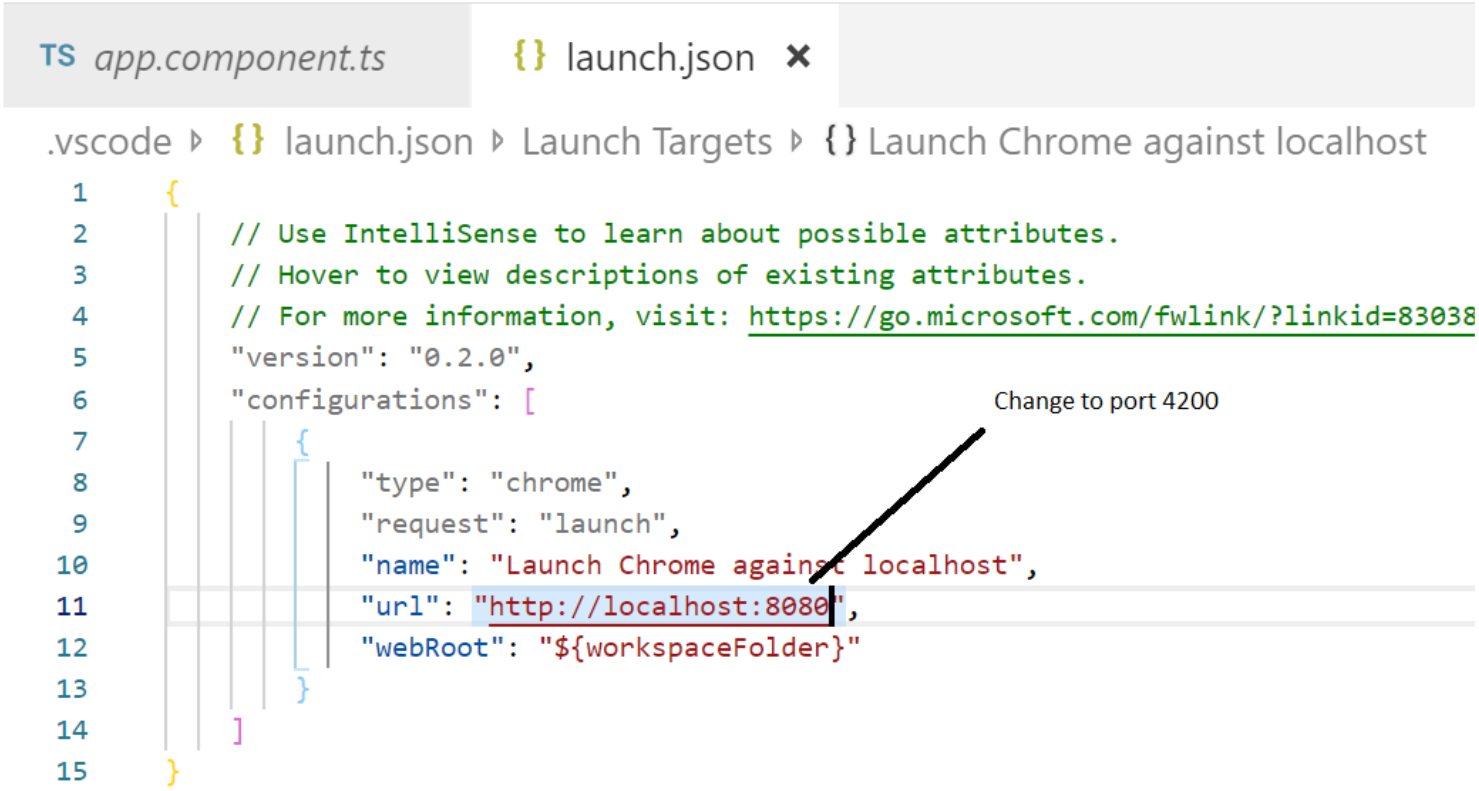


Debugger for Chrome

- Choose “chrome”:



Debugger for Chrome



Debugger for Chrome

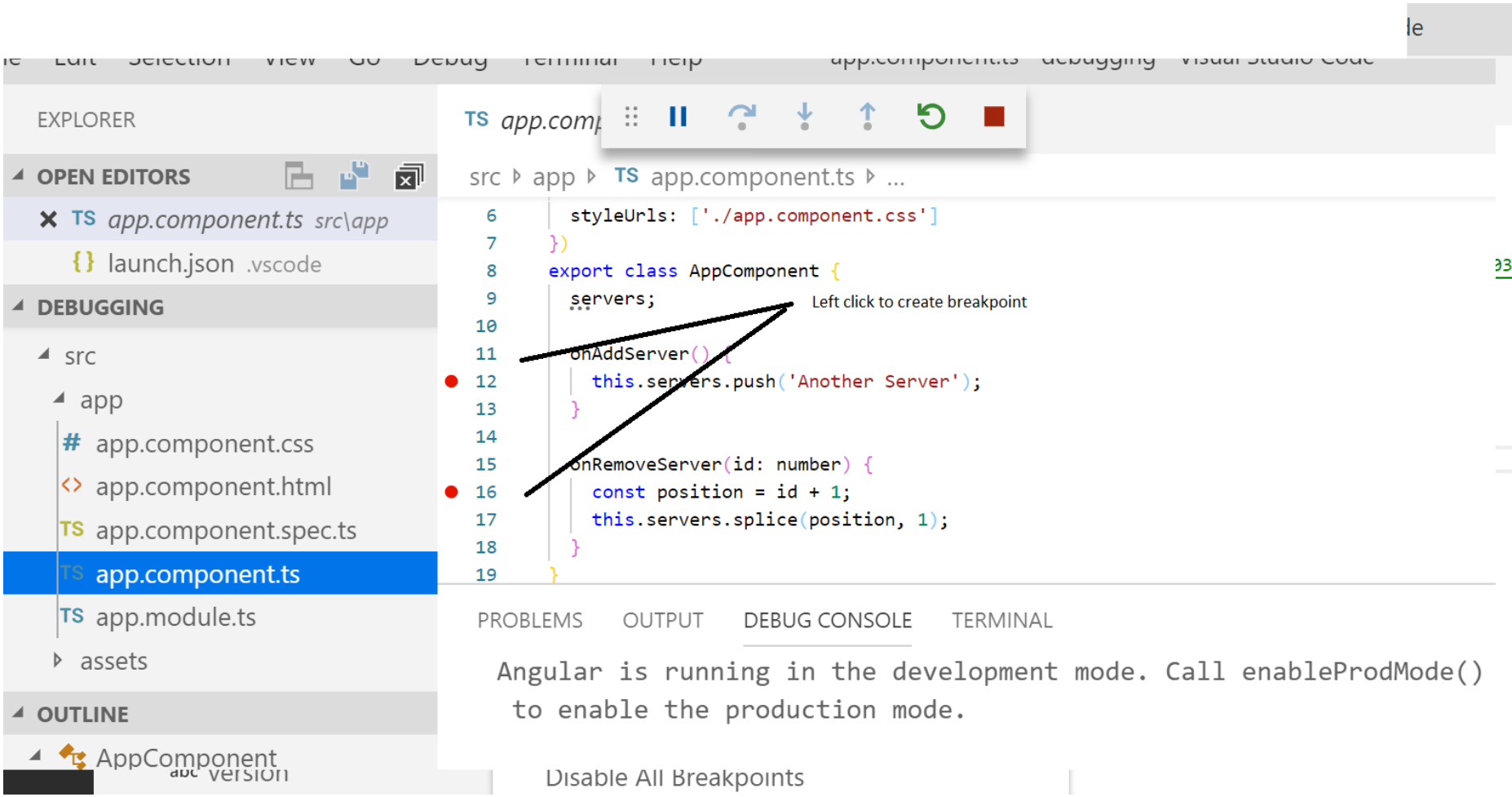
TS app.component.ts ✕ | { } launch.json ✕

.vscode ▸ { } launch.json ▸ Launch Targets ▸ { } Launch Chrome against localhost

```
1   {
2       // Use IntelliSense to learn about possible attributes.
3       // Hover to view descriptions of existing attributes.
4       // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
5       "version": "0.2.0",
6       "configurations": [
7           {
8               "type": "chrome",
9               "request": "launch",
10              "name": "Launch Chrome against localhost",
11              "url": "http://localhost:4200",
12              "webRoot": "${workspaceFolder}"
13          }
14      ]
15  }
16
```

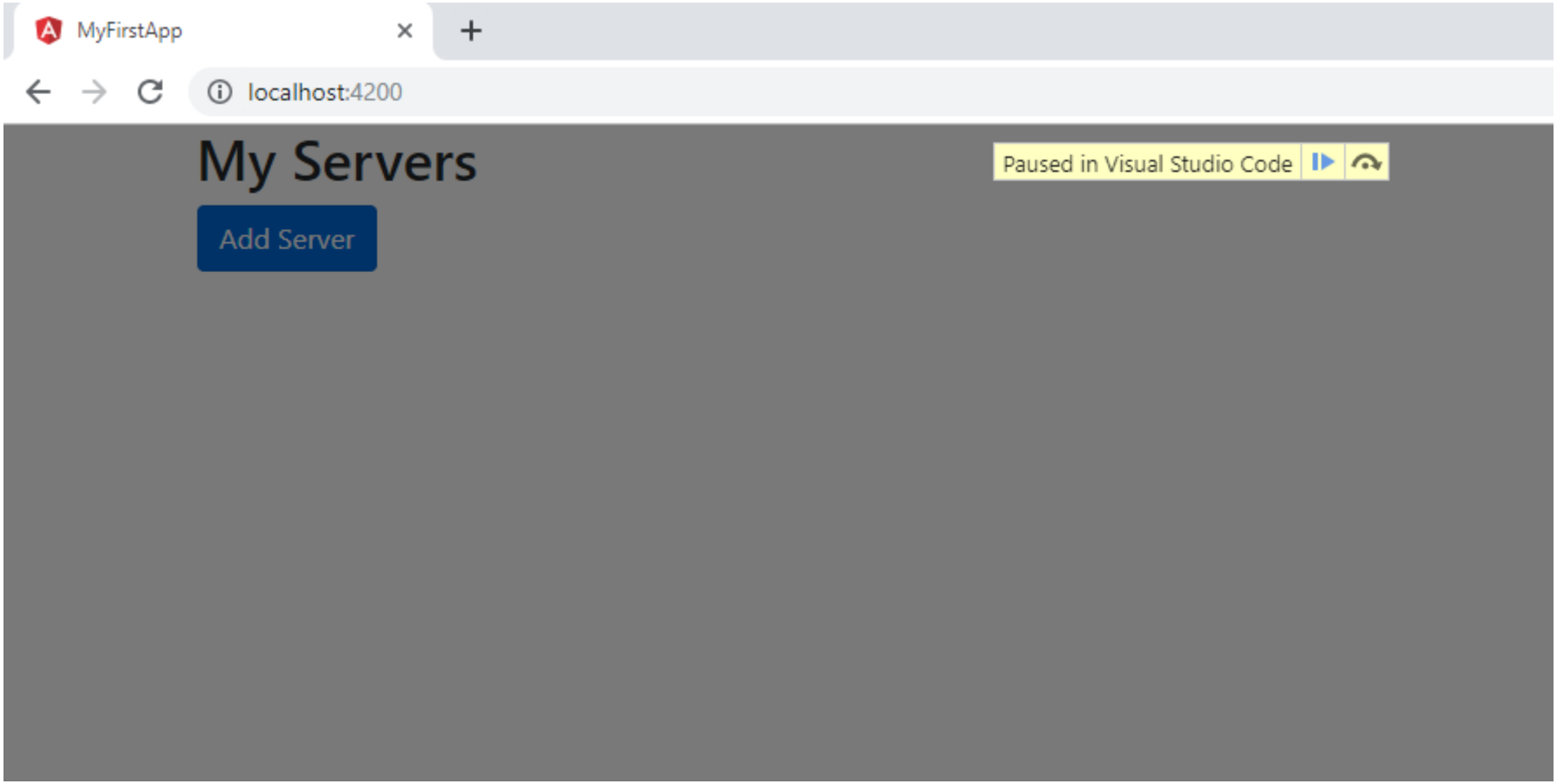
Debugger for Chrome

- Right click to place breakpoints.



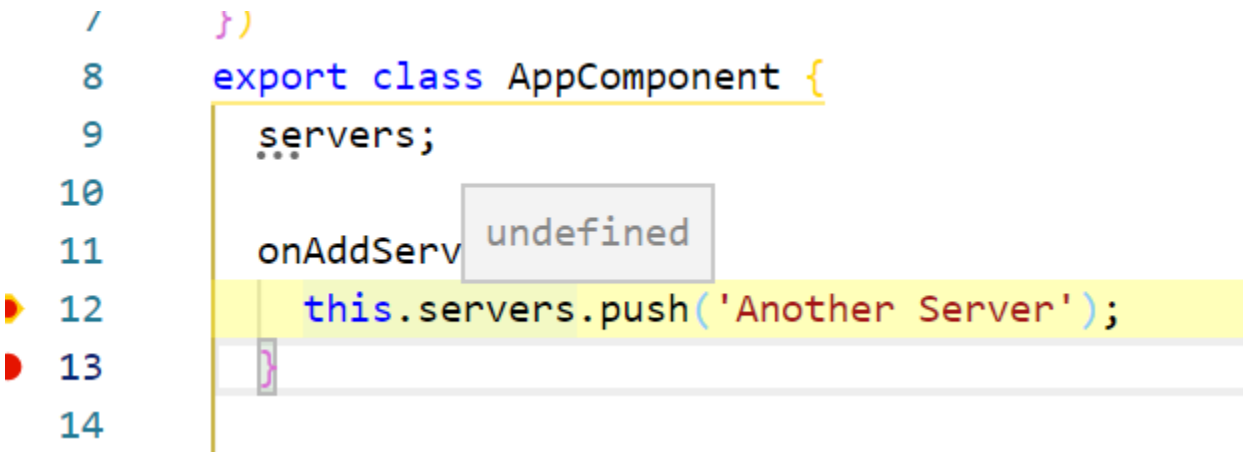
Debugger for Chrome

- Click “Add Server”:



Debugger for Chrome

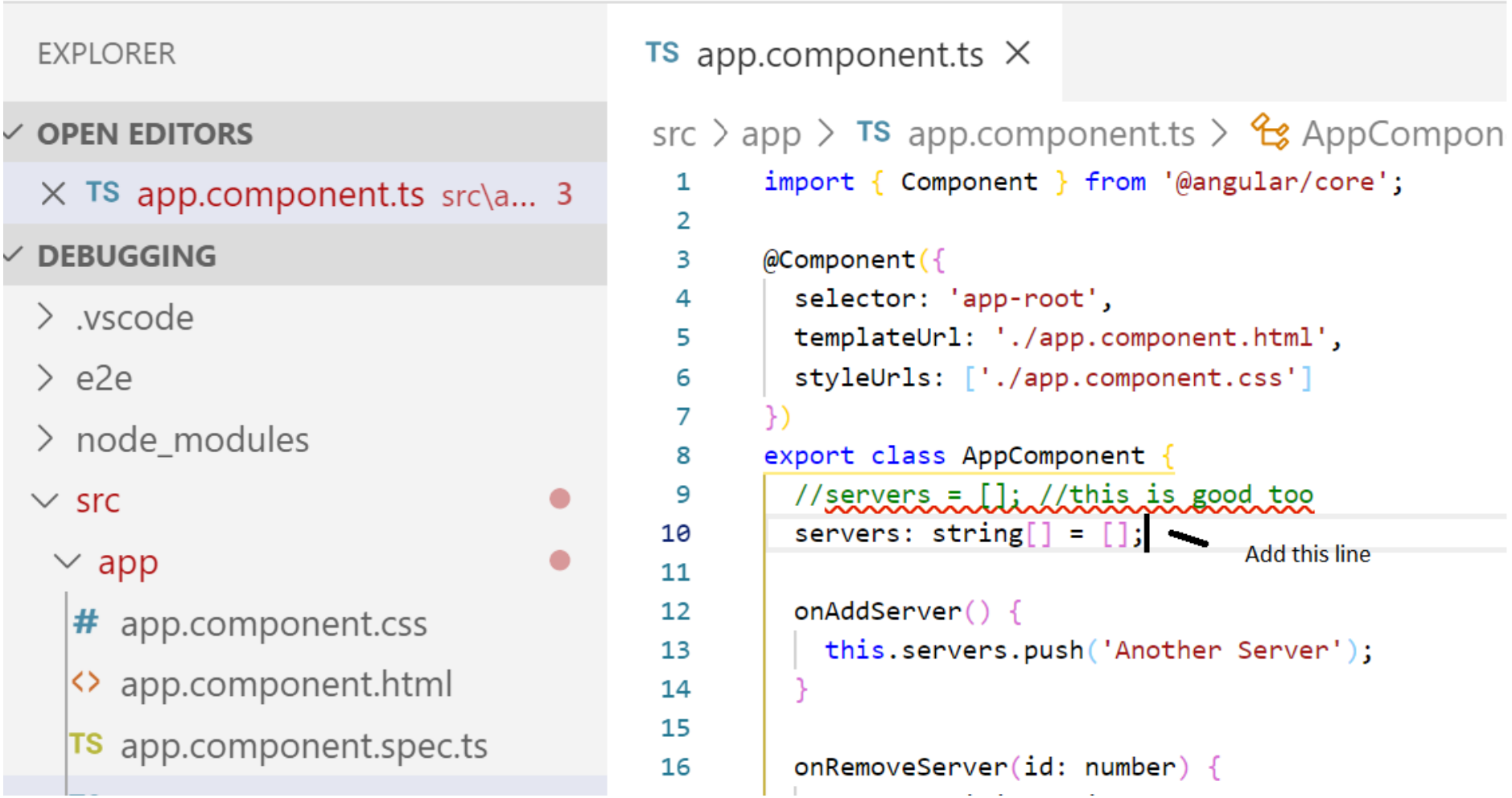
- Get “servers” is undefined.



example1-solution

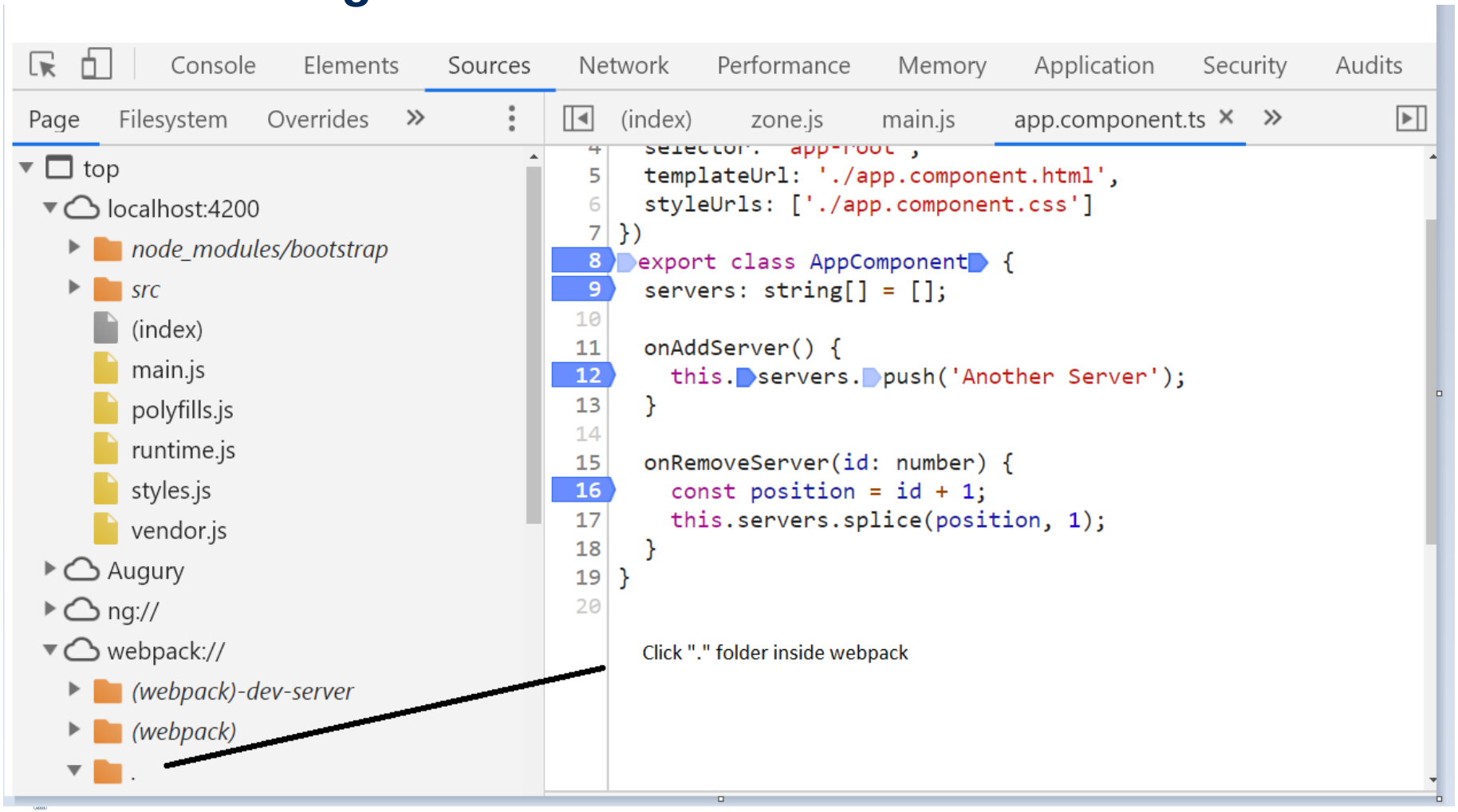
- Copy C:\workspace-UT-Angular8-SEPT-2020\session5\example1-error\debugging to C:\workspace-UT-Angular8-SEPT-2020\session5\example1-solution

example1-solution

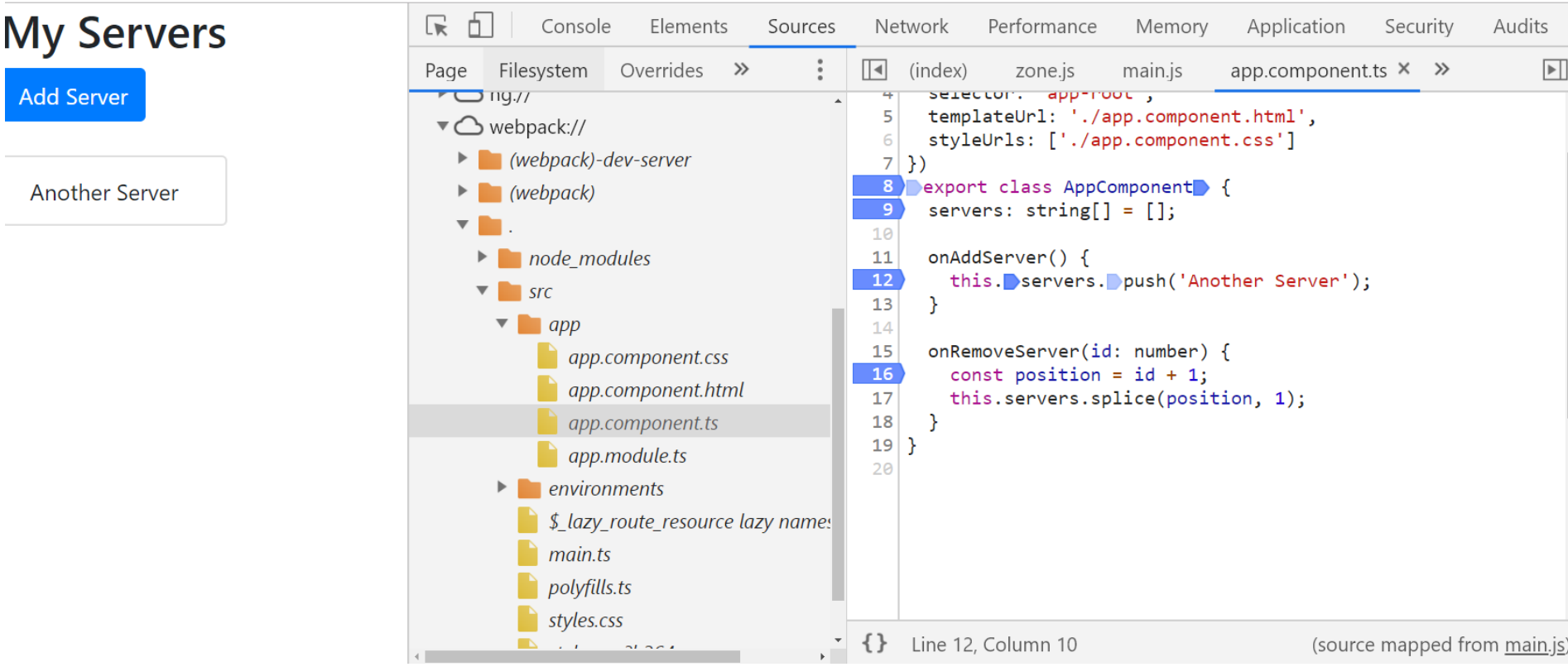


How to debug through Google Chrome

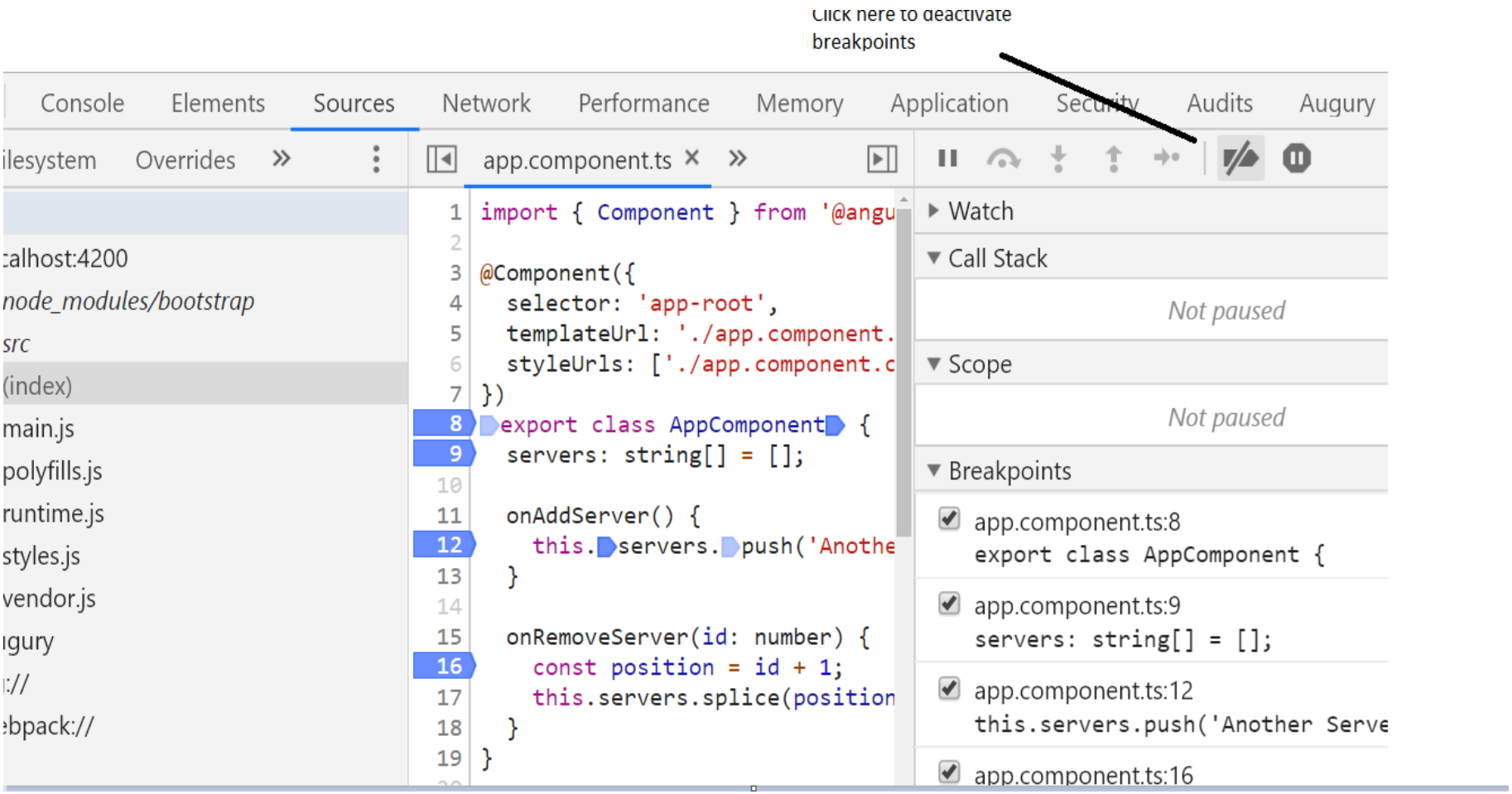
- **Second Way to debug Angular Code:**
- **Go to Google Chrome**



How to debug through Google Chrome



How to debug through Google Chrome



example1-solution(Getting the index when using ngFor”

- C:\workspace-UT-Angular8-SEPT-2020 \session5\example1-solution\debugging\src\app\app.component.html

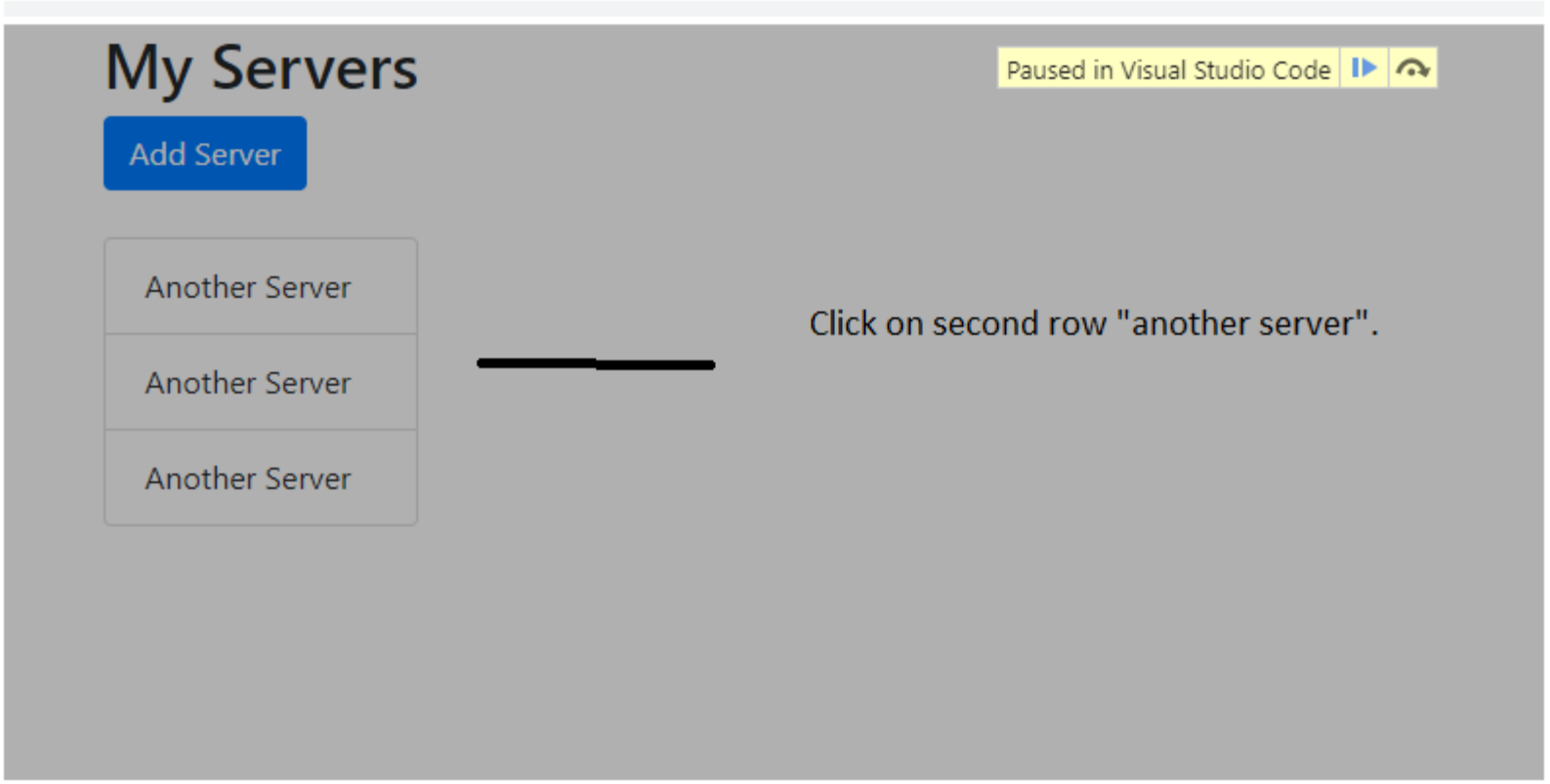
src ▸ app ▸ app.component.html ▸ div.container

```
1  <div class="container">
2    <div class="row">
3      <div class="col-xs-12">
4        <h2>My Servers</h2>
5        <button class="btn btn-primary" (click)="onAddServer()">Add Server</button>
6        <br><br>
7        <ul class="list-group">
8          <li
9            class="list-group-item"
10           *ngFor="let server of servers; let i = index"
11           (click)="onRemoveServer(i)">{{ server }}</li>
12       </ul>
13     </div>
14   </div>
15 </div>
16
```

Create variable i and used it as parameter in onRemoveServer

example1-solution(Getting the index when using ngFor”

- Click “Add Server” 3 times.
- Click 2 row “Another Server”.



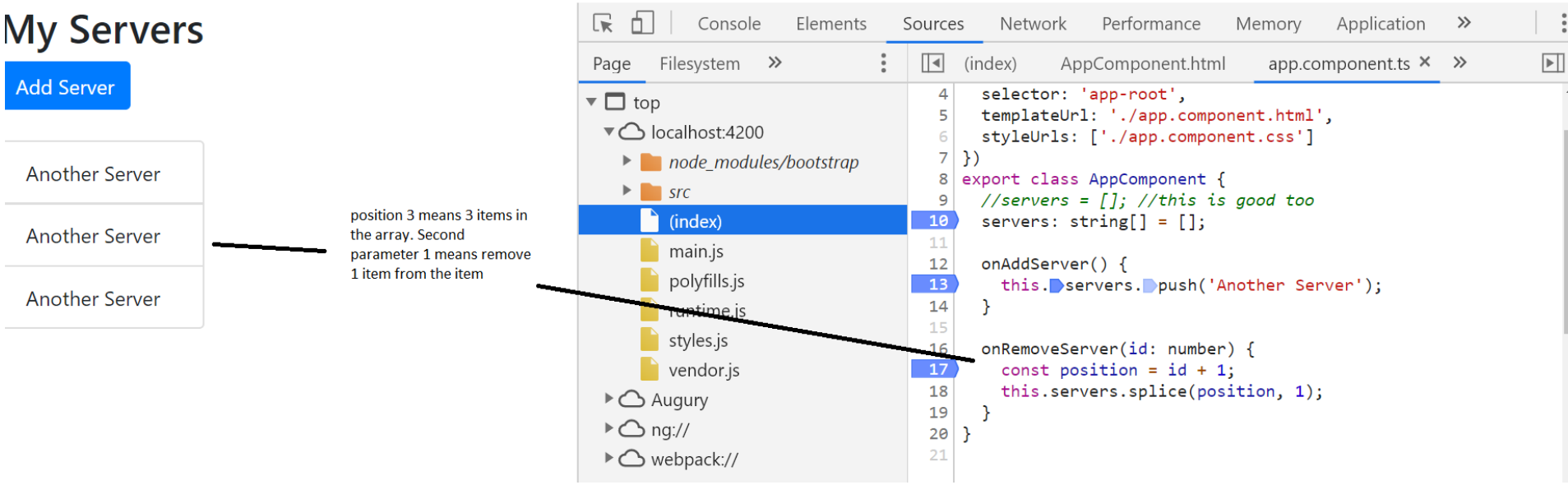
example1-solution(Getting the index when using ngFor”

- Click second row of “another server” and get id is 1.

```
onRemoveServer(id: number) {
  const position = id + 1;
  this.servers.splice(position, 1);
}
}
```

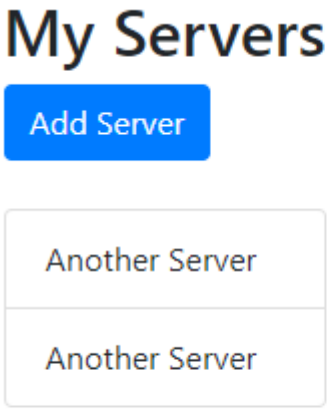
example1-solution(Getting the index when using ngFor”

- The splice() method adds/removes items to/from an array, and returns the removed items(s). Note, this method changes the original array.



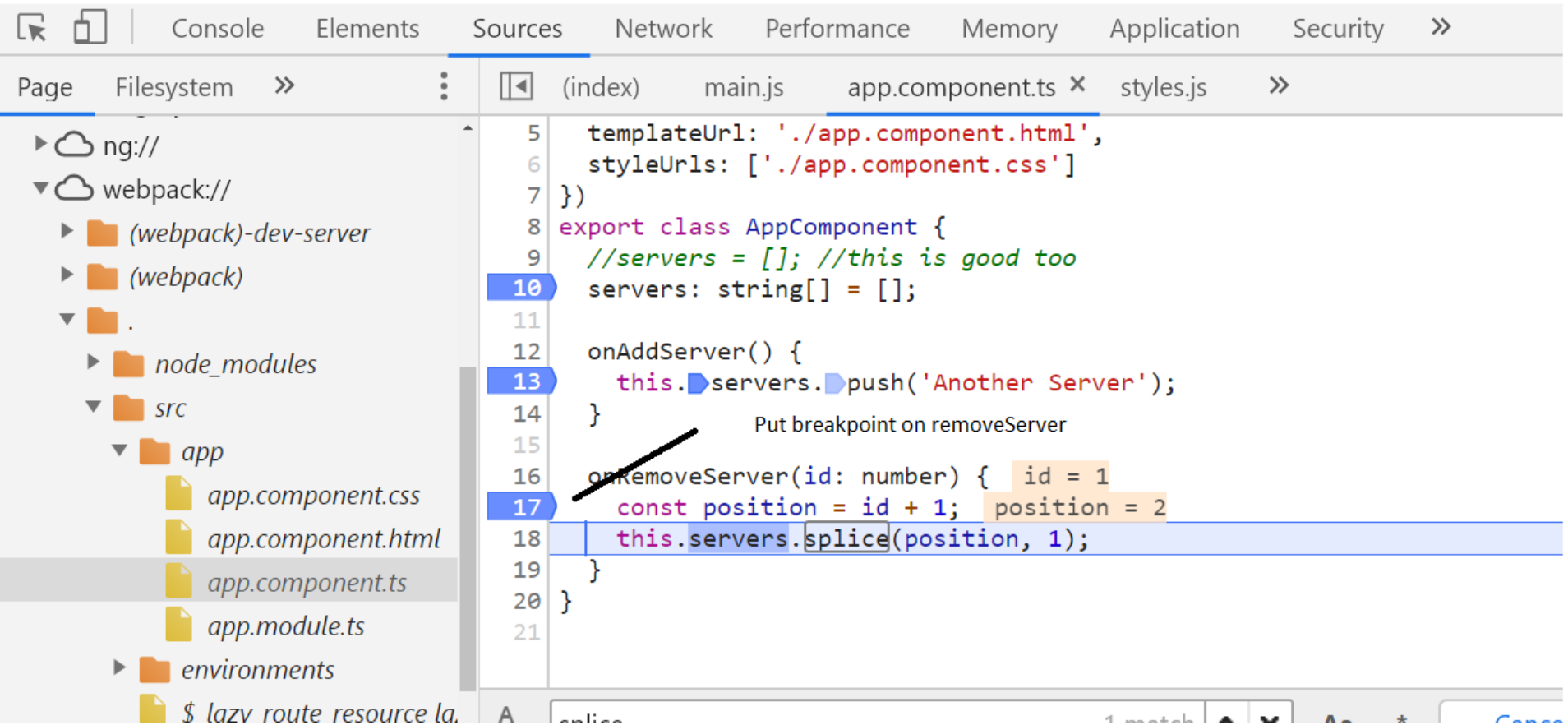
example2-solution(how to debug using browser)

- Copy C:\workspace-UT-Angular8-SEPT-2020 \session5\example1-solution\debugging to C:\workspace-UT-Angular8-SEPT-2020 \session5\example2-solution
- Click “Add Server” twice.
- When click on last “another server” it is not remove



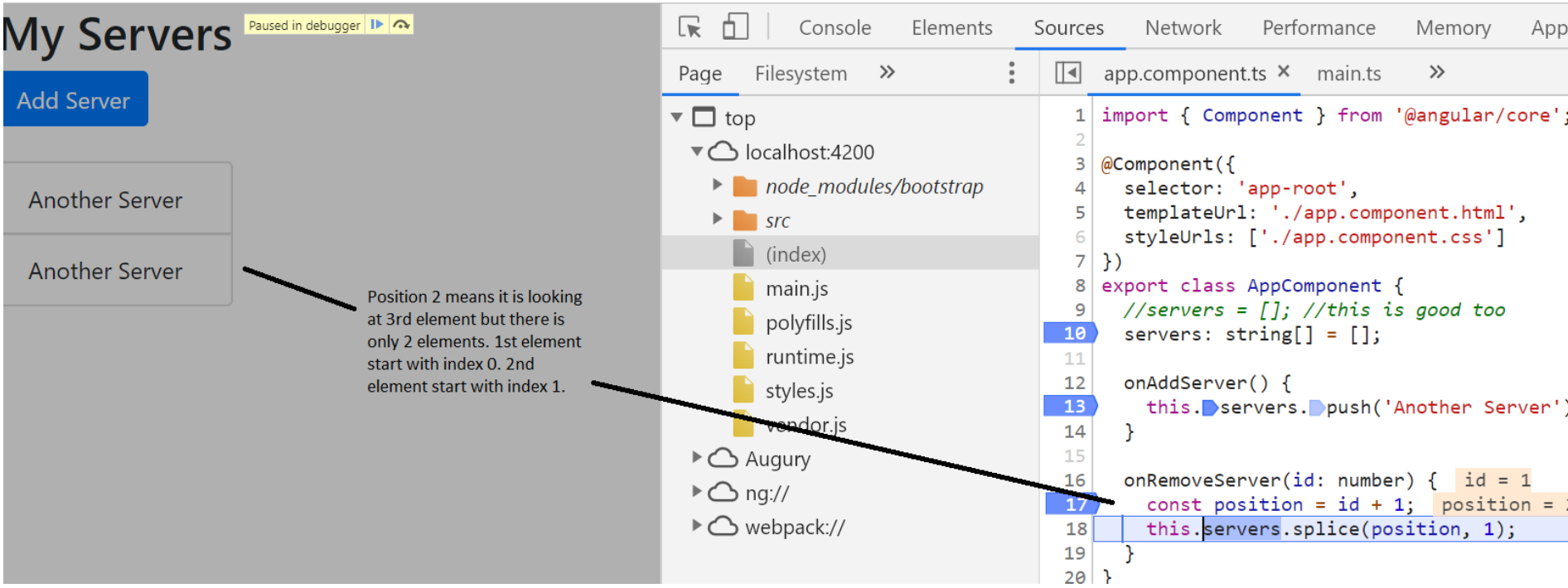
example2-solution(how to debug using browser)

- Click on “sources” tab.
- Click on left menu “WebPack/./Debugging/src/app/app.component.ts



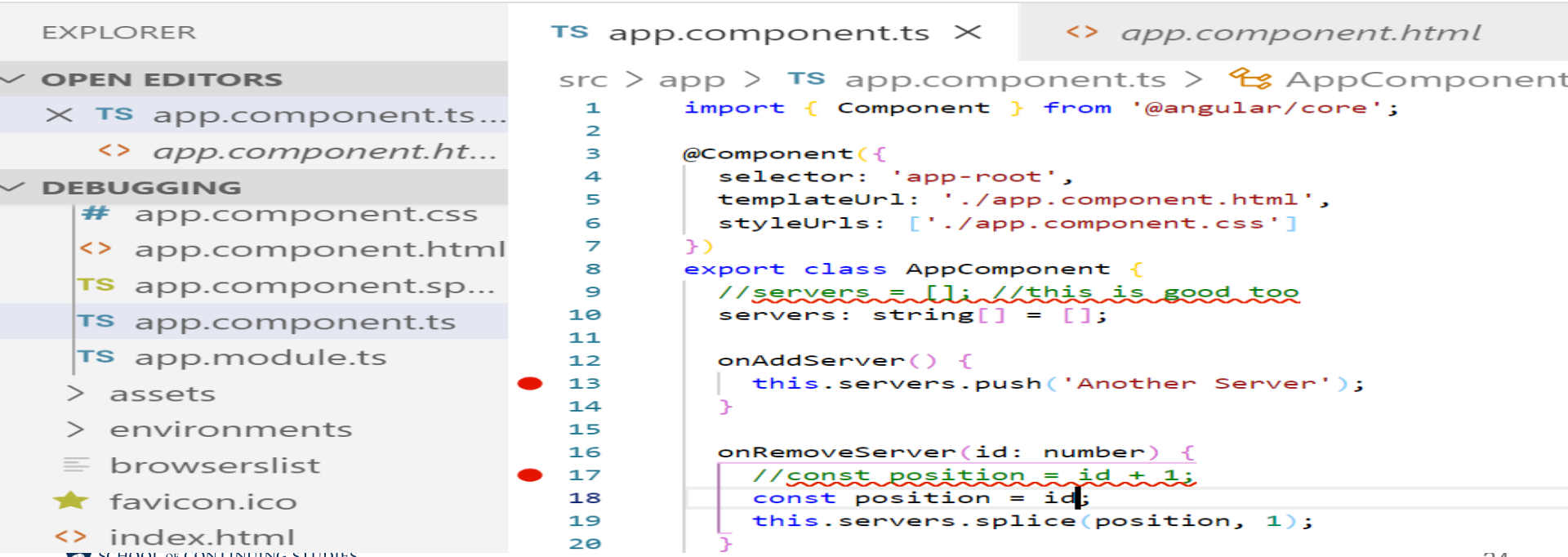
example2-solution(how to debug using browser)

- Click last “Another Server”. Nothing happens because the splice function cannot have position 2 which is 3rd element while the array have only 2 elements in the array.



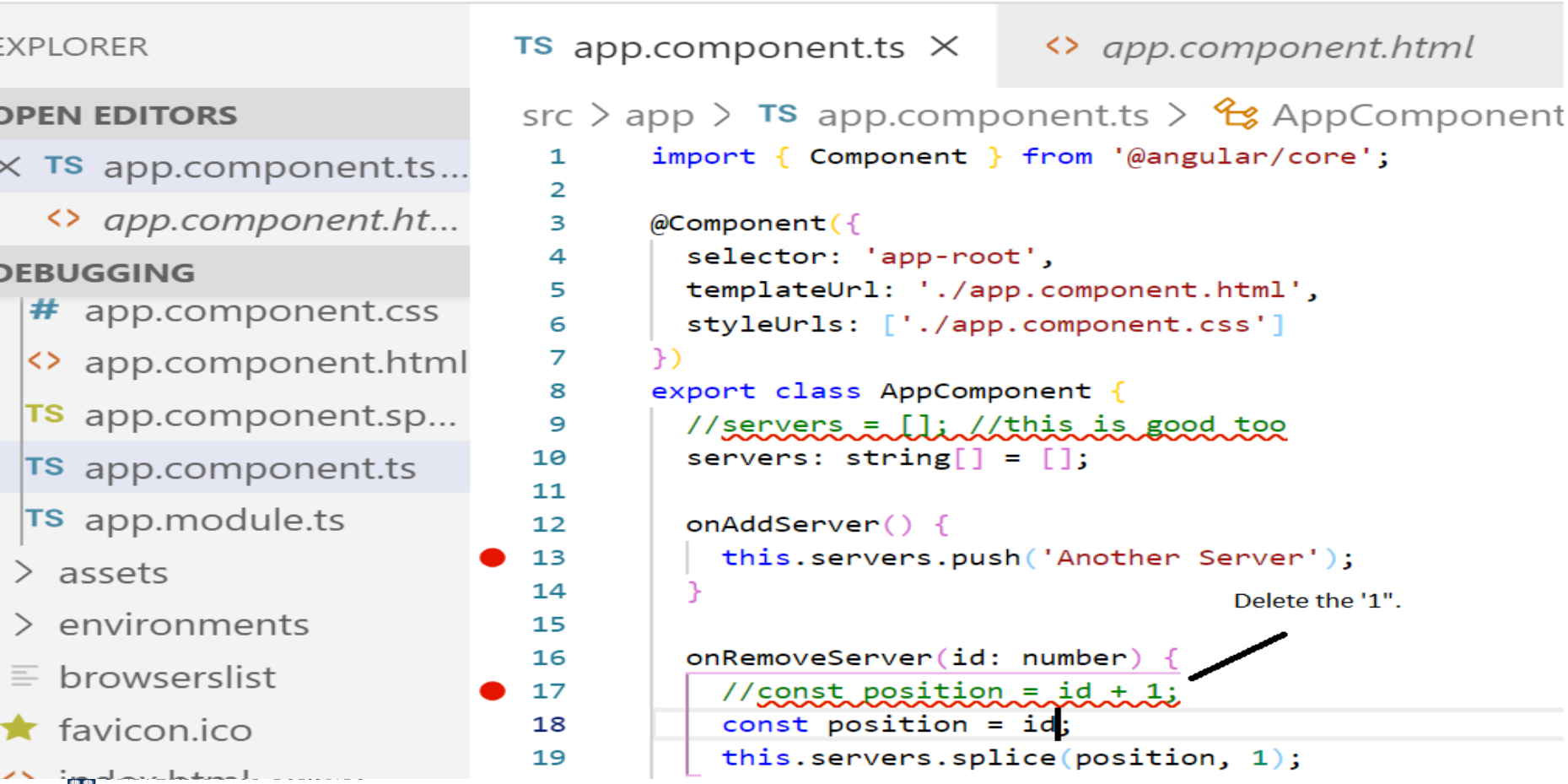
example2-solution(how to debug using browser)

- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example2-solution\debugging\src\app\app.component.ts



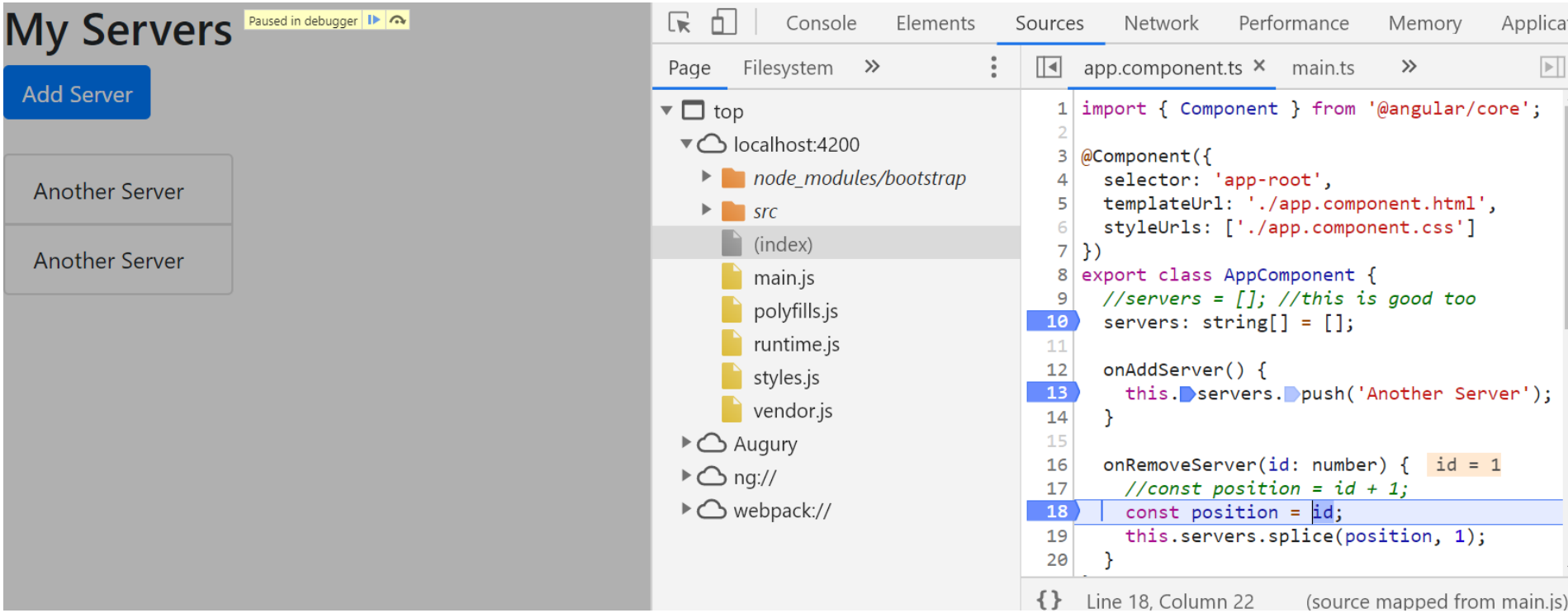
example2-solution(how to debug using browser)

- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example2-solution\debugging\src\app\app.component.ts

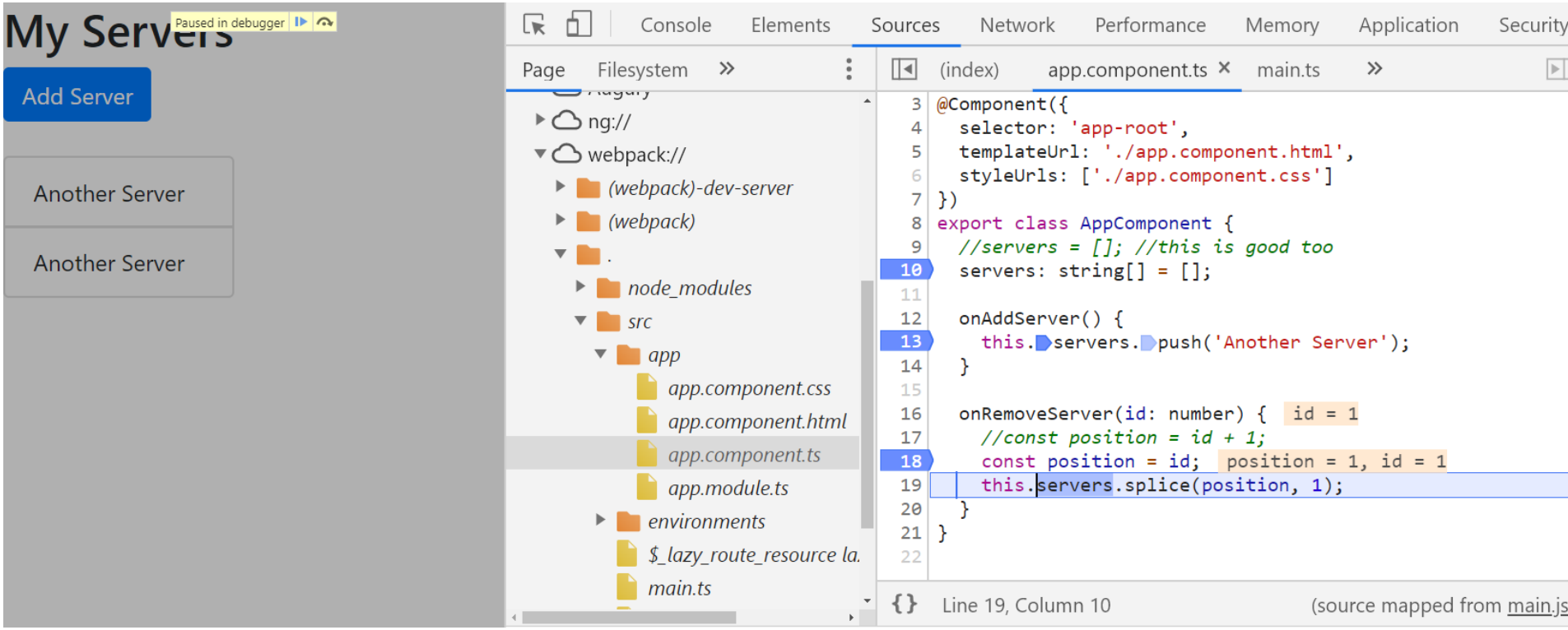


example2-solution(how to debug using browser)

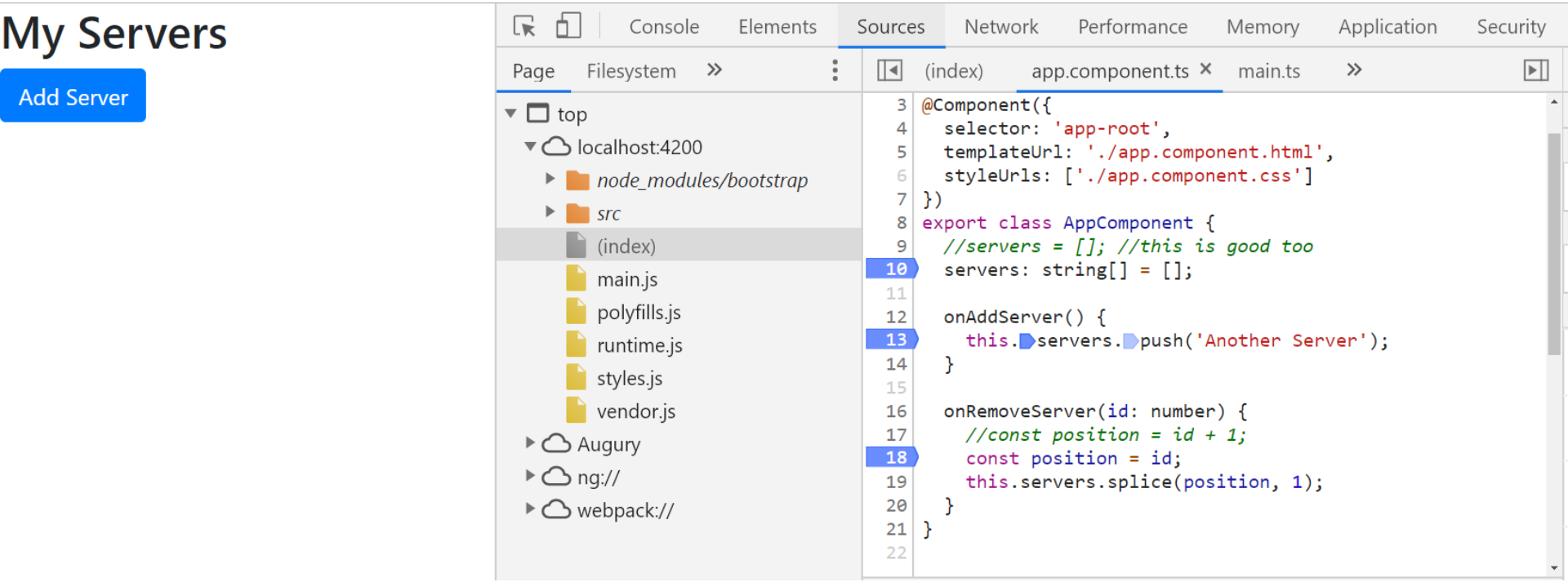
- After the code change, click “add server” twice to get two “Another Server” and than click the last “Another Server”.



example2-solution(how to debug using browser)

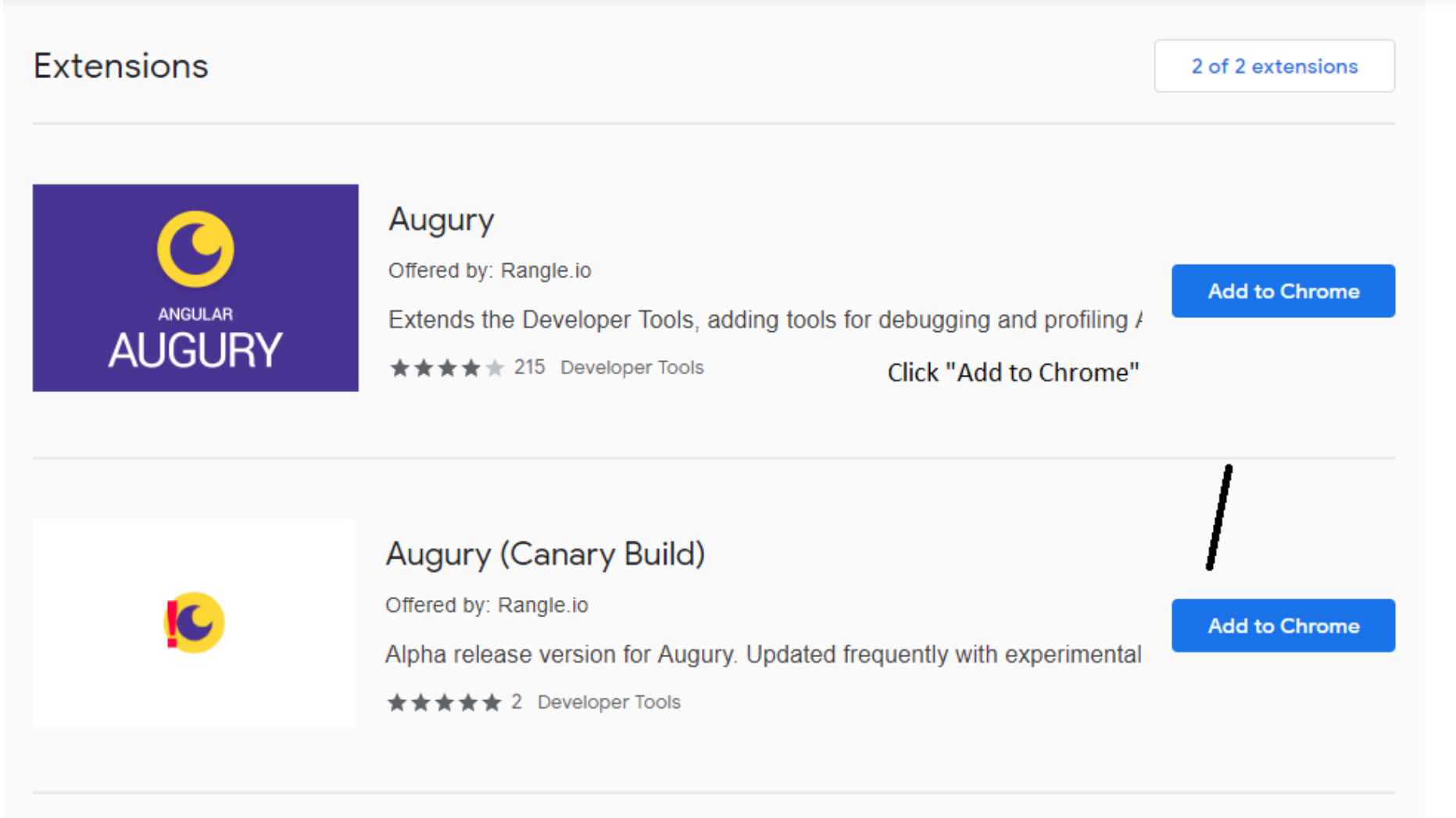


example2-solution(how to debug using browser)



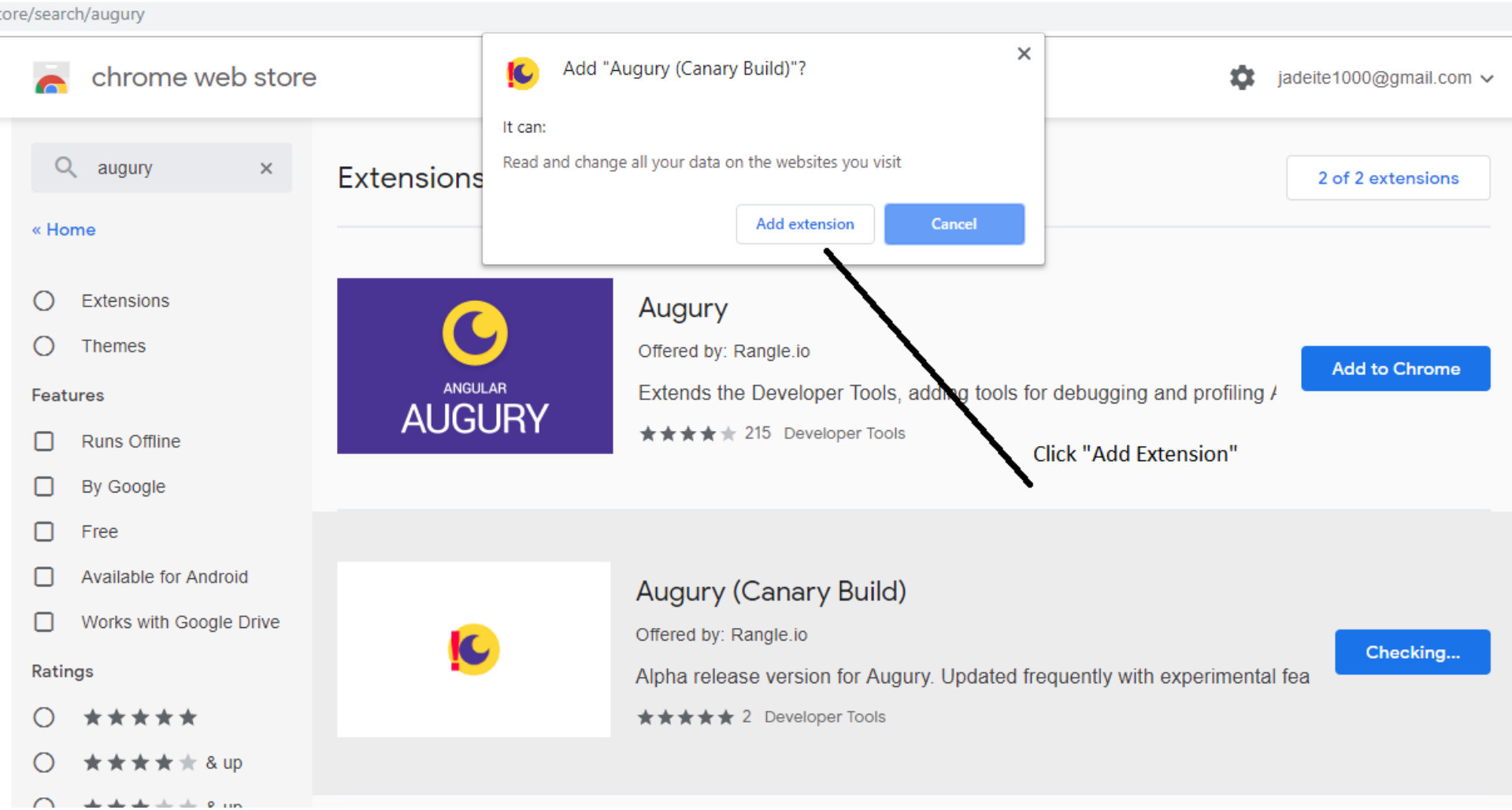
Augury

- https://chrome.google.com/webstore/search/augury



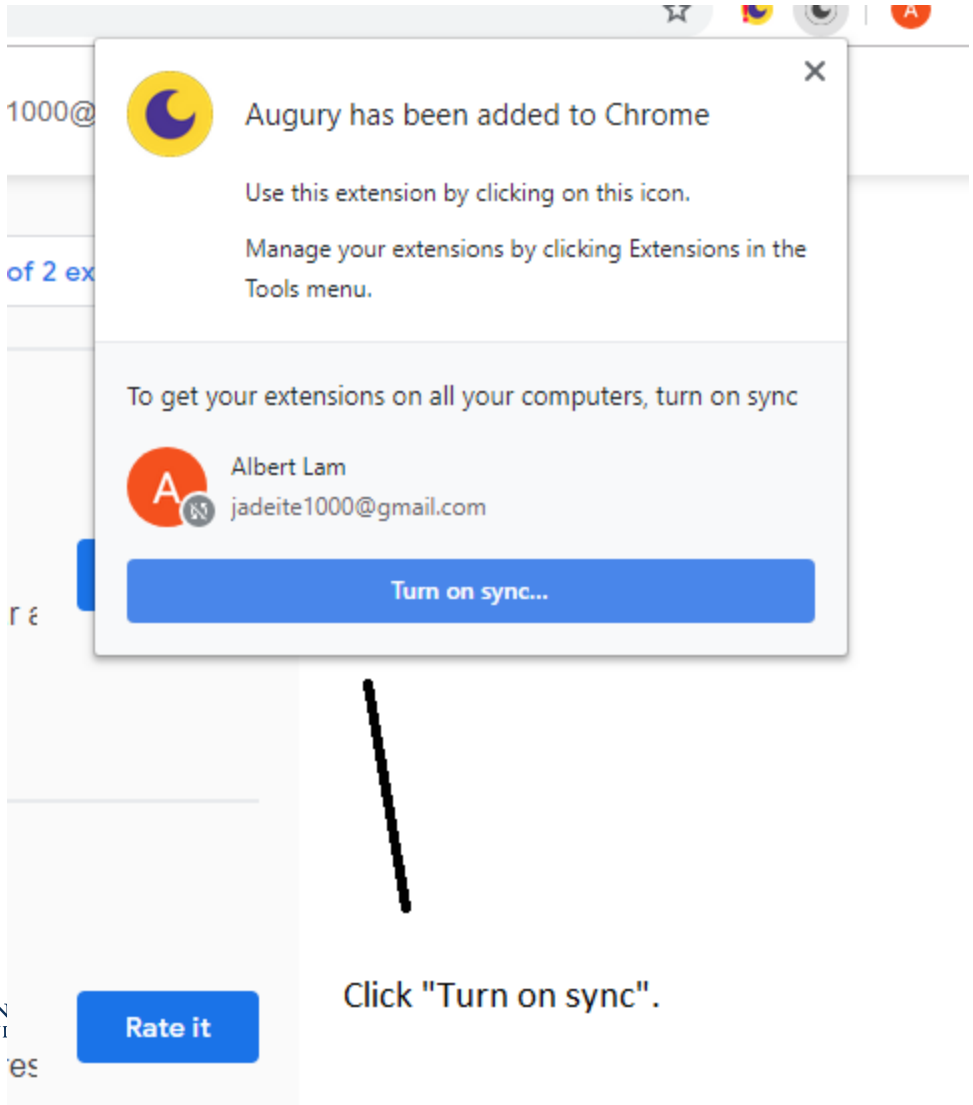
Augury

- https://chrome.google.com/webstore/search/augury



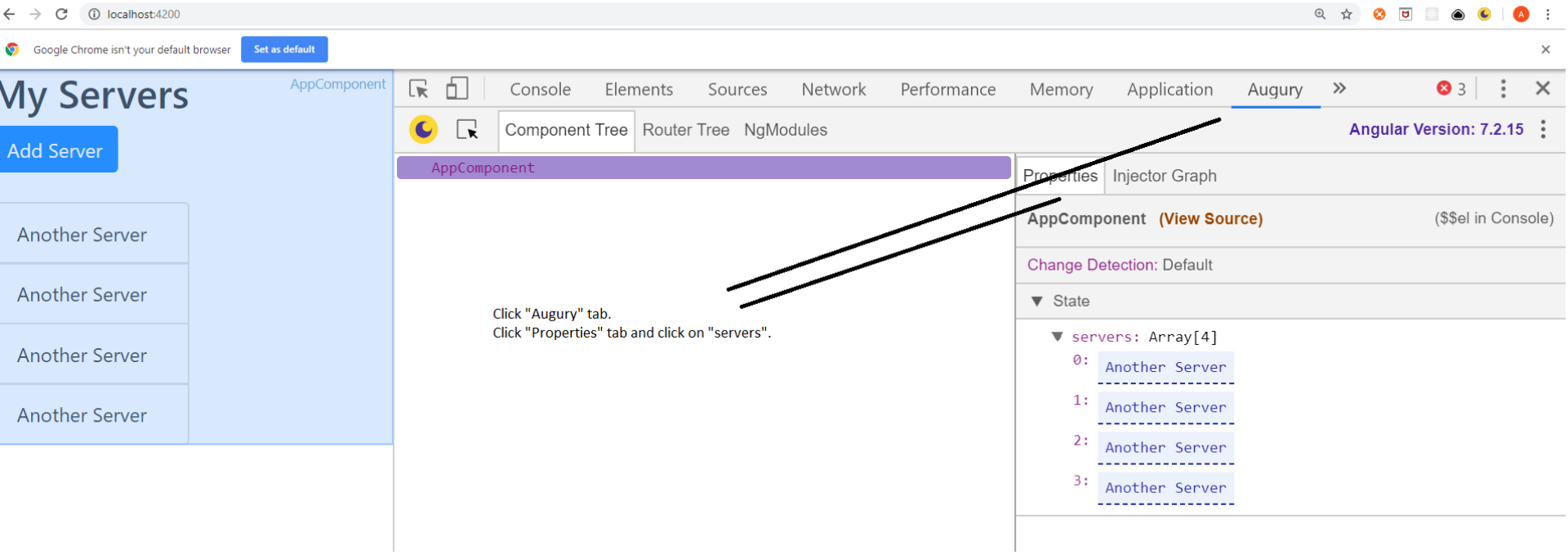
Augury

- https://chrome.google.com/webstore/search/augury



Augury

- https://chrome.google.com/webstore/search/augury
- Click “Add Server” 4 times.
- Right click and choose “inspect” and look for “Augury” tab.



Component Interaction(Pass data from parent to child with input binding).

- Copy C:\workspace-UT-Angular8-SEPT-2020 \ \Session5\code-before-npm-install\component-interaction to C:\workspace-UT-Angular8-SEPT-2020\Session5
- In dos-prompt(windows) or terminal(mac) go to directory C:\ workspace-UT-Angular8-SEPT-2020\Session5\component-interaction and type “npm install” in directory and than type “ng serve –o”.
- C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction

```
C:\workspace-UT-Angular8-JAN-2020\Session5\component-interaction>ng serve -o
Your global Angular CLI version (8.3.19) is greater than your local
```

Component Interaction(Pass data from parent to child with input binding).

- C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction\src\app\app.component.html

Component Communication Cookbook

Pass data from parent to child with input binding ("Heroes")
Intercept input property changes with a setter ("Master")
Intercept input property changes with *ngOnChanges* ("Source code version")
Parent listens for child event ("Colonize Universe")

Master controls 3 heroes

Dr IQ says:

I, Dr IQ, am at your service, Master.

Magneta says:

I, Magneta, am at your service, Master.

Bombasto says:

I, Bombasto, am at your service, Master.

Back to Top

Component Interaction(Pass data from parent to child with input binding).

- Look at the html flow:
  - Look at C:\AlbertLam\ Angular8-UT-Jan-2020 \Session5\Flow\ ComponentInteraction-PassData-From-Parent-To-Chld-BigPicture-Flow1.txt
- Input Decorator
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-PassData-From-Parent-To-Chld-Input-Decorator-Flow1.txt

Component Interaction(Pass data from parent to child with input binding).

- C:\workspace-UT-Angular8-SEPT-2020\Session5\component-interaction\src\app\app.component.html

File Edit Selection View Go Debug Terminal Help app.component.html - component-interaction - Visual Studio C...

EXPLORER

OPEN EDITORS
app.component.html src\app

COMPONENT-INTERACTION
e2e
node_modules
src
app
app.component.html
app.component.ts
app.module.ts
app.module.ts.bak
hero-child.component.ts
hero-parent.component.ts
hero.ts
name-child.component.ts
name-parent.component.ts

app.component.html

src > app > app.component.html > ...

```
1  <h1 id="top">Component Communication Cookbook</h1>
2
3  <a href="#parent-to-child">Pass data from parent to child with input binding ("Heroe
4  <a href="#parent-to-child-setter">Intercept input property changes with a setter ("M
5  <a href="#parent-to-child-on-changes">Intercept input property changes with <i>ngOnC
6  <a href="#child-to-parent">Parent listens for child event ("Colonize Universe")</a><
7
8  <div id="parent-to-child">
9    <app-hero-parent></app-hero-parent>
10 </div>
11 <a href="#top" class="to-top">Back to Top</a>
12
13 <hr>
14 <div id="parent-to-child-setter">
15   <app-name-parent></app-name-parent>
16 </div>
17 <a href="#top" class="to-top">Back to Top</a>
18
19 <hr>
20 <div id="parent-to-child-on-changes">
21   <app-version-parent></app-version-parent>
22 </div>
23 <a href="#top" class="to-top">Back to Top</a>
```

Component Interaction(Pass data from parent to child with input binding).

- The parent component is C:\workspace-UT-Angular8-SEPT-2020\Session5\component-interaction\src\app\hero-parent.component.ts

TS hero-parent.component.ts

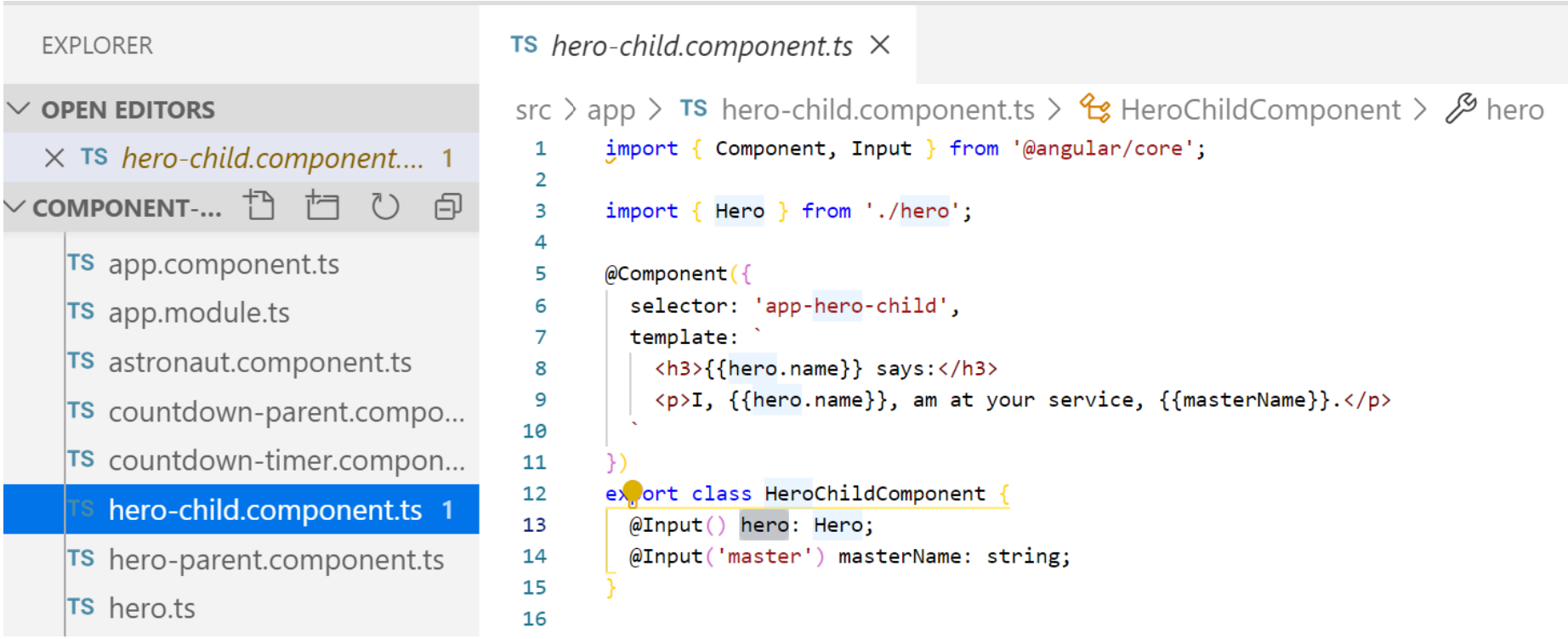
src > app > TS hero-parent.component.ts > HeroParentComponent

```
1   import { Component } from '@angular/core';
2
3   import { HEROES } from './hero';
4
5   @Component({
6     selector: 'app-hero-parent',
7     template: `
8       <h2>{{master}} controls {{heroes.length}} heroes</h2>
9       <app-hero-child *ngFor="let hero of heroes"
10        [hero]="hero"
11        [master]="master">
12      </app-hero-child>
13    `
14  })
15  export class HeroParentComponent {
16    heroes = HEROES;
17    master = 'Master';
18  }
19
```

Component Interaction(Pass data from parent to child with input binding).

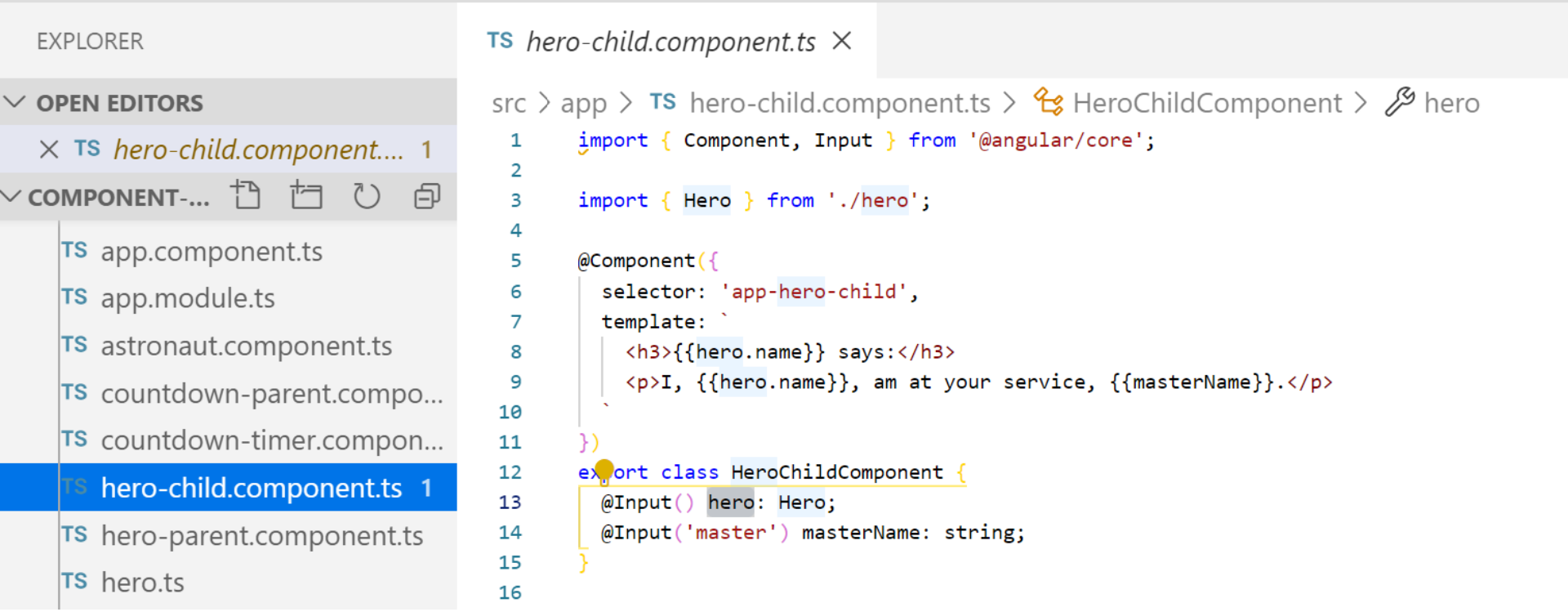
- The child component is C:\workspace-UT-Angular8-SEPT-2020\Session5\component-interaction\src\app\hero-child.component.ts
- HeroChildComponent has two input properties,typically adorned with @Input decorations.
- The second @Input aliases the child component property name masterName as ‘master’.
- The HeroParentComponent nests the child HeroChildComponent inside an *ngFor repeater, binding its master string property to the child’s master alias, and each iteration’s hero instance to the child’s hero property.

Component Interaction(Pass data from parent to child with input binding).



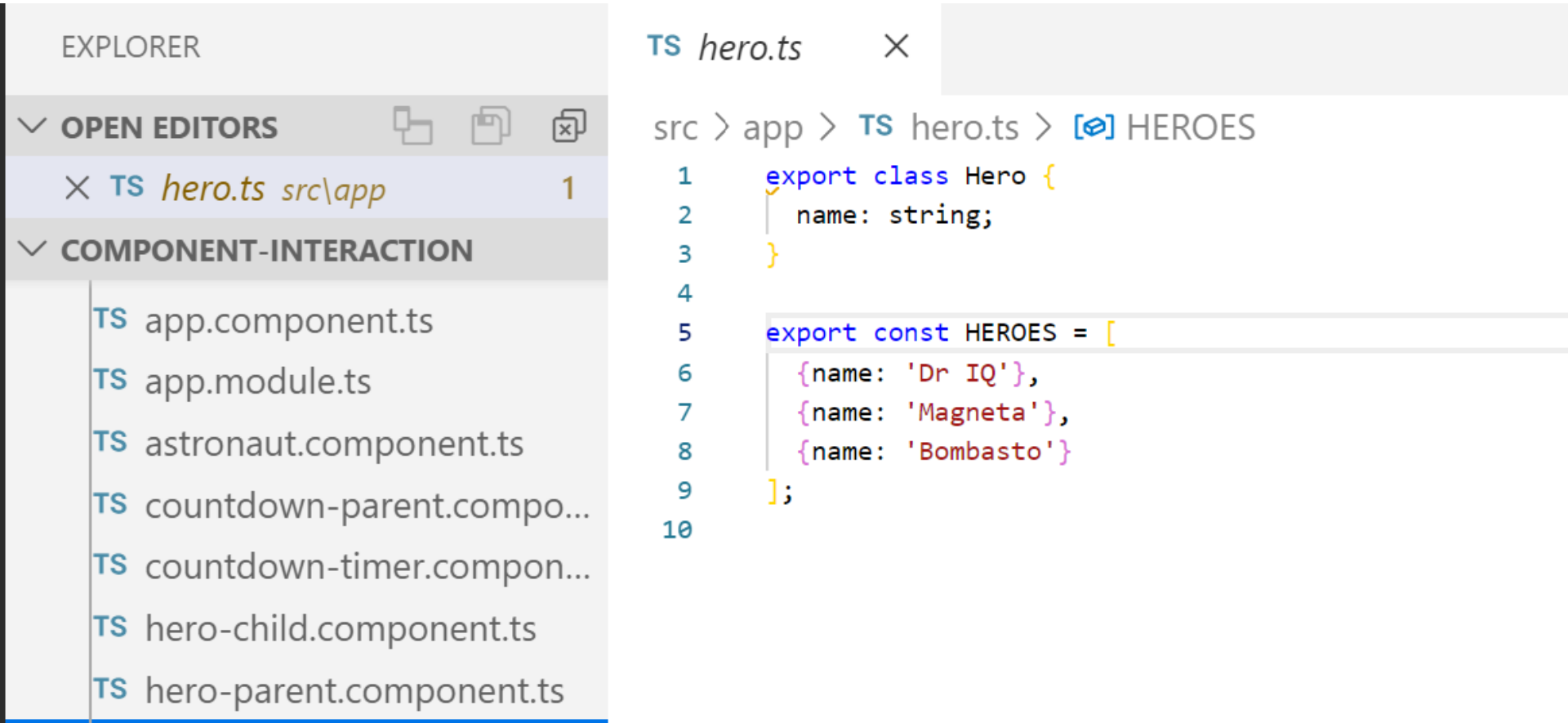
Component Interaction(Pass data from parent to child with input binding).

- The child component is C:\workspace-UT-Angular8-SEPT-2020\Session5\component-interaction\src\app\hero-child.component.ts



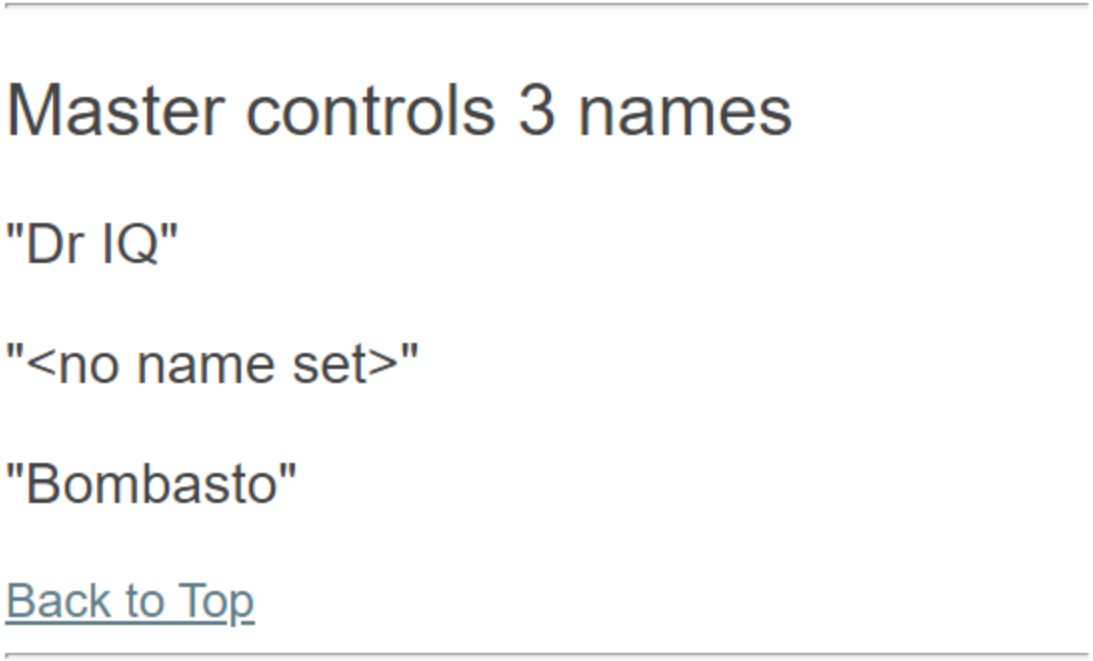
Component Interaction(Pass data from parent to child with input binding).

- The const array C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction\src\app\hero.ts



Component Interaction(Intercept input property changes with a setter).

- Use an input property setter to intercept and act upon a value from the parent. The setter of the name input property in the child NameChildComponent trims the whitespace from a name and replaces an empty value with default text.



Component Interaction(Intercept input property changes with a setter).

- Look at the html flow:
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-InterceptInputProperty--BigPicture-Flow1.txt

Component Interaction(Intercept input property changes with a setter).

- C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction\src\app\name-parent.component.ts

TS name-parent.component.ts

src > app > TS name-parent.component.ts > NameParentComponent

```
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-name-parent',
5     template: `
6     <h2>Master controls {{names.length}} names</h2>
7     <app-name-child *ngFor="let name of names" [name]="name"></app-name-child>
8     `
9   })
10  export class NameParentComponent {
11    // Displays 'Dr IQ', '<no name set>', 'Bombasto'
12    names = ['Dr IQ', '   ', '  Bombasto  '];
13  }
14
```

Component Interaction(Intercept input property changes with a setter).

- C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction\src\app\name-child.component.ts

TS name-child.component.ts

src > app > TS name-child.component.ts > ...

```
1   import { Component, Input } from '@angular/core';
2
3   @Component({
4     selector: 'app-name-child',
5     template: '<h3>"{{name}}"</h3>'
6   })
7   export class NameChildComponent {
8     private _name = '';
9
10    @Input()
11    set name(name: string) {
12      this._name = (name && name.trim()) || '<no name set>';
13    }
14
15    get name(): string
16    {
17      return this._name;
18    }
19  }
20
```

Component Interaction(Intercept input property changes with ngOnChanges()).

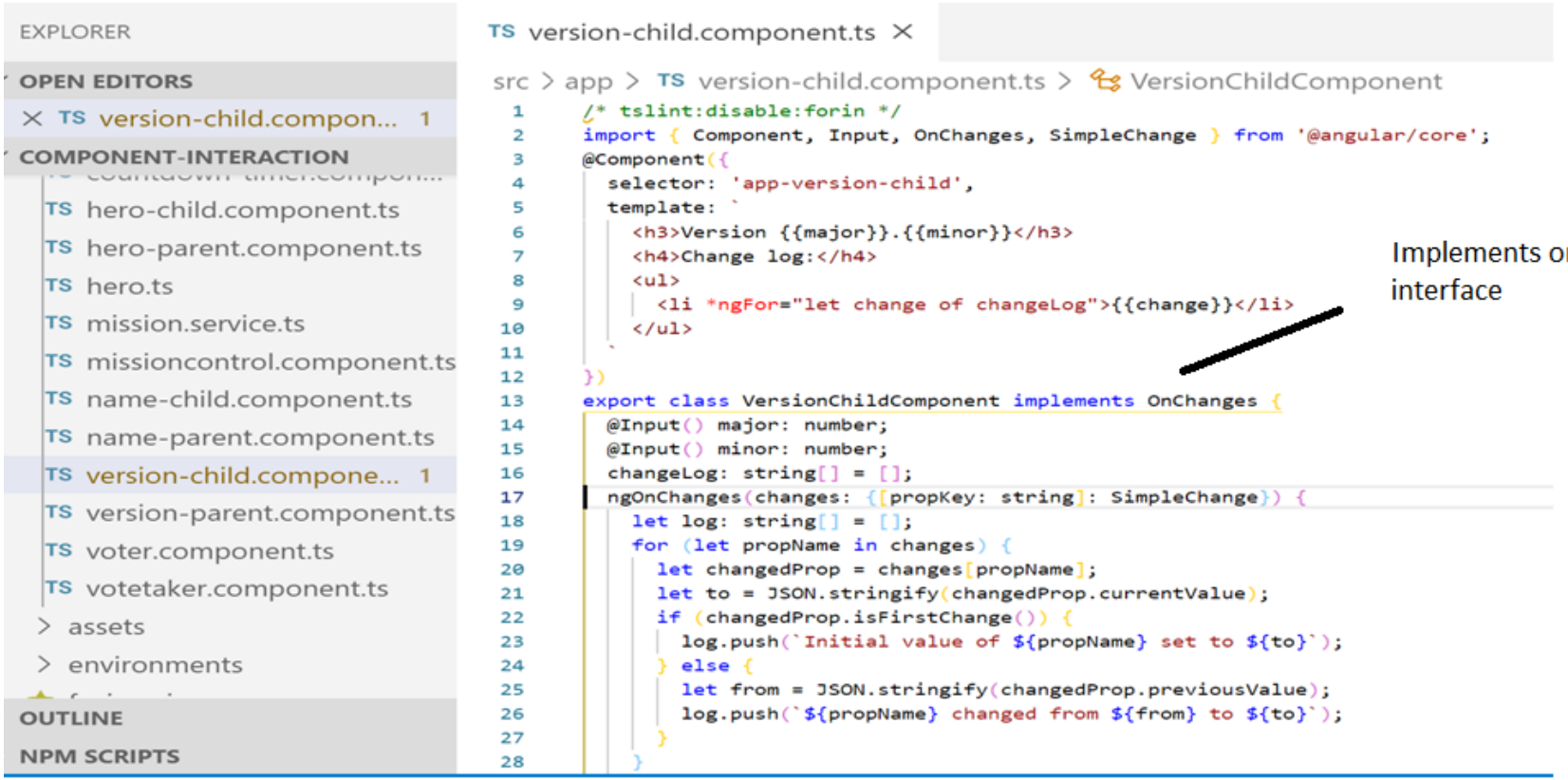
- Detect and act upon changes to **input property values** with the ngOnChanges(0 method of the OnChanges lifecycle hook interface.
- You may prefer this approach to the property setter when watching multiple, interacting input properties.

Component Interaction(Intercept input property changes with ngOnChanges()).

- Look at the html flow:
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\ComponentInteraction-NgOnChanges--BigPicture-Flow1.txt
- Click “Major”.
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\ ComponentInteraction-NgOnChanges--ClickMajor--Flow1.txt
- Click first “Minor”.
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\ ComponentInteraction-NgOnChanges--ClickMinor--Flow1.txt

Component Interaction(Intercept input property changes with ngOnChanges()).

- This VersionChildComponent detects changes to the major and minor input properties and composes a log message reporting these changes. Place breakpoints in ngOnChanges method

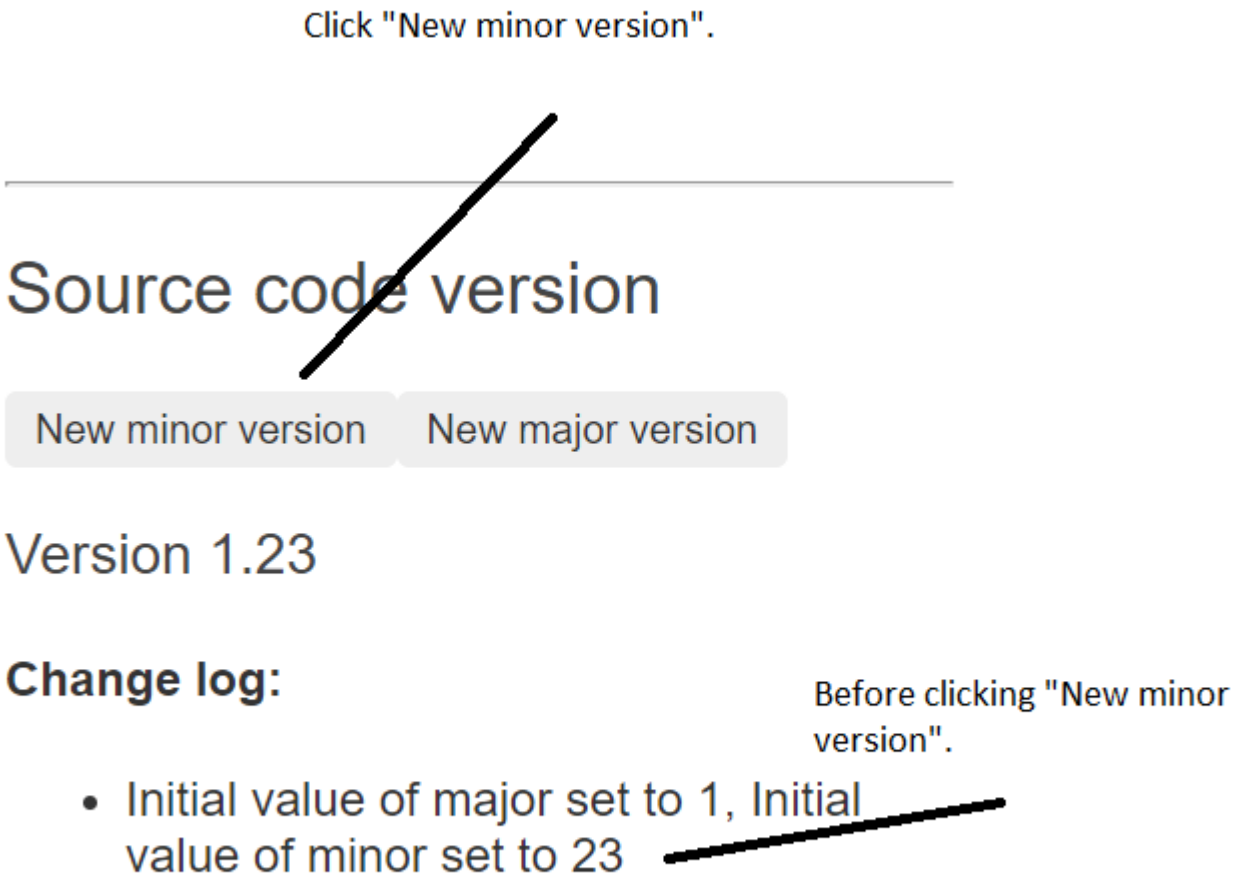


Component Interaction(Intercept input property changes with ngOnChanges()).

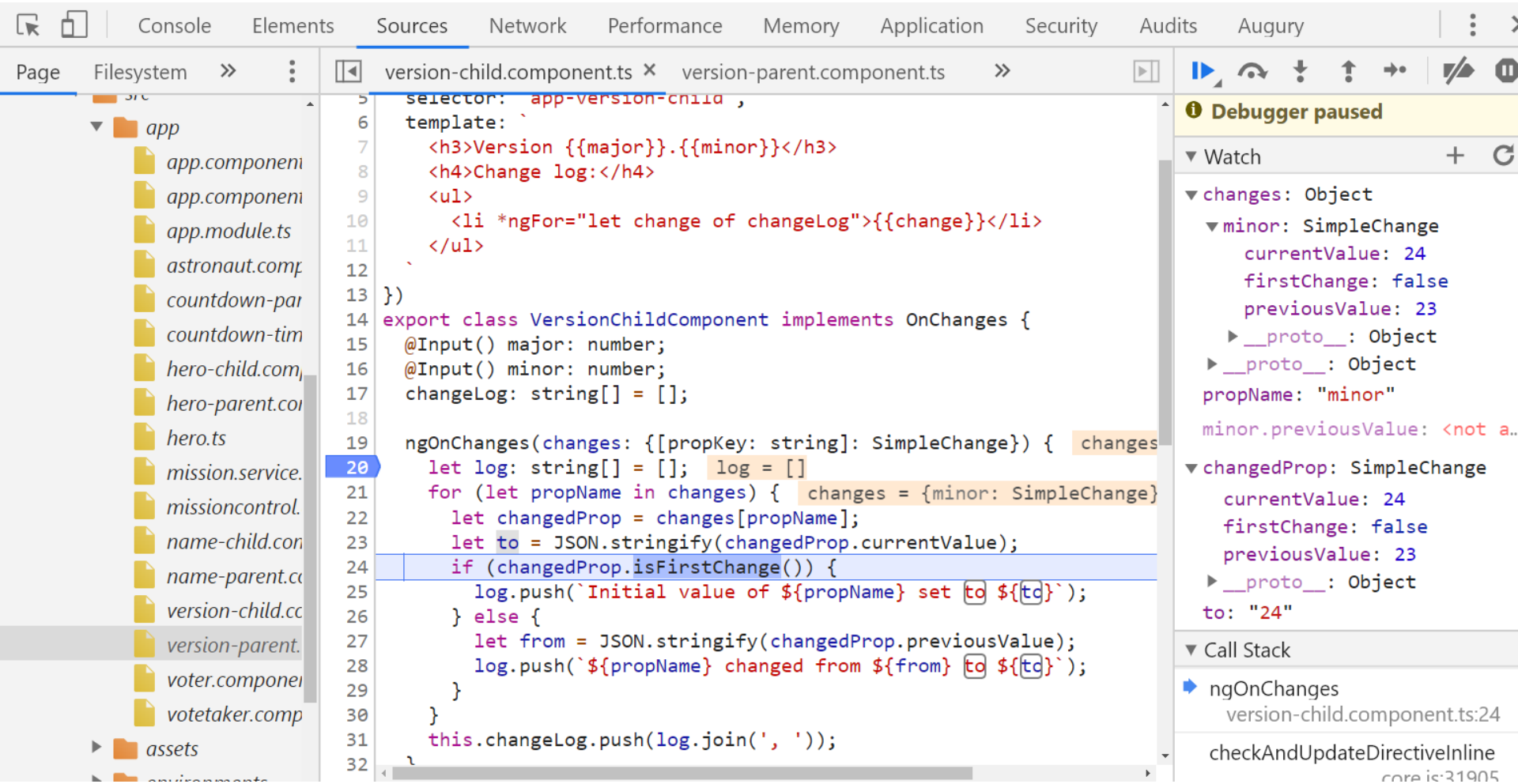
- The VersionParentComponent supplies the minor and major values and binds buttons to methods that change them.

Component Interaction(Intercept input property changes with ngOnChanges()).

- The VersionParentComponent supplies the minor and major values and binds buttons to methods that change them.
- Click “New Minor Version”. See flow\ ComponentInteraction-Intercept input property changes with ngOnChanges-ClickNewMinorVersion-FlowVersion1.txt



Component Interaction(Intercept input property changes with ngOnChanges()).



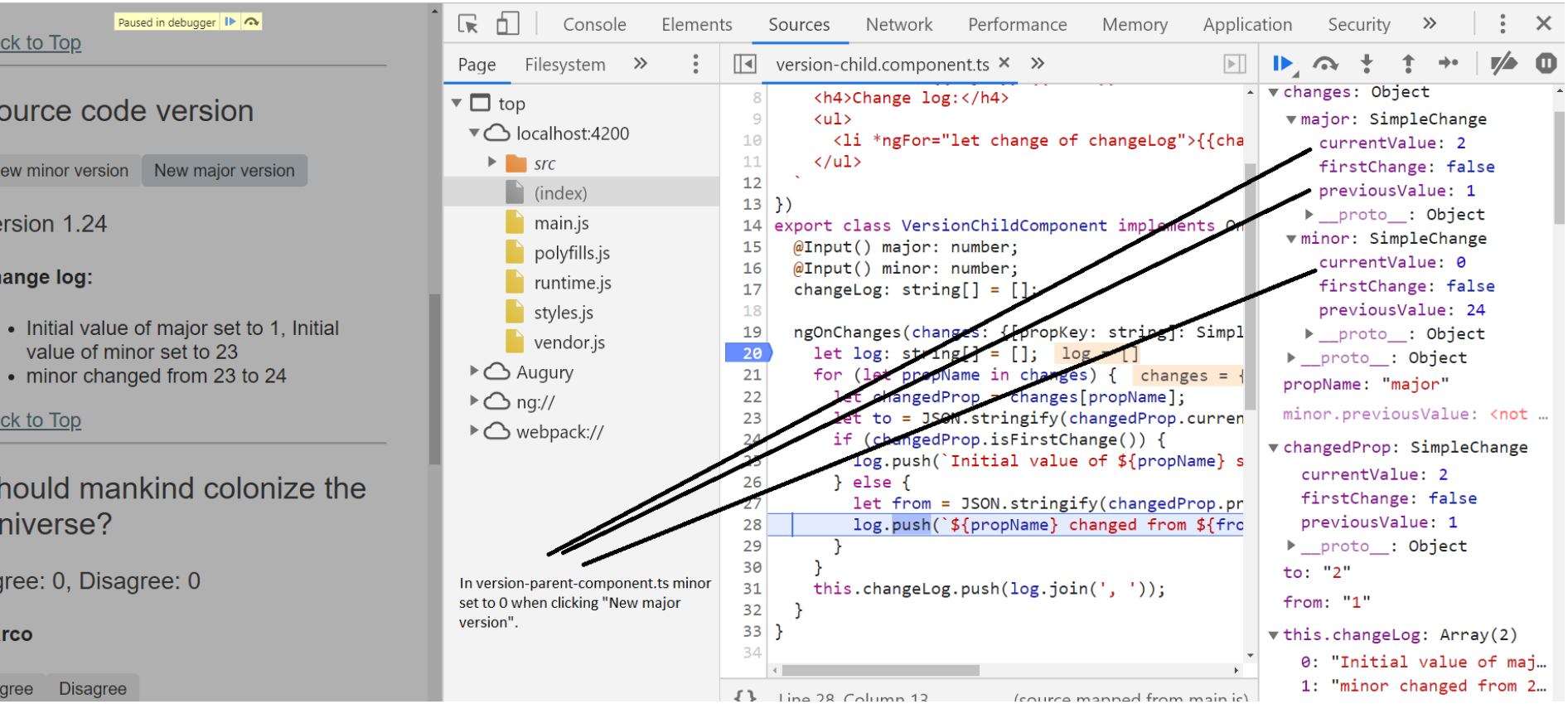
Component Interaction(Intercept input property changes with ngOnChanges()).

Component Interaction(Intercept input property changes with ngOnChanges()).

- Click “New Major Version”

Component Interaction(Intercept input property changes with ngOnChanges()).

- Click “New Major Version”.
- Looked at Flows\ComponentInteraction-Intercept input property changes with ngOnChanges-ClickNewMajorVersion-FlowVersion1.txt



Component Interaction(Intercept input property changes with ngOnChanges()).

- After clicking “New Major Version”:

Component Interaction(Parent listens for child event).

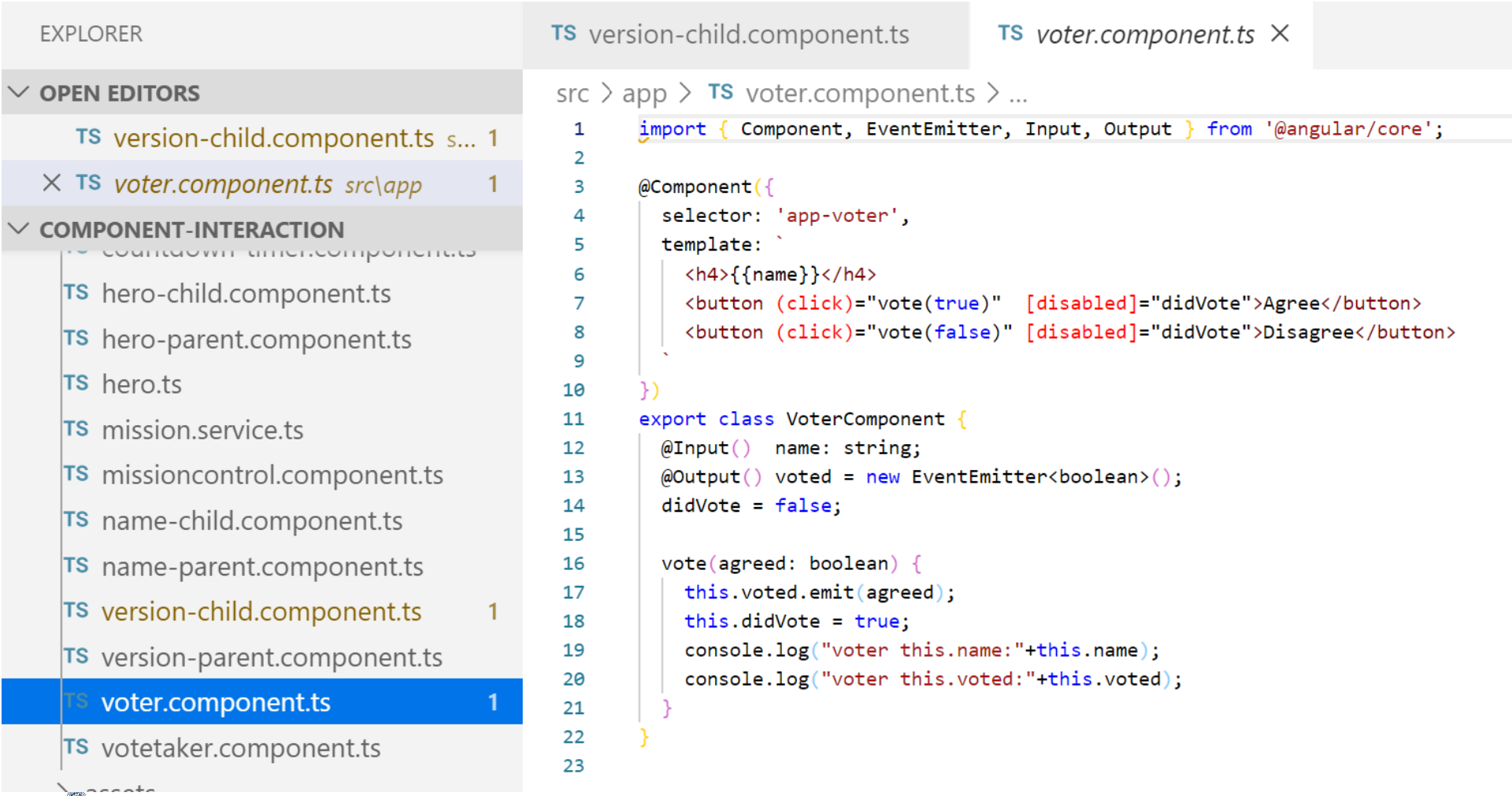
- The child component exposes an EventEmitter property with which it emits when something happens. The parent binds to that event property and reacts to those events.
- The child's EventEmitter property is an **output property**, typically adorned with an **@Output decoration** as seen in the VoterComponent.

Component Interaction(Parent listens for child event).

- Look at the html flow:
  - Look at C:\AlbertLam\ Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-Parent-Listen-ChildEvent--BigPicture-Flow1.txt
- Flow for Parent listens for child
  - Open file C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow \ComponentInteraction-ParentListensForChildEvent-FlowVersion1.txt

Component Interaction(Parent listens for child event).

- The child is voter.component.ts

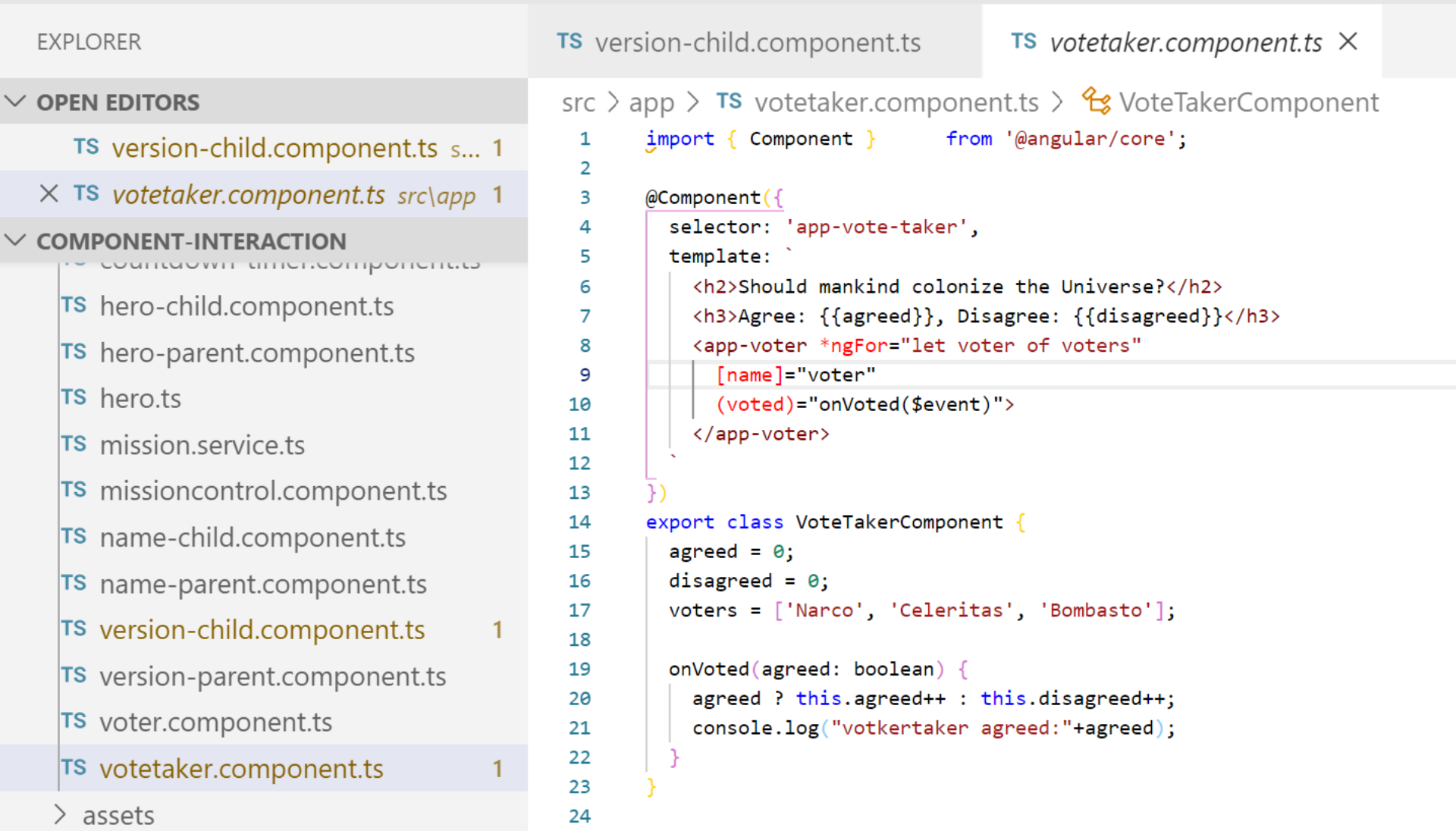


Component Interaction(Parent listens for child event).

- Clicking a button triggers emission of a true or false, the Boolean payload.
- The parent VoteTakerComponent binds an event handler called onVoted() that responds to the child event payload $event and updates a counter.
- The framework passes the event argument-represented by $event – to the handler method, and the method processes it.

Component Interaction(Parent listens for child event).

- The parent is votertaker.component.ts

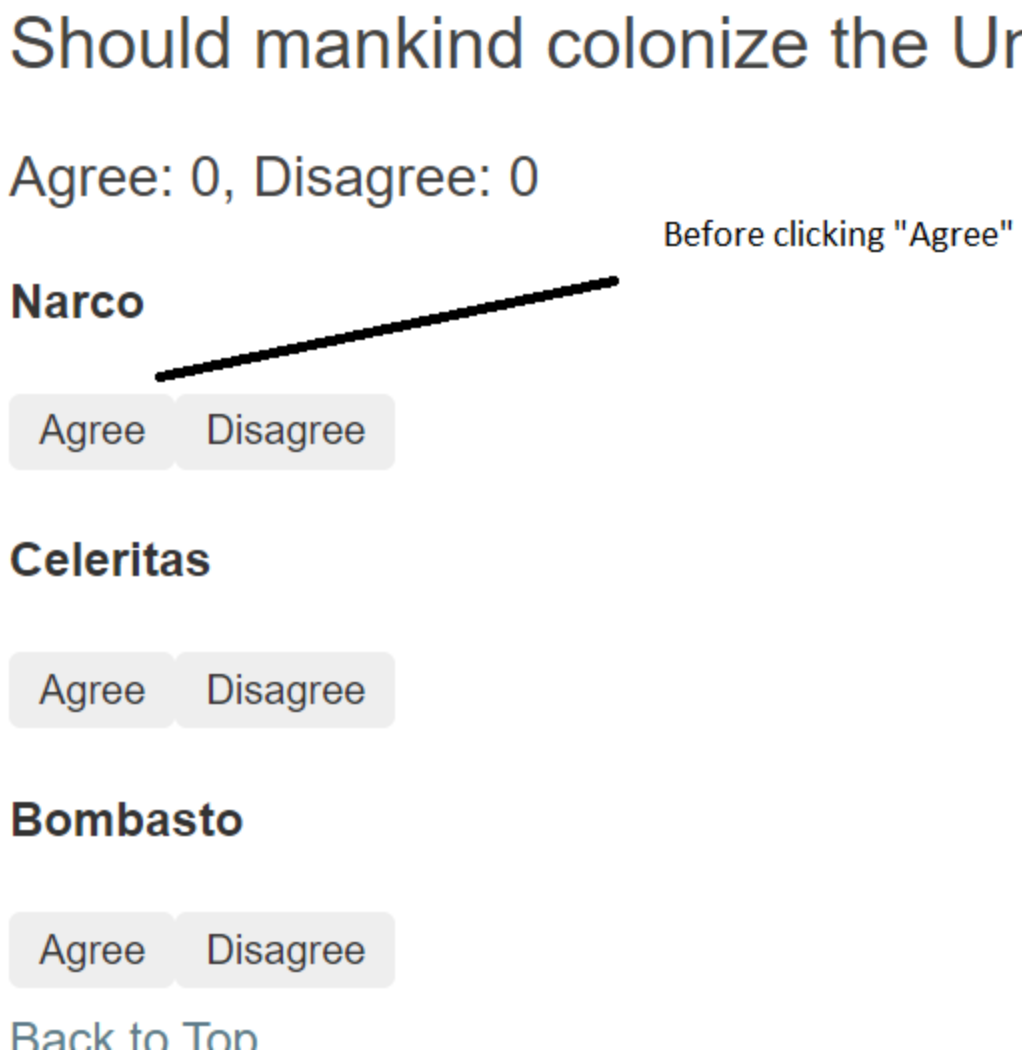


Component Interaction(Parent listens for child event).

- The app.component.html:

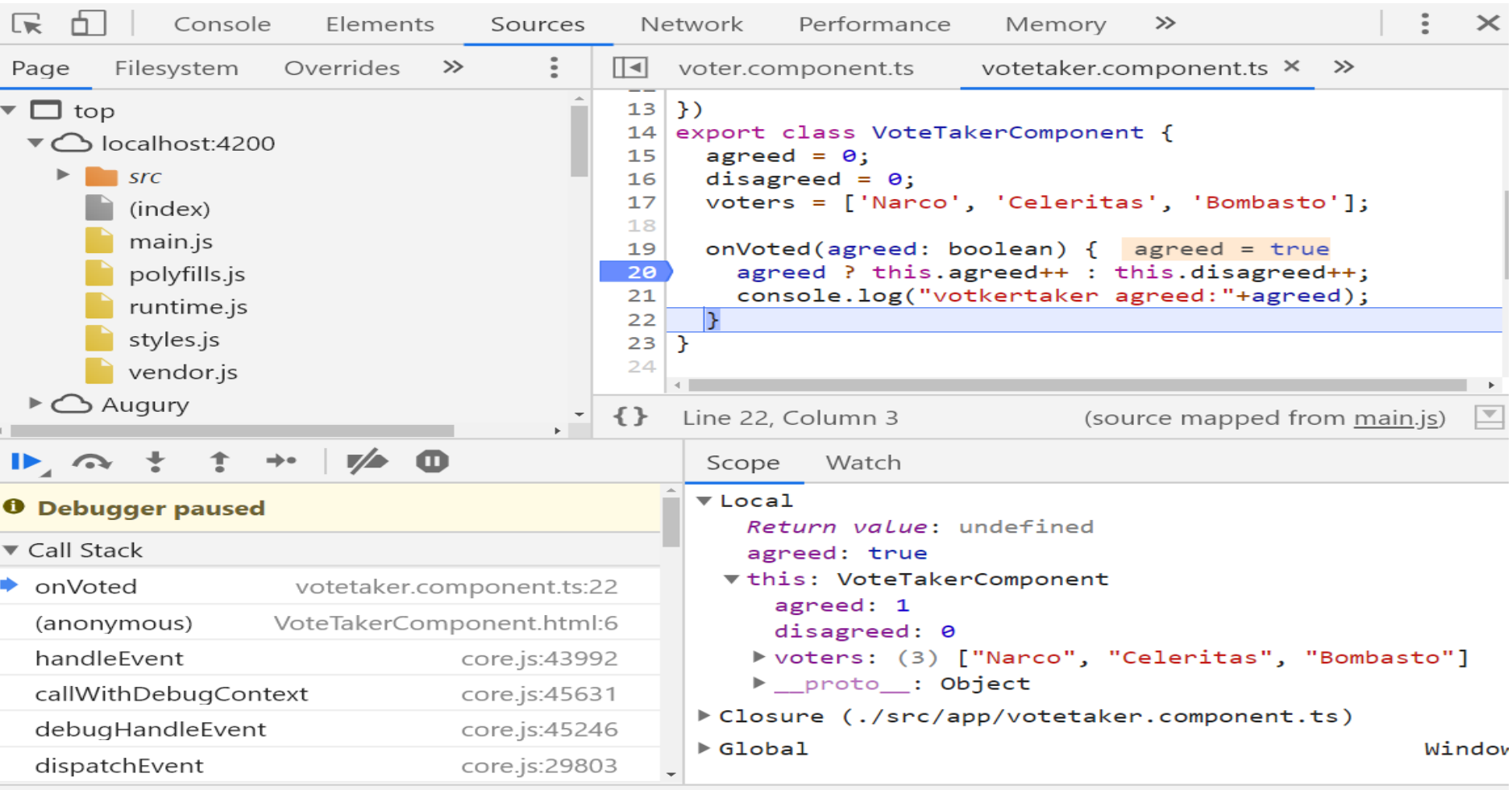
```
27
28      <hr>
29      <div id="child-to-parent">
30        <app-vote-taker></app-vote-taker>
31      </div>
32      <a href="#top" class="to-top">Back to Top</a>
33      <hr>
34
```

Component Interaction(Parent listens for child event).



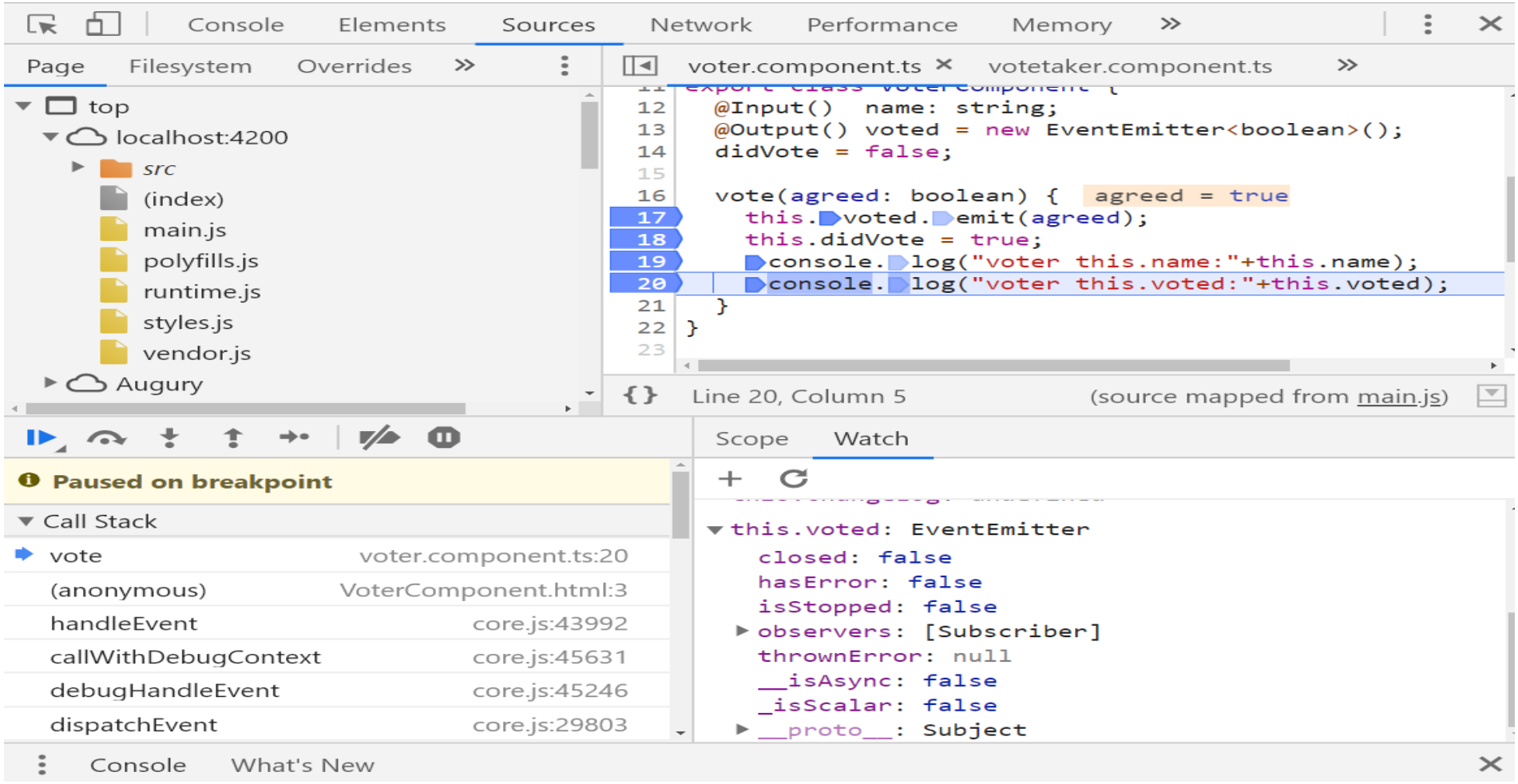
Component Interaction(Parent listens for child event).

- Parent Component votetaker.component.ts



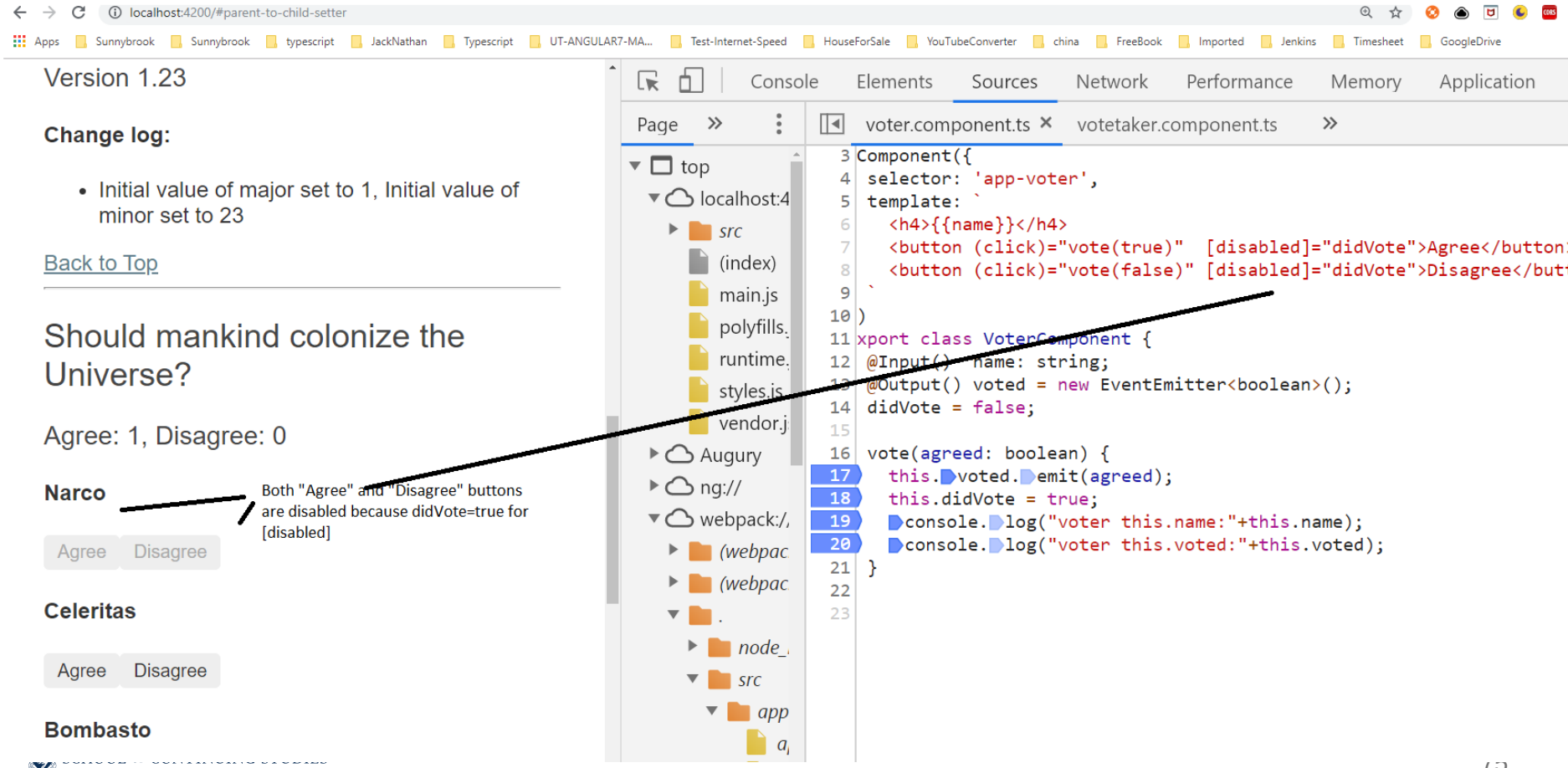
Component Interaction(Parent listens for child event).

- Child Component voter.component.ts



Component Interaction(Parent listens for child event).

- After clicking “Agree”.



Component Interaction(Parent interacts with child via local variable).

- A parent component cannot use data binding to read child properties or invoke child methods. You can do both by creating a template reference variable for the child element and then reference that variable within the parent template as seen in the following example.
- In C:\workspace-UT-Angular8-SEPT-2020 \Session5\component-interaction\src\app\app.component.html

In C:\workspace-UT-Angular7-Sept-2019\Session5\component-interaction\src\app \app.component.html

```
<div id="parent-to-child-local-var">
  <app-countdown-parent-lv></app-countdown-parent-lv>
</div>
<a href="#top" class="to-top">Back to Top</a>
<hr>
```

Component Interaction(Parent interacts with child via local variable).

- The parent component cannot data bind to the child's start and stop methods nor to its seconds property.
- You can place a local variable, #timer, on the tag<countdown-timer> representing the child component. That gives you a reference to the child component and the ability to access any of its properties or methods from within the parent template.

Component Interaction(Parent interacts with child via local variable).

- Look at the html flow:
  - Look at C:\AlbertLam\ Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-Parent-InteractWithChild-LocalVariable--BigPicture-Flow1.txt
- Look at C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\ComponentInteraction-Parent-Interacts-With-Child-Via-Local-Variable-FirstLoad-FlowVersion1.tx
- Look at C:\AlbertLam\ Angular8-UT-Jan-2020 \Session5\Flow\ComponentInteraction-Parent-Interacts-With-Child-Via-Local-Variable-ClickStart--FlowVersion1.txt

Component Interaction(Parent interacts with child via local variable).

Local variable, #timer

In parent component C:\workspace-UT-Angular7-Sept-2019\Session5\component-interaction\src\app\countdown-parent.component.ts

You can place a local variable, #timer, on the tag <countdown-timer> representing the child component. That gives you a reference to the child component and the ability to access any of its properties or methods from within the parent template

```
6   //// Local variable, #timer
7   @Component({
8     selector: '',
9     template: `
10    <h3>Countdown to Liftoff (via local variable)</h3>
11    <button (click)="timer.start()">Start</button>
12    <button (click)="timer.stop()">Stop</button>
13    <div class="seconds">{{timer.seconds}}</div>
14    <app-countdown-timer #timer></app-countdown-timer>
15    `,
16    styleUrls: ['../assets/demo.css']
17  })
18  export class CountdownLocalVarParentComponent { }
19
```

Component Interaction(Parent interacts with child via local variable).

declare timer variable

and used in html template

```
6   //// Local variable, #timer, version
7   @Component({
8     selector: 'app-countdown-parent-lv',
9     template: `
10    <h3>Countdown to Liftoff (via local variable)</h3>
11    <button (click)="timer.start()">Start</button>
12    <button (click)="timer.stop()">Stop</button>
13    <div class="seconds">{{timer.seconds}}</div>
14    <app-countdown-timer #timer></app-countdown-timer>
15    `,
16    styleUrls: ['../assets/demo.css']
17  })
```

Component Interaction(Parent calls an @ViewChild).

- The local variable approach is simple and easy. But it is limited because the parent-child wiring must be done entirely within the parent template. The parent component itself has no access to the child.
- You can’t use the local variable technique if an instance of the parent component class must read or write child component values or must call child component methods.
- When the parent component class requires that kind of access, inject the child component into the parent as a ViewChild.
- The following example illustrates this technique with the same Countdown Timer example. Neither its appearance nor its behavior will change. The child CountdownTimerComponent is the same as well.

Component Interaction(Parent calls an @ViewChild).

- Look at the html flow:
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-Parent-CallViewChild--BigPicture-Flow1.txt
- Click start:
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\ ComponentInteraction-Parent-CallViewChild--ClickStart-Flow1.txt

Component Interaction(Parent calls an @ViewChild).

```
1  @Component({
2    selector: 'app-countdown-parent-vc',
3    template: `
4    <h3>Countdown to Liftoff (via ViewChild)</h3>
5    <button (click)="start()">Start</button>
6    <button (click)="stop()">Stop</button>
7    <div class="seconds">{{ seconds() }}</div>
8    <app-countdown-timer></app-countdown-timer>
9    `,
10   styleUrls: ['../assets/demo.css']
11 })
12 export class CountdownViewChildParentComponent implements AfterViewInit {
13
14   @ViewChild(CountdownTimerComponent, {static: false})
15   private timerComponent: CountdownTimerComponent;
16
17   seconds() { return 0; }
18
19   ngAfterViewInit() {
20     // Redefine `seconds()` to get from the `CountdownTimerComponent.seconds` ...
21     // but wait a tick first to avoid one-time devMode
22     // unidirectional-data-flow-violation error
23     setTimeout(() => this.seconds = () => this.timerComponent.seconds, 0);
24   }
25
26   start() {
27      this.timerComponent.start();
28      }
29   stop() {
30     this.timerComponent.stop();
31   }
32 }
33
```

Inside C:\workspace-UT-Angular7-Sept-2019\Session5\component-interaction\src\app\countdown-parent.component.ts if you plan on accessing the selected element inside of ngOnInit() set static:true. If you don't access the selected element in ngOnInit(but anywhere else in your component), set static: false instead.

call the child component start and stop methods inside parent component typescript code.

Component Interaction(Parent and children communicate via a service).

- A parent component and its children share a service whose interface enables bi-directional communication within the family.
- The scope of the service instance is the parent component and its children. Components outside this component subtree have no access to the service or their communications.
- This MissionService connects the MissionControlComponent to multiple AstronautComponent children.

Component Interaction(Parent and children communicate via a service).

- Observable.subscribe()
  - The observable subscribe method is used by angular components to **subscribe to messages** that are send to an observable.
- Subject.next()
  - The subject next method is used to **send messages** to an observable which are then sent to all angular components that are subscribers(a.k.a observers) of that observable.

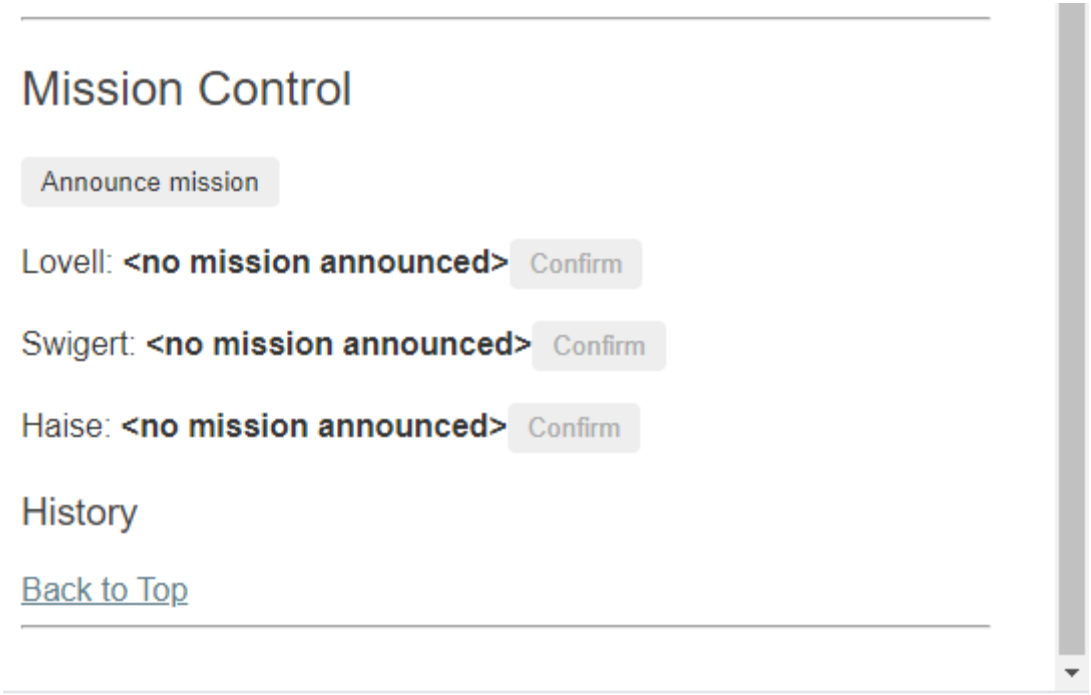
Component Interaction(Parent and children communicate via a service).

- Look at the html flow:
  - Look at C:\AlbertLam\ Angular8-UT-Jan-2020\Session5\Flow\Parent-And-Child-Communicate-Via-A-Service-BigPicture-AnnounceMission-FlowVersion1.txt

- When initial load page:
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\Parent-And-Child-Communicate-Via-A-Service-InitialLoad-AnnounceMission-FlowVersion1.txt
- Click “Announce Mission”.
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\Parent-And-Child-Communicate-Via-A-Service-ClickAnnounceMission-AnnounceMission-FlowVersion1.txt
- Click first “Confirm”.
  - Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\Parent-And-Child-Communicate-Via-A-Service-ClickFirstConfirm-AnnounceMission-FlowVersion1.txt

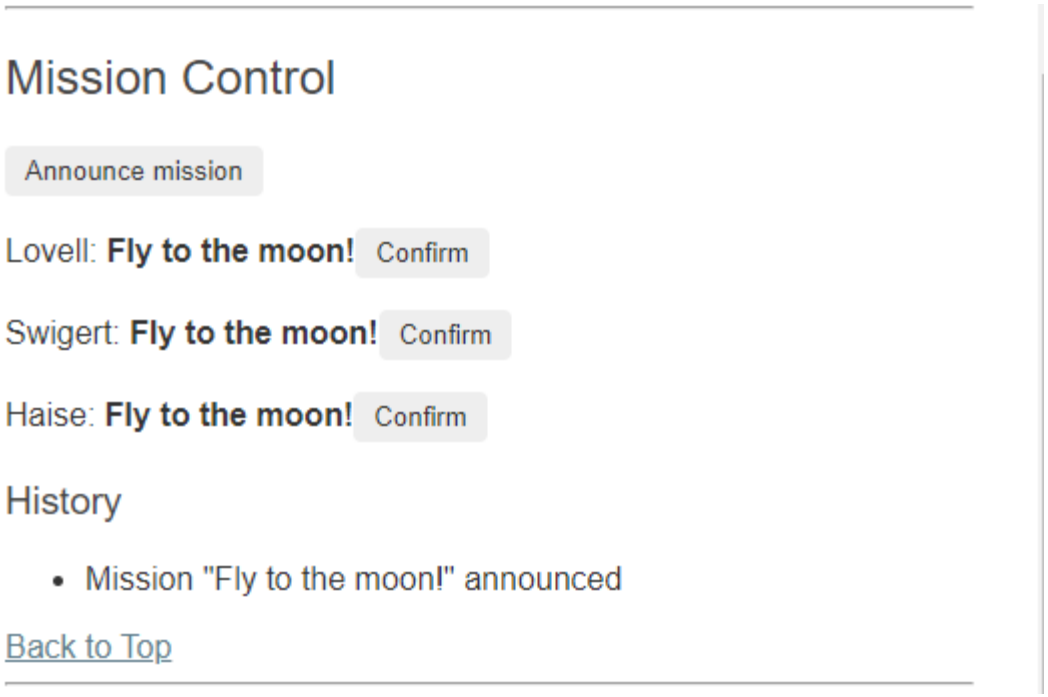
Component Interaction(Parent and children communicate via a service).

- Initial Load:



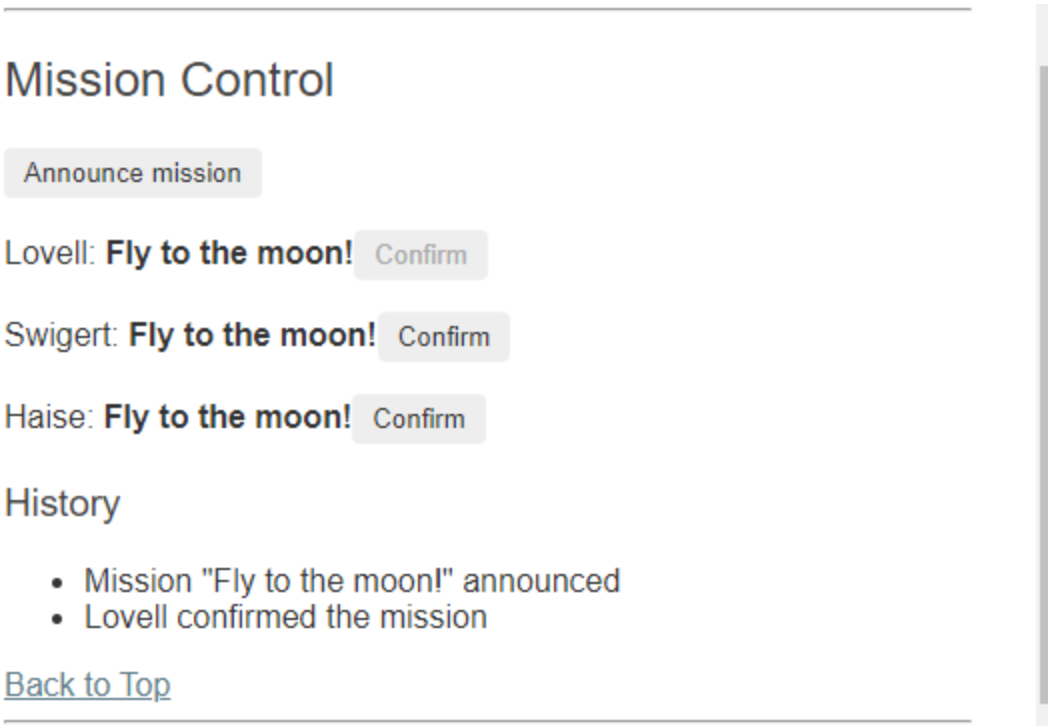
Component Interaction(Parent and children communicate via a service).

- Announce Mission:



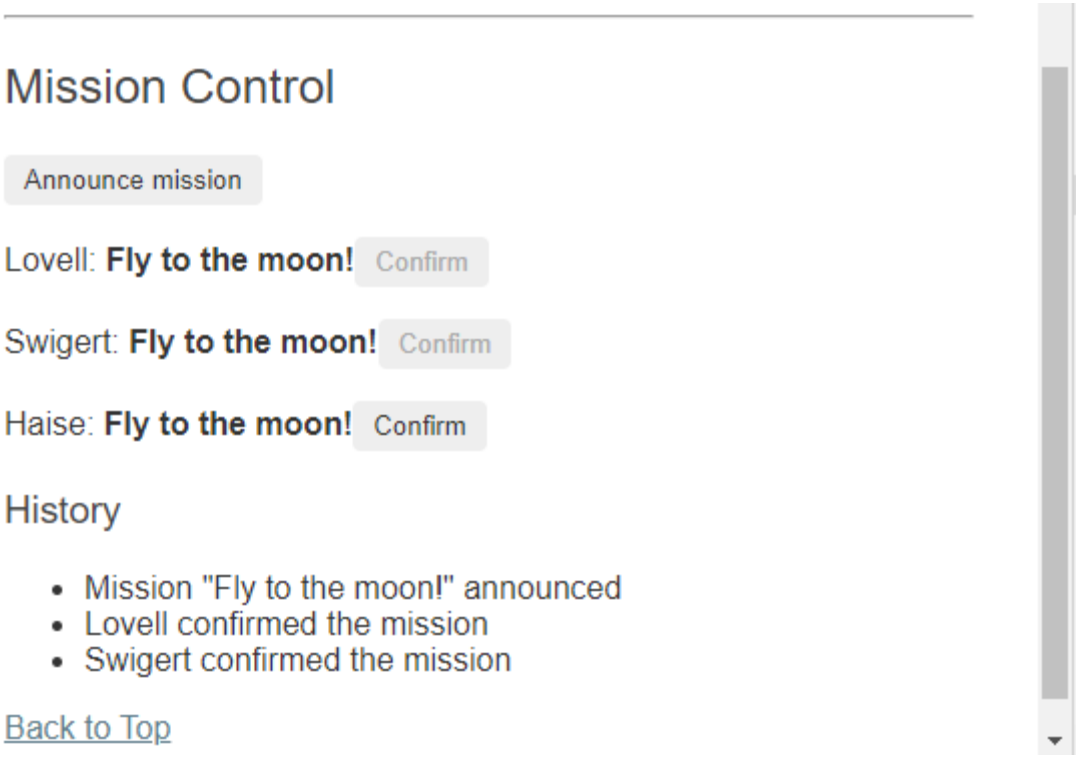
Component Interaction(Parent and children communicate via a service).

- Click First “Confirm”.



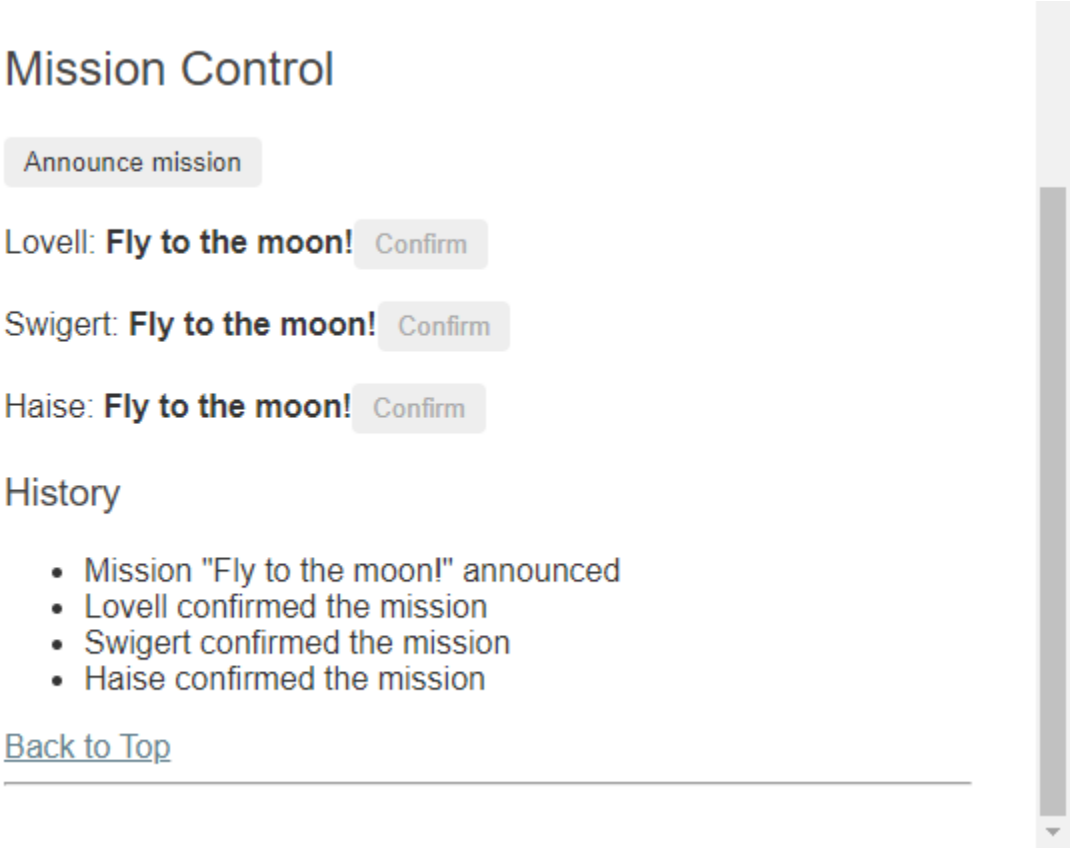
Component Interaction(Parent and children communicate via a service).

- Click Second “Confirm”.



Component Interaction(Parent and children communicate via a service).

- Click Third “Confirm”.



Component Interaction(Parent and children communicate via a service).

ent.ts | TS countdown-parent.component.ts | TS *mission.service.ts* ×

src > app > TS mission.service.ts > ...

Inside mission.service.ts file

```ts
import { Injectable } from '@angular/core';
import { Subject }    from 'rxjs';

@Injectable()
export class MissionService {

  // Observable string sources
  private missionAnnouncedSource = new Subject<string>();
  private missionConfirmedSource = new Subject<string>();

  // Observable string streams
  missionAnnounced$ = this.missionAnnouncedSource.asObservable();
  missionConfirmed$ = this.missionConfirmedSource.asObservable();

  // Service message commands
  announceMission(mission: string) {
    this.missionAnnouncedSource.next(mission);
  }

  confirmMission(astronaut: string) {
    this.missionConfirmedSource.next(astronaut);
  }
```

Component Interaction(Parent and children communicate via a service).

- The MissionControlComponent both provides the instance of the service that it shares with its children(through the providers metadata arra) and injects that instance into itself through its constructor

Component Interaction(Parent and children communicate via a service).

```
1  import { Component }          from '@angular/core';
2
3  import { MissionService }     from './mission.service';
4
5  @Component({
6    selector: 'app-mission-control',
7    template: `
8      <h2>Mission Control</h2>
9      <button (click)="announce()">Announce mission</button>
10     <app-astronaut *ngFor="let astronaut of astronauts"
11       [astronaut]="astronaut">
12     </app-astronaut>
13     <h3>History</h3>
14     <ul>
15       <li *ngFor="let event of history">{{event}}</li>
16     </ul>
17   `,
18   providers: [MissionService]
19 })
20 export class MissionControlComponent {
21   astronauts = ['Lovell', 'Swigert', 'Haise'];
22   history: string[] = [];
23   missions = ['Fly to the moon!',
24               'Fly to mars!',
25               'Fly to Vegas!'];
26   nextMission = 0;
27
28   constructor(private missionService: MissionService) {
29     missionService.missionConfirmed$.subscribe(
30       astronaut => {
31         this.history.push(`${astronaut} confirmed the mission`);
32       });
33   }
34
35   announce() {
36     let mission = this.missions[this.nextMission++];
37     this.missionService.announceMission(mission);
38     this.history.push(`Mission "${mission}" announced`);
39     if (this.nextMission >= this.missions.length) { this.nextMission = 0; }
40   }
41 }
42
```

The MissionControlComponent injects the MissionService instance through its contructor.

Component Interaction(Parent and children communicate via a service).

- The AstronautComponent also injects the service in its constructor. Each AstronautComponent is a child of the MissionControlComponent and therefore receives its parent’s service instance.

Component Interaction(Parent and children communicate via a service).

```
import { Subscription }   from 'rxjs';

@Component({
  selector: 'app-astronaut',
  template: `
    <p>
      {{astronaut}}: <strong>{{mission}}</strong>
      <button
        (click)="confirm()"
        [disabled]="!announced || confirmed">
        Confirm
      </button>
    </p>
  `
})
export class AstronautComponent implements OnDestroy {
  @Input() astronaut: string;
  mission = '<no mission announced>';
  confirmed = false;
  announced = false;
  subscription: Subscription;

  constructor(private missionService: MissionService) {
    this.subscription = missionService.missionAnnounced$.subscribe(
      mission => {
        this.mission = mission;
        this.announced = true;
        this.confirmed = false;
    });
  }

  confirm() {
    this.confirmed = true;
    this.missionService.confirmMission(this.astronaut);
  }

  ngOnDestroy() {
    // prevent memory leak when component destroyed
    this.subscription.unsubscribe();
  }
}
```

This is astronaut.component.ts

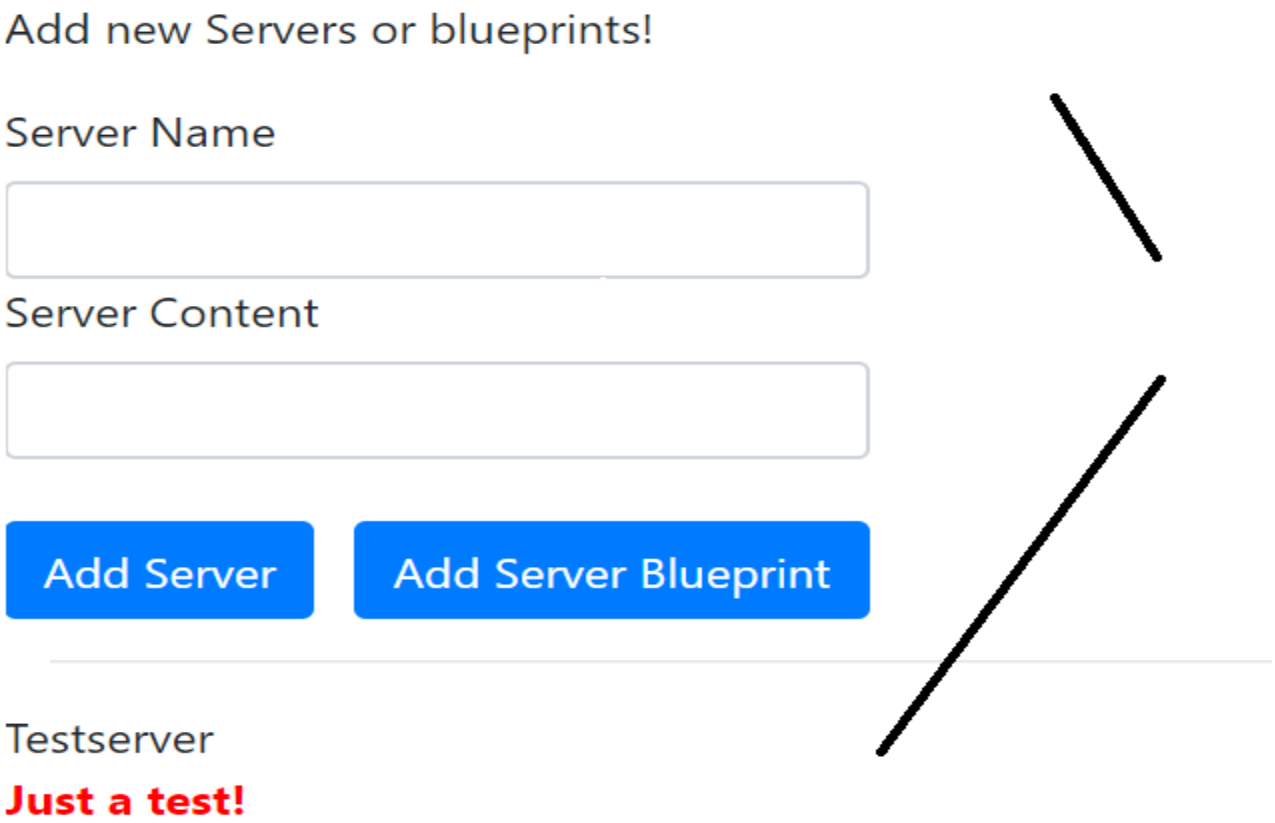
The AstronautComponent injects the MissionService service in its constructor

Splitting Apps into components

- Download example3-component-and-databinding-Version1\cmp-databinding-start.
- Copy C:\workspace-UT-Angular8-SEPT-2020 \session5\example3-component-and-databinding-Version1\cmp-databinding-start to C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version2

Splitting Apps into components

- This is what it looks like when example4-component-and-databinding-Version2 to example4-component-and-databinding-Version5 is finished.

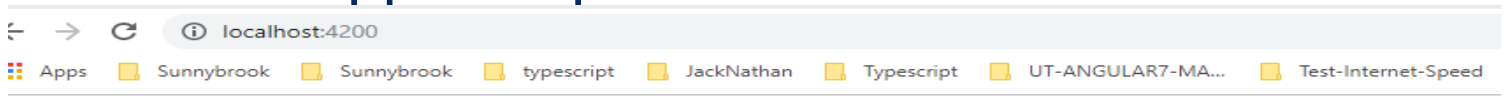


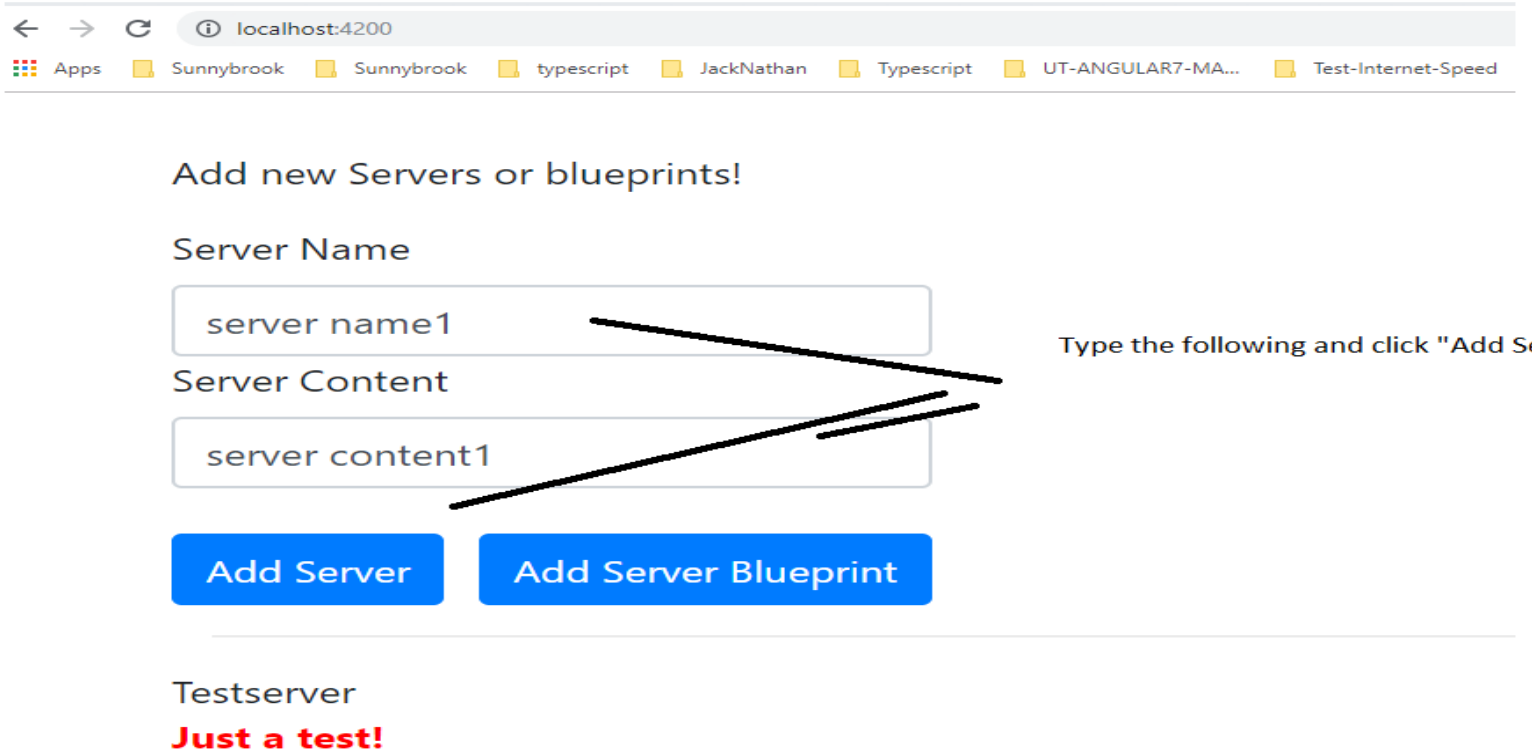
Splitting Apps into components

- Copy C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\code-before-npm-install\example4-component-and-databinding-Version5-Finish to C:\workspace-UT-Angular8-SEPT-2020 \Session5
- Go to C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version5-Finish\cmp-databinding-start in dos-prompt for windows or terminal for mac and do the following
- Command:npm install
- Go over C:\AlbertLam\Angular8-UT-SEPT-2020 \Session5\Flow\example4-component-and-databinding-Version5-app-server-element-FlowVersion1.txt for app-server-element.

Splitting Apps into components

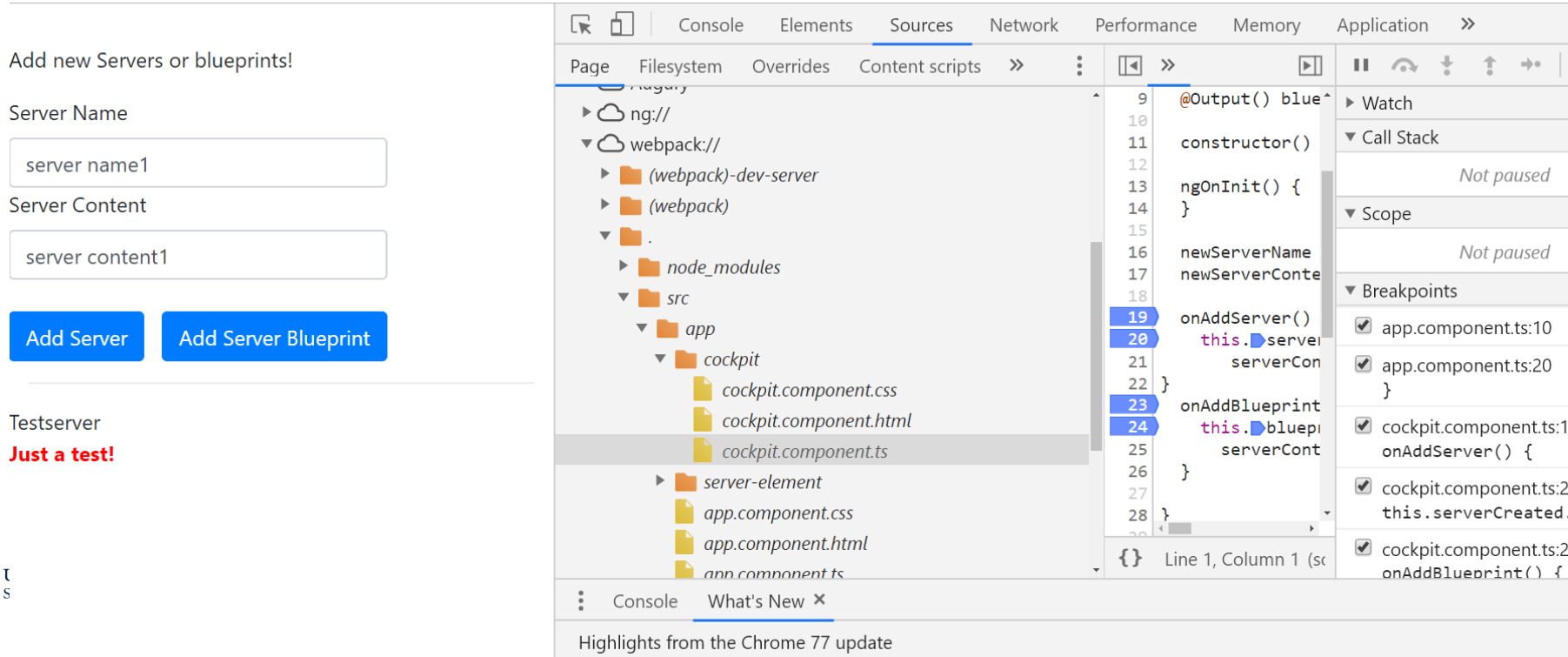
- Go over C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\example4-component-and-databinding-Version5-app-cockpit-FlowVersion1.txt





Splitting Apps into components

- Go over C:\AlbertLam\Angular8-UT-Jan-2020 \Session5\Flow\example4-component-and-databinding-Version5-app-cockpit-ActualFlowVersion1.txt



Splitting Apps into components

- Create two new components.
- First component:
- ng g c cockpit --spec false

```
C:\workspace-UT-Angular7-Sept-2019\Session5\example4-component-and-databinding-Version2\cmp-databinding-start>ng g c cockpit --spec false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/cockpit/cockpit.component.html (26 bytes)
CREATE src/app/cockpit/cockpit.component.ts (273 bytes)
CREATE src/app/cockpit/cockpit.component.css (0 bytes)
UPDATE src/app/app.module.ts (523 bytes)

C:\workspace-UT-Angular7-Sept-2019\Session5\example4-component-and-databinding-Version2\cmp-databinding-start>
```

Splitting Apps into components

C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version2\cmp-databinding-start>ng g c cockpit --spec false

Option "spec" is deprecated: Use "skipTests" instead.

CREATE src/app/cockpit/cockpit.component.html (26 bytes)

CREATE src/app/cockpit/cockpit.component.ts (273 bytes)

CREATE src/app/cockpit/cockpit.component.css (0 bytes)

UPDATE src/app/app.module.ts (523 bytes)

C:\workspace-UT-Angular8-SEPT-2020\Session5\example4-component-and-databinding-Version2\cmp-databinding-start>

Splitting Apps into components

- Create another new component:
- ng g c server-element --spec false

```
C:\workspace-UT-Angular7-Sept-2019\Session5\example4-component-and-databinding-Version2\cmp-databinding-start>ng g c server-element --spec false
Option "spec" is deprecated: Use "skipTests" instead.
CREATE src/app/server-element/server-element.component.html (33 bytes)
CREATE src/app/server-element/server-element.component.ts (300 bytes)
CREATE src/app/server-element/server-element.component.css (0 bytes)
UPDATE src/app/app.module.ts (635 bytes)
```

Splitting Apps into components

- Create another new component:
- ng g c server-element --spec false

C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version2\cmp-databinding-start>ng g c server-element --spec false

Option "spec" is deprecated: Use "skipTests" instead.

CREATE src/app/server-element/server-element.component.html (33 bytes)

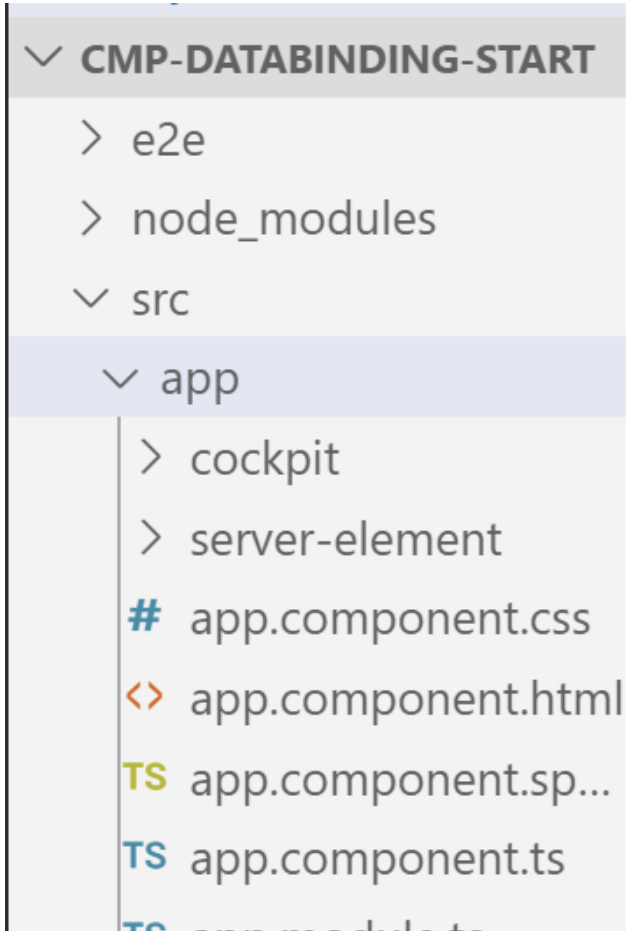
CREATE src/app/server-element/server-element.component.ts (300 bytes)

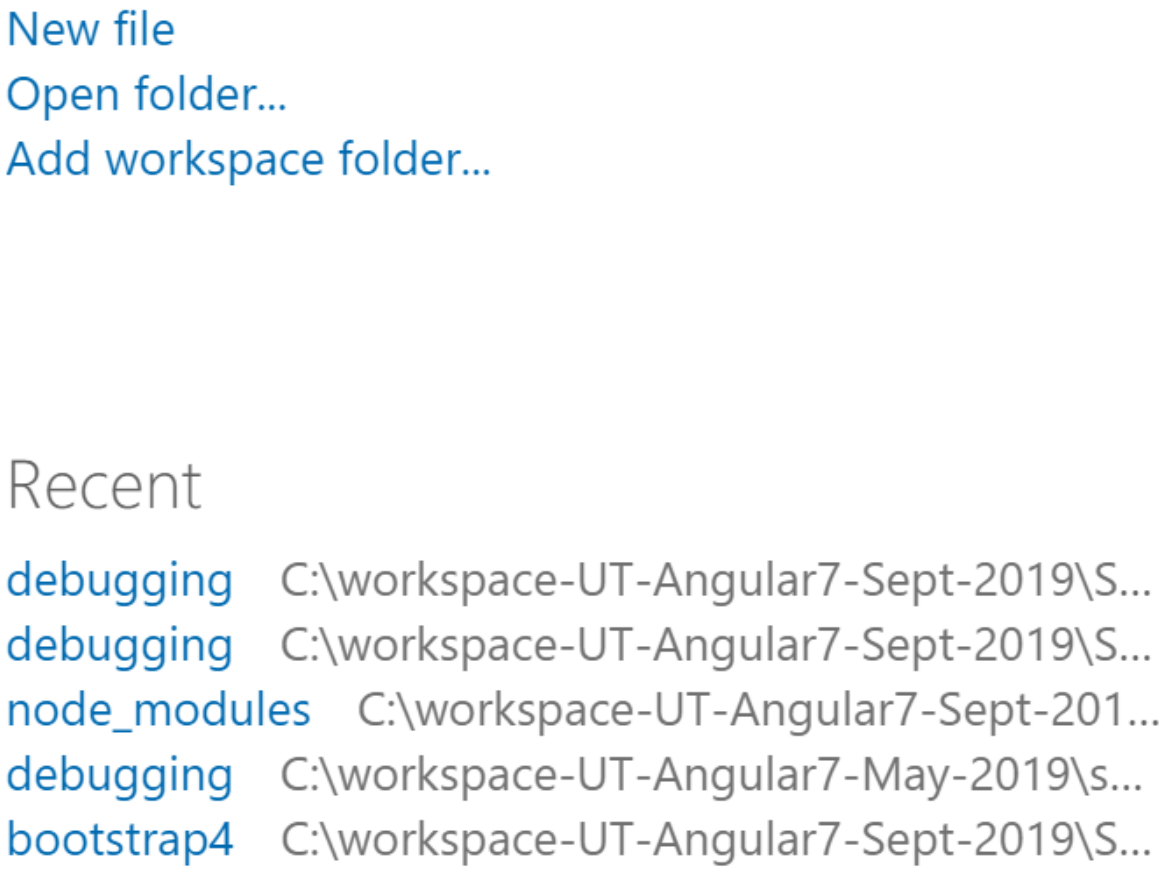
CREATE src/app/server-element/server-element.component.css (0 bytes)

UPDATE src/app/app.module.ts (635 bytes)

Splitting Apps into components

- Two new folders in app components:cockpit and server-element





example4-component-and-databinding-Version2

- From C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version2\cmp-databinding-start\src\app\app.component.html cut (in the select data for “<!-- Begin cockpit-->” to “<!-- End cockpit-->”)the html and paste to C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version2\cmp-databinding-start\src\app\cockpit\cockpit.component.html

example4-component-and-databinding-Version2

app.component.html | cockpit.component.html

src ▸ app ▸ app.component.html ▸ div.container ▸ div.row

```
1   <div class="container">
2     <!-- Begin cockpit-->
3     <div class="row">
4       <div class="col-xs-12">
5         <p>Add new Servers or blueprints!</p>
6         <label>Server Name</label>
7         <input type="text" class="form-control" [(ngModel)]="newServerName">
8         <label>Server Content</label>
9         <input type="text" class="form-control" [(ngModel)]="newServerContent">
10        <br>
11        <button
12          class="btn btn-primary"
13          (click)="onAddServer()">Add Server</button>
14        <button
15          class="btn btn-primary"
16          (click)="onAddBlueprint()">Add Server Blueprint</button>
17      </div>
18    </div>
19    <!-- End cockpit-->
20    <hr>
```

Cut the code between "Begin cockpit" and "End cockpit" and paste to cockpit.component.html and overwrite everything in this file.

example4-component-and-databinding-Version2

- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version2\cmp-databinding-start\src\app\app.component.ts
- Copy the code from app.component.ts to cockpit.component.ts:

example4-component-and-databinding-Version2

- Copy the begin and end code from app.component.ts to cockpit.component.ts:

```
11
12    /* Begin cockpit */
13
14    newServerName = '';
15    newServerContent = '';
16
17    onAddServer() {
18      this.serverElements.push({
19        type: 'server',
20        name: this.newServerName,
21        content: this.newServerContent
22      });
23    }
24
25    onAddBlueprint() {
26      this.serverElements.push({
27        type: 'blueprint',
28        name: this.newServerName,
29        content: this.newServerContent
30      });
31    }
32
33    /* End cockpit */
```

example4-component-and-databinding-Version2

- Move from app.component.html to server-element.component.html

ease Notes: 1.36.1 | TS cockpit.component.ts | <> app.component.html ✕

cmp-databinding-start ▸ src ▸ app ▸ <> app.component.html ▸ div.container ▸

```
1   <div class="container">
2     <!-- Begin cockpit-->
3
4     <!-- End cockpit-->
5     <hr>
6     <div class="row">
7       <div class="col-xs-12">
8         <!-- begin server element-->
9         <div
10          class="panel panel-default"
11          *ngFor="let element of serverElements">
12          <div class="panel-heading">{{ element.name }}</div>
13          <div class="panel-body">
14            <p>
15              <strong *ngIf="element.type === 'server'" style="color: red">{{ ele
16              <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
17            </p>
18          </div>
19        </div>
20        <!-- end server element-->
```

Move from app.component.html to server-element.component.html

example4-component-and-databinding-Version3

- Copy C:\workspace-UT-Angular7-May-2019\session5\example4-component-and-databinding-Version2\cmp-databinding-start to C:\workspace-UT-Angular7-May-2019\session5\example4-component-and-databinding-Version3\
- Edit C:\workspace-UT-Angular7-May-2019\session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\server-element\server-element.component.html and delete

example4-component-and-databinding-Version3

Delete "*ngFor="let element of serverElements"

server-element.component.html

src ▸ app ▸ server-element ▸ server-element.component.html ▸ ...

```
1   <!-- begin server element-->
2   <div
3   class="panel panel-default"
4   *ngFor="let element of serverElements">
5   <div class="panel-heading">{{ element.name }}</div>
6   <div class="panel-body">
7     <p>
8       <strong *ngIf="element.type === 'server'" style="color: red">{{ element.content }}
9       <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
10    </p>
11  </div>
12  </div>
13  <!-- end server element-->
14
```

example4-component-and-databinding-Version3

After deletion

server-element.component.html

src ▸ app ▸ server-element ▸ server-element.component.html ▸ div.panel.panel-default

```
1   <!-- begin server element-->
2   <div
3   class="panel panel-default">
4   <div class="panel-heading">{{ element.name }}</div>
5   <div class="panel-body">
6     <p>
7       <strong *ngIf="element.type === 'server'" style="color: red">{{ element.content }}</
8       <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
9     </p>
10  </div>
11  </div>
12  <!-- end server element-->
13
```

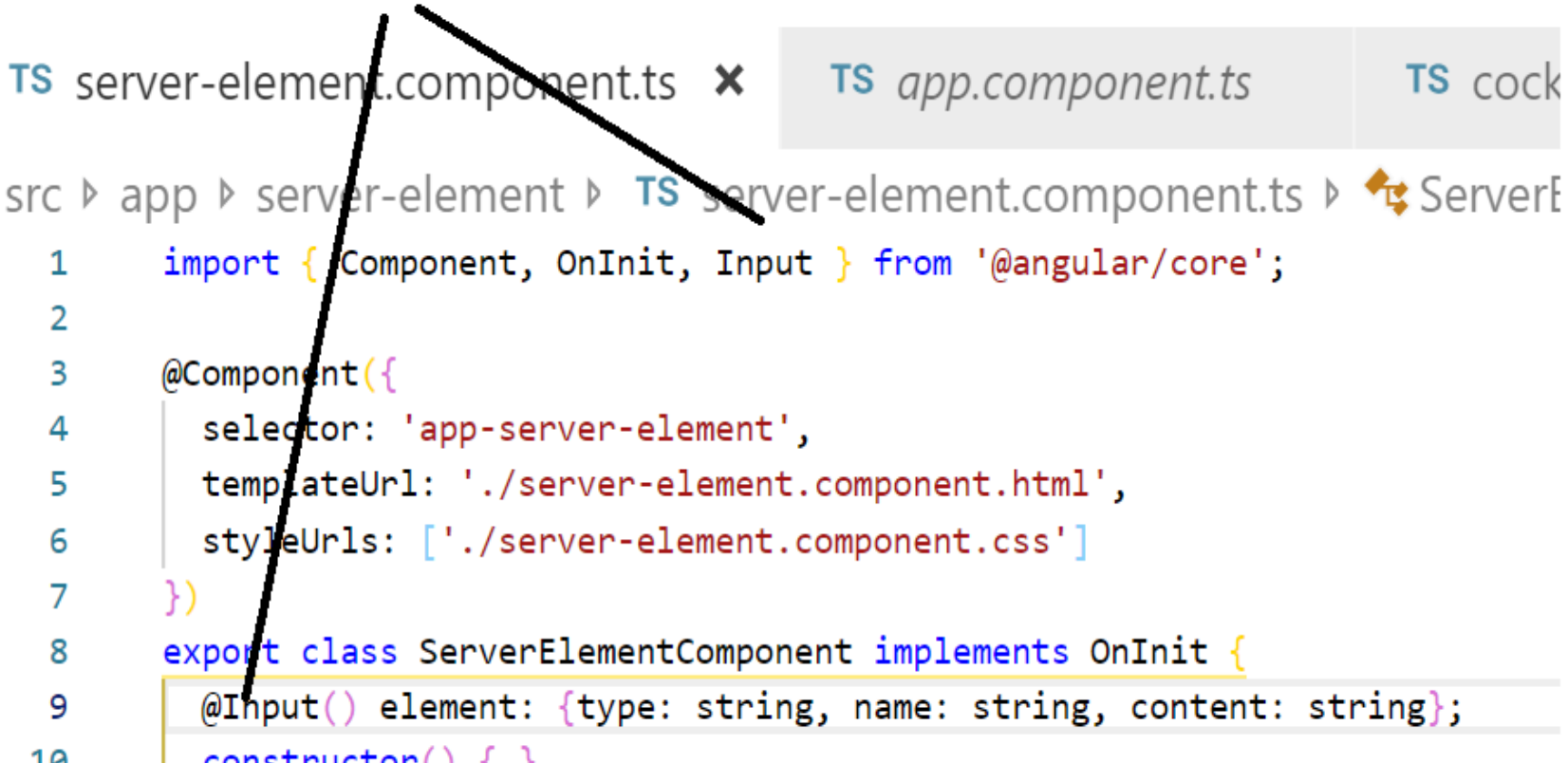
example4-component-and-databinding-Version3

- We need to be able to send data into a component or receive data receive an event and Angela of course give us great tools to implement this flow.
- We can use property in event binding not only on HTML elements and their native properties and events.
- We can also use property in event binding on our own components and bind to our own custom properties and custom events. We can emit our own events.
- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\server-element\server-element.component.ts

example4-component-and-databinding-Version3

- Edit C:\workspace-UT-Angular8-SEPT-2020\session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\server-element\server-element.component.ts. **Server-element.component.ts is a child component because it contains "@Input() element". The parent component is the component that uses "app-server-element".**

The decorator "@Input()" allow us to expose this property element to the world. In "import" add "Input"



example4-component-and-databinding-Version3

- Edit C:\workspace-UT-Angular8-SEPT-2020\session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\cockpit\cockpit.component.ts

example4-component-and-databinding-Version3

- Edit C:\workspace-UT-Angular8-SEPT-2020\session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\app.component.ts

Add this line

app-root is parent component that uses app-server-element.

src ▸ app ▸ TS app.component.ts ▸ AppComponent

```
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     serverElements = [{type: 'server', name: 'Testserver', content: 'Just a test!'}];
10
11
```

example4-component-and-databinding-Version3

- Edit C:\workspace-UT-Angular8-SEPT-2020\session5\example4-component-and-databinding-Version3\cmp-databinding-start\src\app\app.component.html

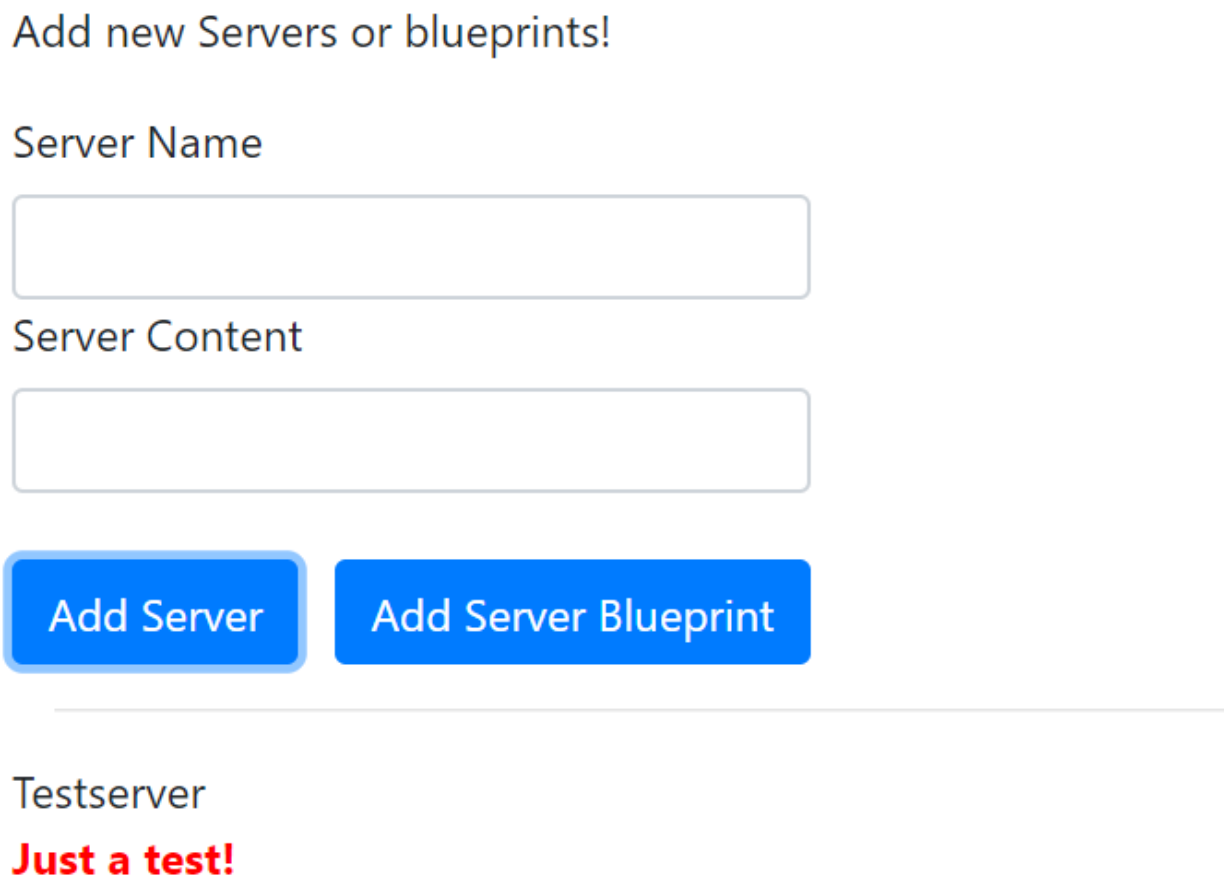
onent.ts | TS cockpit.component.ts | TS app.component.ts | app.component.html

src > app > app.component.html > div.container > app-cockpit

```
1   <div class="container">
2     <!-- Begin cockpit-->
3     <app-cockpit></app-cockpit>
4     <!-- End cockpit-->
5     <hr>
6     <div class="row">
7       <div class="col-xs-12">
8         <!-- begin server element-->
9         <app-server-element *ngFor="let serverElement of serverElements"
10        [element]="serverElement"></app-server-element>
11          <!-- end server element-->
12      </div>
13    </div>
14  </div>
15
```

Add these two new lines

example4-component-and-databinding-Version3



Assigning an Alias to custom properties(example4-component-and-databinding-Version4)

- Copy C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version3\cmp-databinding-start to C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version4
- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version4\cmp-databinding-start\src\app\server-element\server-element.component.ts

Assigning an Alias to custom properties(example4-component-and-databinding-Version4)

TS server-element.component.ts ✖

src ▸ app ▸ server-element ▸ TS server-element.component.ts ▸ ...

```
1   import { Component, OnInit, Input } from '@angular/core';
2
3   @Component({
4     selector: 'app-server-element',
5     templateUrl: './server-element.component.html',
6     styleUrls: ['./server-element.component.css']
7   })
8   export class ServerElementComponent implements OnInit {
9     @Input('srvElement') element: {type: string, name: string, content: string};
10    constructor() { }
11
12    ngOnInit() {
13    }
14
15  }
16
```

Add alias 'srvElement'

Assigning an Alias to custom properties(example4-component-and-databinding-Version4)

- Edit C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version4\cmp-databinding-start\src\app\app.component.html

TS server-element.component.ts ✕ <> app.component.html ✕

p ▸ <> app.component.html ▸ div.container ▸ div.row ▸ div.col-xs-12 ▸ ap

```
1   <div class="container">
2     <!-- Begin cockpit-->
3     <app-cockpit></app-cockpit>
4     <!-- End cockpit-->
5     <hr>
6     <div class="row">
7       <div class="col-xs-12">
8         <!-- begin server element-->
9           <app-server-element *ngFor="let serverElement of serverElements"
10          [srvElement]="serverElement"></app-server-element>
11        <!-- end server element-->
12      </div>
13    </div>
```

Change from element to 'srvElement' for the new alias.

Binding to custom events(example4-component-and-databinding-Version5)

- Copy C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version4\cmp-databinding-start\ to C:\workspace-UT-Angular8-SEPT-2020 \session5\example4-component-and-databinding-Version5
- In previous examples, we learn how to pass data from a component to another component which was implemented there. We pass data from parent component app.component.html to child component
- Now about the other direction what if we have a component and something changes in there and we want to inform our parent component so the component which implements the average component.

Binding to custom events(example4-component-and-databinding-Version5)

- For example here an app component we're implementing app cockpit. And in this child component in app cockpit something can change because here we got two buttons and when we click the buttons we want to do something. And right now the code which would normally get executed was simply commented out.

Binding to custom events(example4-component-and-databinding-Version5: app.component.ts)

Binding to custom events(example4-component-and-databinding-Version5:app.component.ts)

```
export class AppComponent {
serverElements = [{type: 'server', name: 'Testserver', content: 'Just a test!'}];


onServerAdded(serverData: {serverName: string, serverContent: string}) {

  this.serverElements.push({
    type: 'server',
    name: serverData.serverName,
    content: serverData.serverContent
  });

}

onBlueprintAdded(blueprintData: {serverName: string, serverContent: string}) {
  this.serverElements.push({
     type: 'blueprint',
     name: blueprintData.serverName,
     content: blueprintData.serverContent
  });
 }
}
```

Binding to custom events(example4-component-and-databinding-Version5:app.component.html)

app.component.html | app.component.ts

src > app > app.component.html > ...

add this line

```
1   <div class="container">
2     <!-- Begin cockpit-->
3     <app-cockpit (serverCreated)="onServerAdded($event)"
4     (blueprintCreated)="onBlueprintAdded($event)"
5     ></app-cockpit>
6     <!-- End cockpit-->
7     <hr>
8     <div class="row">
9       <div class="col-xs-12">
10        <!-- begin server element-->
11        <app-server-element *ngFor="let serverElement of serverElements"
12        [srvElement]="serverElement"></app-server-element>
13          <!-- end server element-->
14      </div>
15    </div>
16  </div>
```

Binding to custom events(example4-component-and-databinding-Version5:app.component.html

```
<!-- Begin cockpit-->
 <app-cockpit (serverCreated)="onServerAdded($event)"
 (blueprintCreated)="onBlueprintAdded($event)"
 ></app-cockpit>
 <!-- End cockpit-->
```

Binding to custom events(example4-component-and-databinding-Version5:app.component.html)

```
<!-- Begin cockpit-->
 <app-cockpit (serverCreated)="onServerAdded($event)"
 (blueprintCreated)="onBlueprintAdded($event)"
 ></app-cockpit>
 <!-- End cockpit-->
```

Binding to custom events(example4-component-

src > app > cockpit > TS cockpit.component.ts > CockpitComponent

```
1   import { Component, OnInit, EventEmitter, Output } from '@angular/core';
2   @Component({
3     selector: 'app-cockpit',
4     templateUrl: './cockpit.component.html',
5     styleUrls: ['./cockpit.component.css']
6   })
7   export class CockpitComponent implements OnInit {
8     @Output() serverCreated = new EventEmitter<{serverName: string, serverContent: string}>();
9     @Output() blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();
10
11    constructor() { }
12
13    ngOnInit() {
14    }
15
16    newServerName = '';
17    newServerContent = '';
18
19    onAddServer() {
20      this.serverCreated.emit({serverName: this.newServerName,
21        serverContent: this.newServerContent });
22    }
23    onAddBlueprint() {
24      this.blueprintCreated.emit({serverName: this.newServerName,
25        serverContent: this.newServerContent });
26    }
```

Add these import

Add these two lines

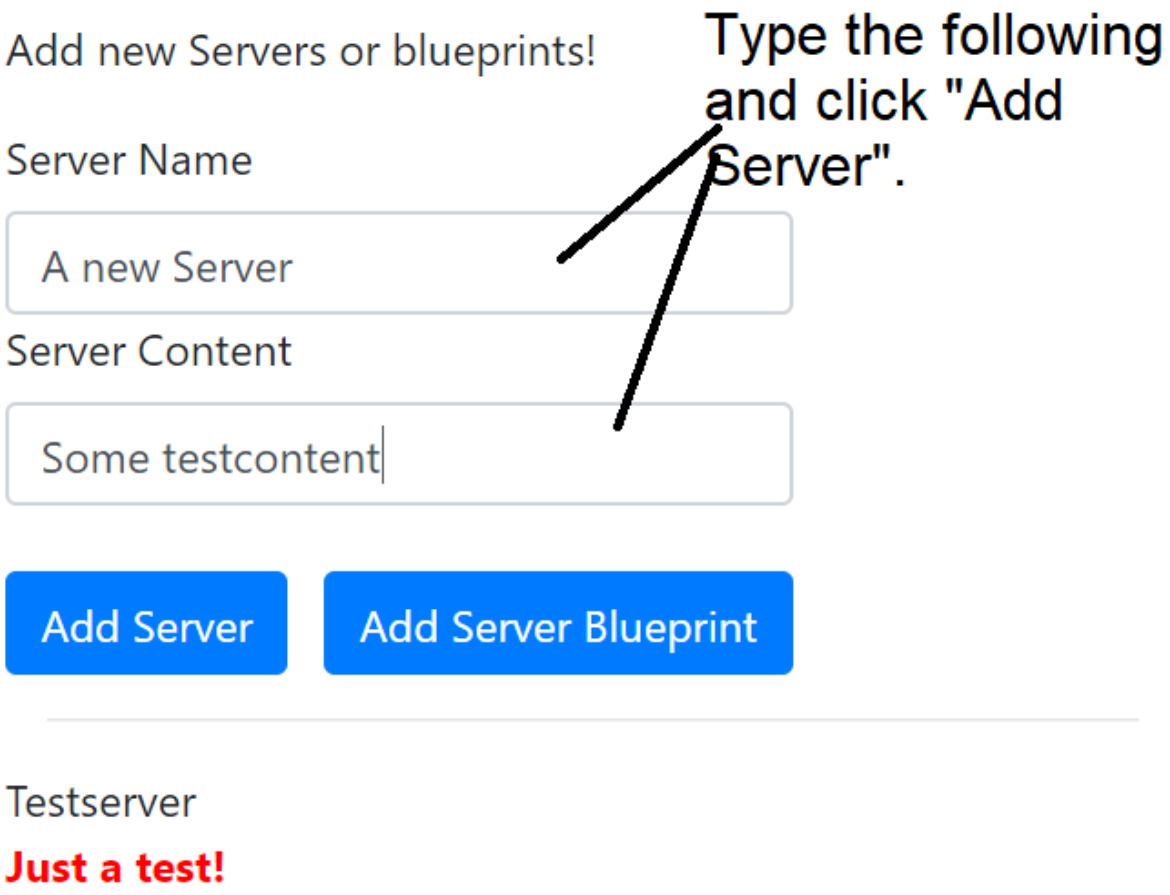
Make sure these 2 variables exist

Add these two methods

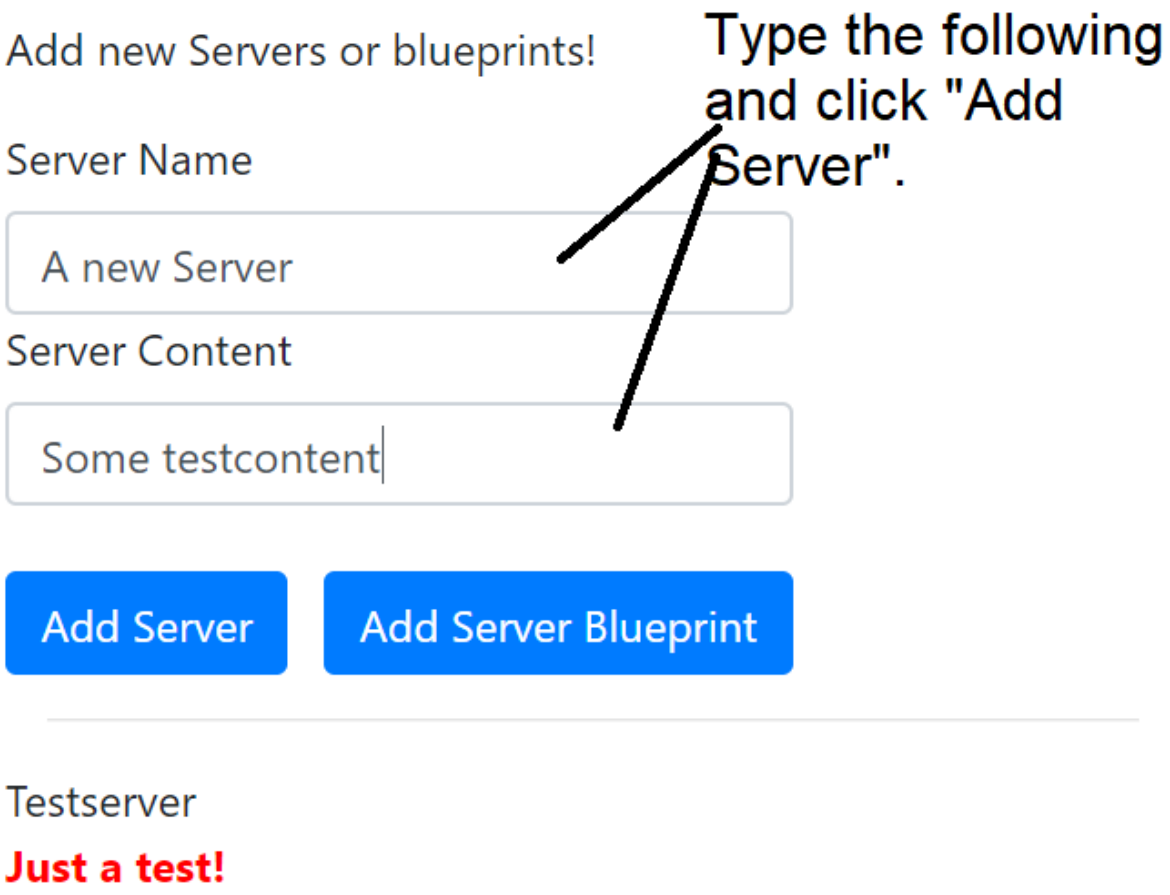
Binding to custom events(example4-component-and-databinding-Version5:cockpitComponent.ts)

```
import { Component, OnInit, EventEmitter, Output } from '@angular/core';
@Component({
  selector: 'app-cockpit',
  templateUrl: './cockpit.component.html',
  styleUrls: ['./cockpit.component.css']
})
export class CockpitComponent implements OnInit {
  @Output() serverCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  @Output() blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();

  constructor() { }

  ngOnInit() {
  }

  newServerName = '';
  newServerContent = '';

  onAddServer() {
    this.serverCreated.emit({serverName: this.newServerName,
      serverContent: this.newServerContent });
}
  onAddBlueprint() {
    this.blueprintCreated.emit({serverName: this.newServerName,
      serverContent: this.newServerContent });
  }

}
```

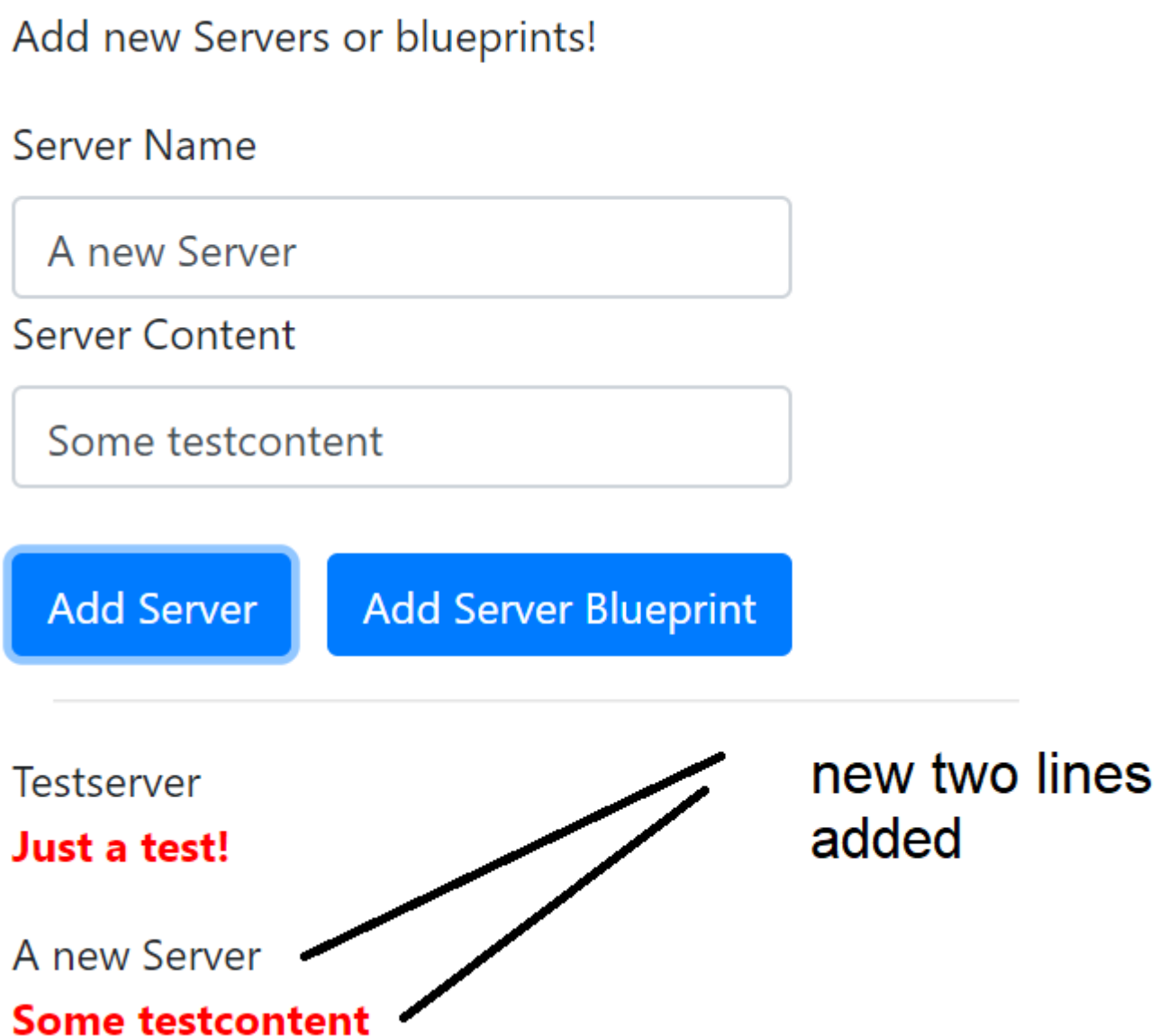
Binding to custom events(example4-component-and-databinding-Version5:cockpitComponent.ts)



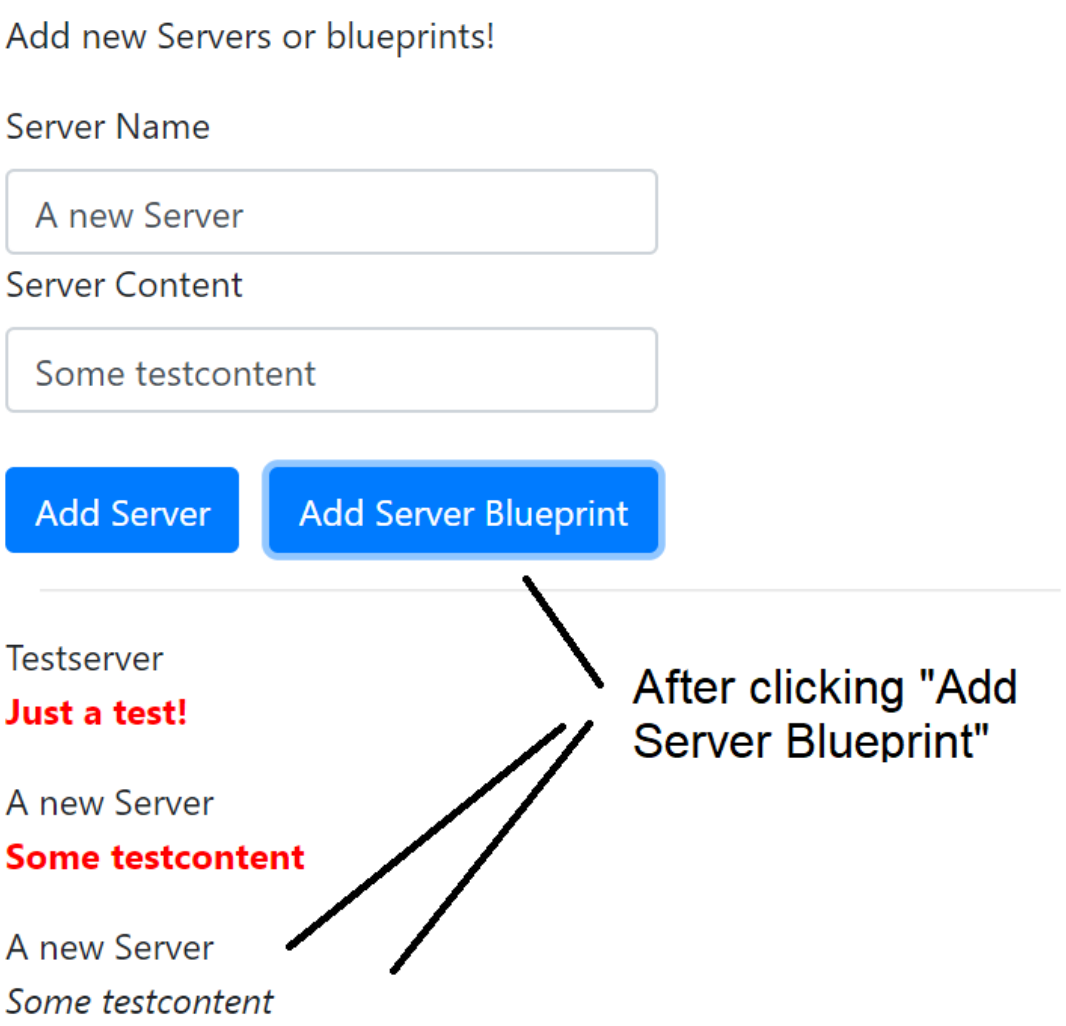
Binding to custom events(example4-component-and-databinding-Version5:cockpitComponent.ts)



Binding to custom events(example4-component-and-databinding-Version5:cockpitComponent.ts)



Binding to custom events(example4-component-and-databinding-Version5:cockpitComponent.ts)



Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)

- Copy C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version5\cmp-databinding-start to C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version6

Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)

- Copy C:\workspace-UT-Angular8-SEPT-2020 \Session5\example4-component-and-databinding-Version5\cmp-databinding-start to C:\workspace-UT-Angular8-SEPT-2020\Session5\example4-component-and-databinding-Version6

Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)

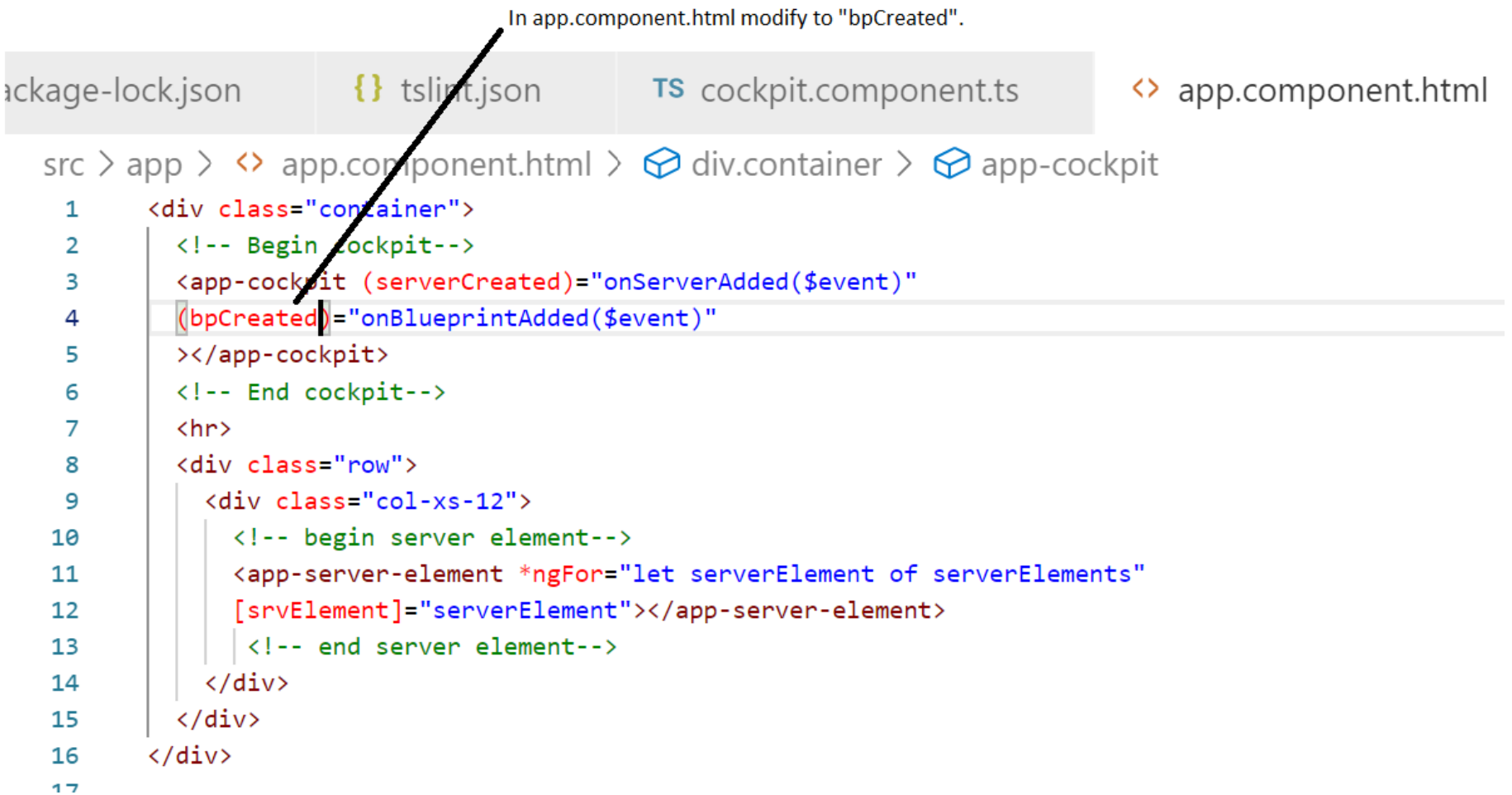
package-lock.json | tslint.json | cockpit.component.ts

src > app > cockpit > cockpit.component.ts > CockpitComponent > blueprintCreated

```
1   import { Component, OnInit, EventEmitter, Output } from '@angular/core';
2   @Component({
3     selector: 'app-cockpit',
4     templateUrl: './cockpit.component.html',
5     styleUrls: ['./cockpit.component.css']
6   })
7   export class CockpitComponent implements OnInit {
8     @Output() serverCreated = new EventEmitter<{serverName: string, serverContent: string}>();
9     @Output('bpCreated') blueprintCreated = new EventEmitter<{serverName: string, serverContent: str
10
11    constructor() { }
12
13    ngOnInit() {
14    }
15
16    newServerName = '';
```

Add this alias "bpCreated" in cockpit.component.ts

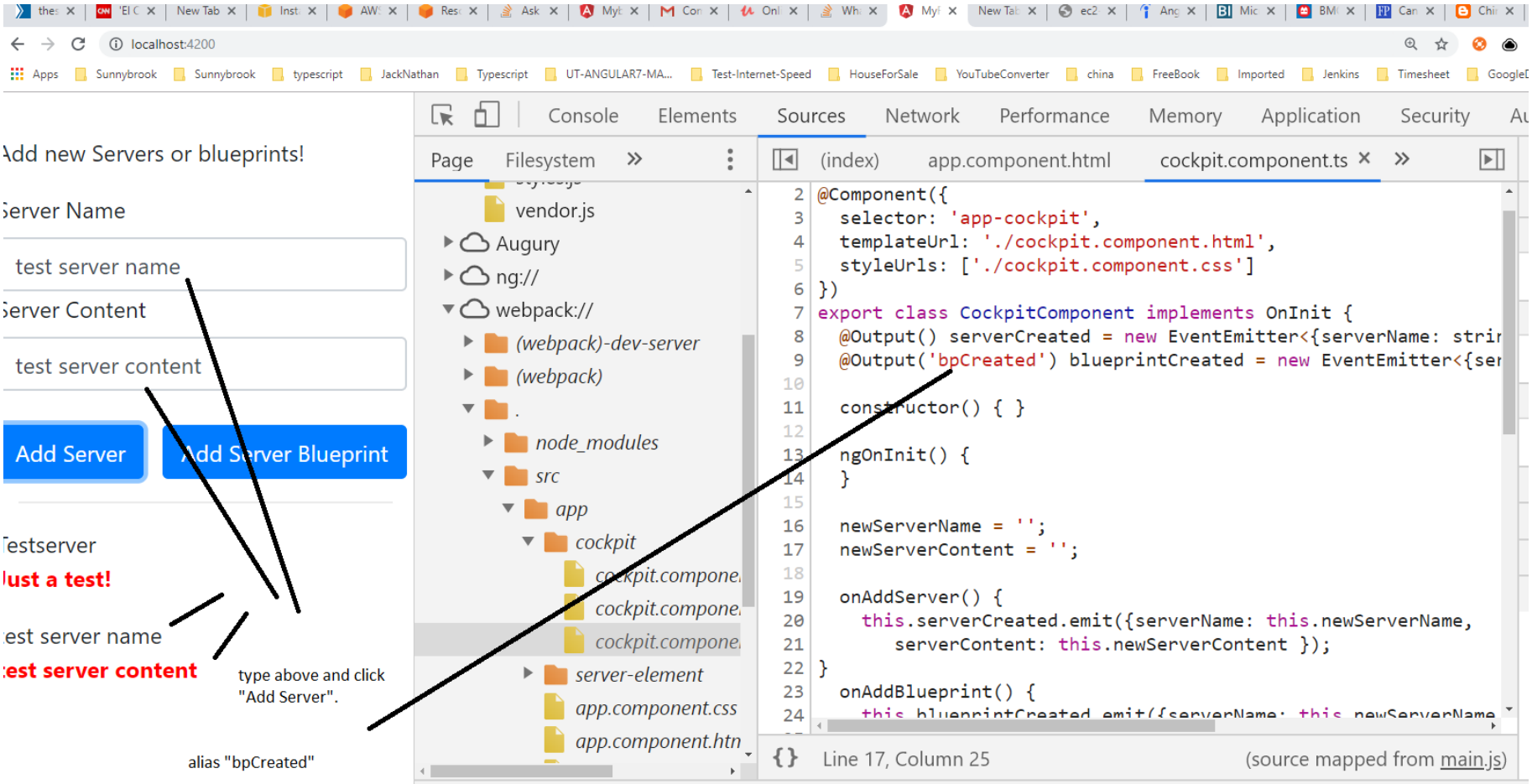
Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)



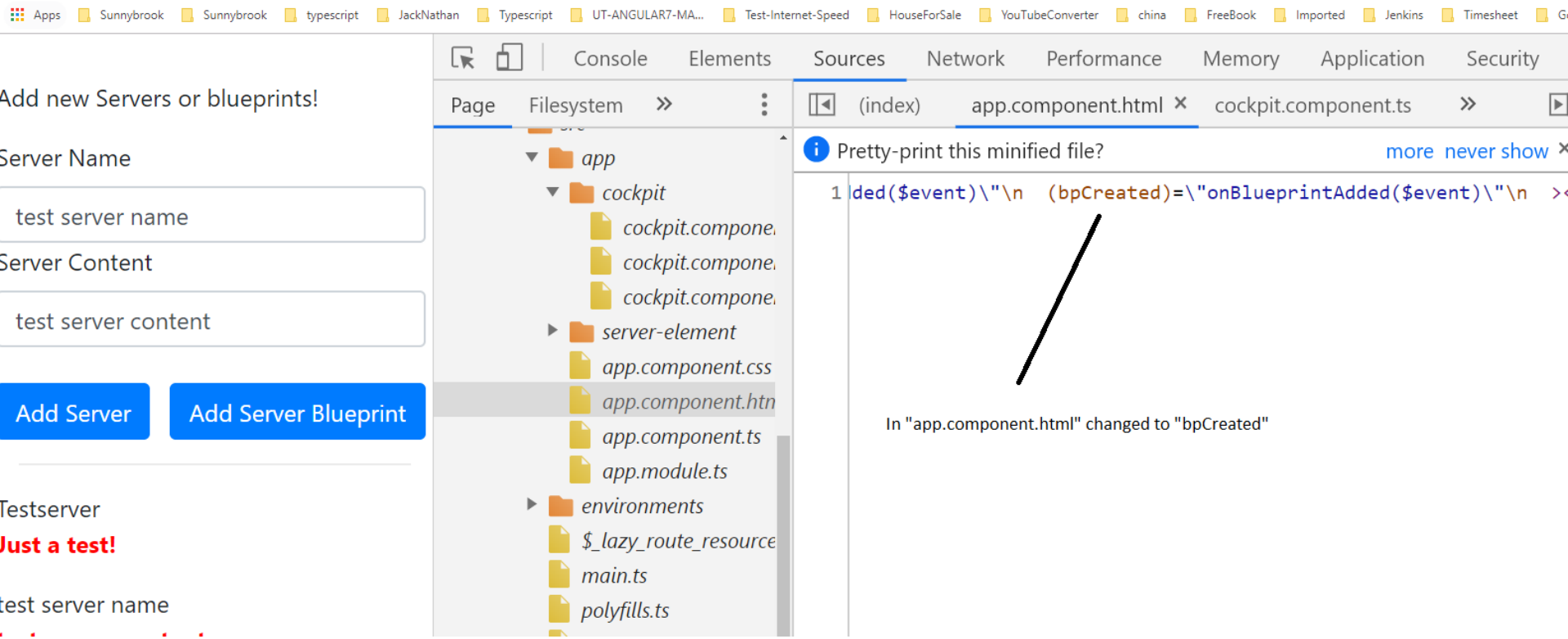
Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)

- Run command:ng serve -o

Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)



Assigning an Alias to Custom Events(example4-component-and-databinding-Version6)



Custom Property and Event Binding Summary

- Let me quickly summarize what we've learned thus far because it really is important this component communication is such a key feature and with an input and the ability to make your properties bind them all from outside from the parent component using this component. And the same for at output which allows parent components using this component to listen to your own events which you carried with you even the emitter. These are such important features in your app.