



3416-003 Web Development Using Angular 8 and TypeScript Module 5: Components and Databinding DeepDived



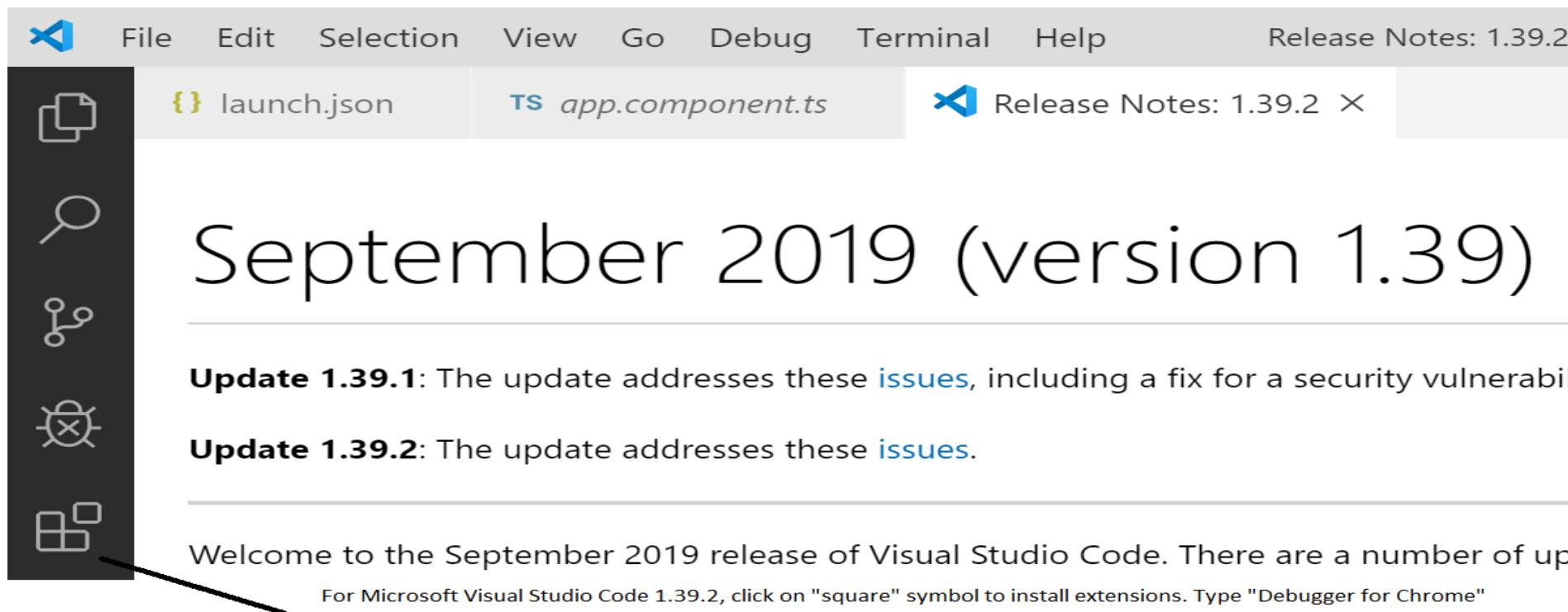
UNIVERSITY OF TORONTO
SCHOOL OF CONTINUING STUDIES

Module 1: Section 1

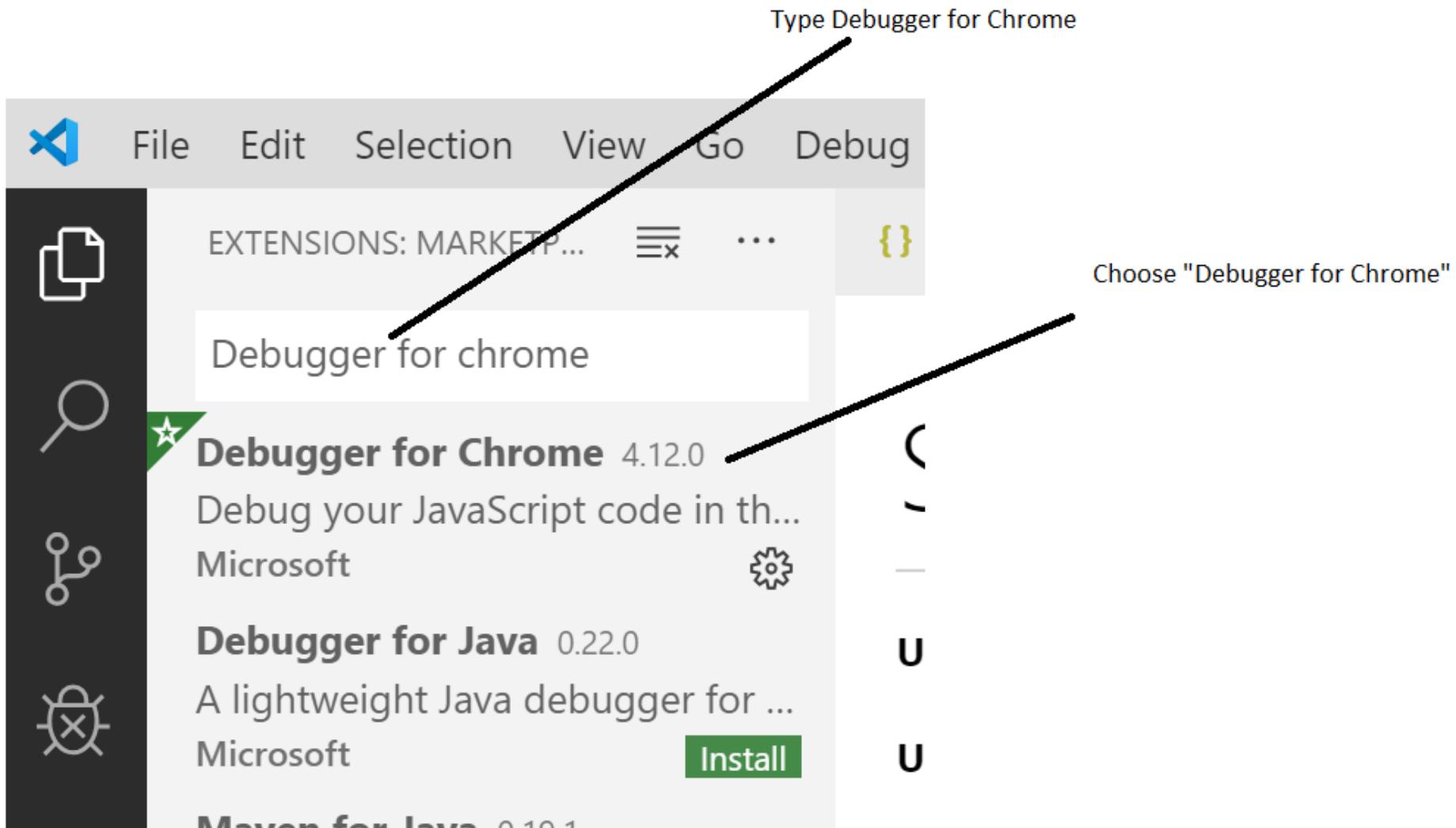
What Is This Course About

Debugger for Chrome

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



Debugger for Chrome



Debugger for Chrome

File Edit Selection View Go Debug Terminal Help Extension: Debugger for Chrome - debugging - Visual S

EXTENSIONS: MARKETP... ⚡ ... { } launch.json  Release Notes: 1.39.2  Extension: Debugger for chrome

Debugger for chrome

Debugger for Chrome 4.12.0 Microsoft | ✓ Uninstalled  ms

Debug your JavaScript code in th...
Debugger for Java 0.22.0
A lightweight Java debugger for ...
Microsoft 

Debugger for Java 0.22.0 Microsoft | ✓ Uninstalled  Click "Install".

Maven for Java 0.19.1 Manage Maven projects, execute...
Microsoft 

 Debugger for Chrome Microsoft | ↗ 4,185,661 | ★★★★
Debug your JavaScript code in the Chrome bro...
Details Contributions Changelog

Debugger for Chrome

The screenshot shows the Microsoft Store page for the "Debugger for Chrome" extension by Microsoft. At the top, there are tabs for "launch.json" and "Release Notes: 1.39.2". The main title is "Extension: Debugger for Chrome". A note says "After clicking "install" becomes "Uninstall"" with a cursor icon over the "install" button. The extension icon is a circular logo with red, green, and yellow segments. The title "Debugger for Chrome" is displayed prominently, followed by the developer "Microsoft" and the download count "4,188,383". The rating is 4.5 stars. Below the title, the description reads "Debug your JavaScript code in the Chrome browser, or any". At the bottom, there are "Disable" and "Uninstall" buttons, and a note stating "This extension is enabled globally".

{} launch.json  Release Notes: 1.39.2  Extension: Debugger for Chrome X

After clicking "install" becomes "Uninstall".

Debugger for Chrome

msjsdiag.debug

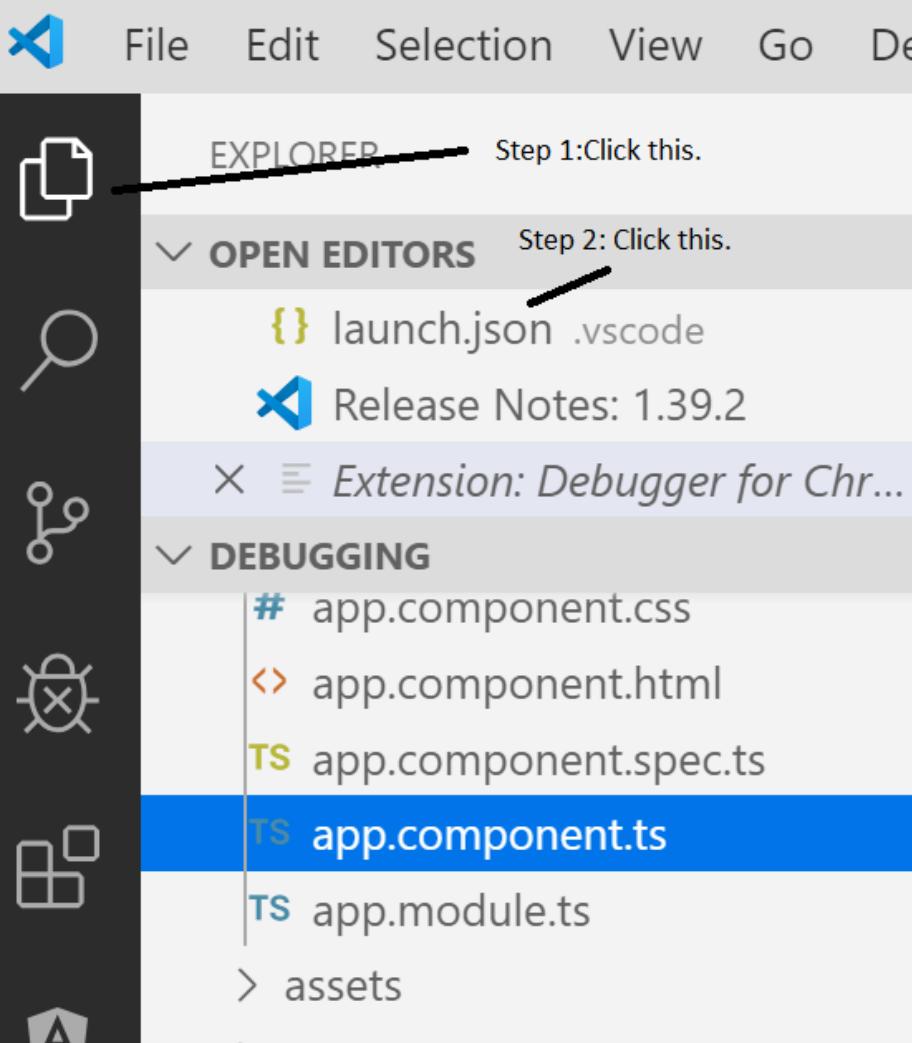
Microsoft | 4,188,383 | ★★★★☆ | Repo

Debug your JavaScript code in the Chrome browser, or any

[Disable](#) ▾ [Uninstall](#) *This extension is enabled globally.*

[Details](#) [Contributions](#) [Changelog](#)

Debugger for Chrome



The screenshot shows the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, Help, and Extension: Debugger for Chrome. The left sidebar has icons for Explorer, Open Editors, Search, Symbols, and others. The Explorer section shows the 'OPEN EDITORS' view with a 'launch.json' file and a 'Release Notes: 1.39.2' entry. Below it is an 'Extension: Debugger for Chr...' entry. The 'DEBUGGING' section lists files: app.component.css, app.component.html, app.component.spec.ts, app.component.ts (which is highlighted in blue), and app.module.ts. An arrow points from the text 'Step 1: Click this.' to the 'Extension: Debugger for Chr...' entry. Another arrow points from the text 'Step 2: Click this.' to the 'Release Notes: 1.39.2' entry.

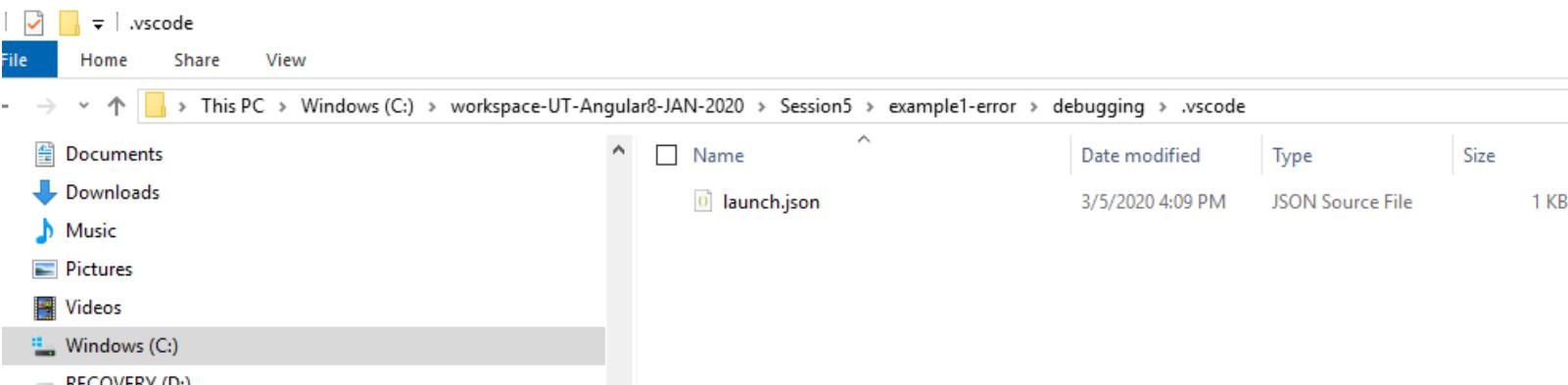
Debugger for Chrome Microsoft | 

Debug your JavaScript with the Debugger for Chrome extension. It provides a comprehensive set of tools for debugging your code, including breakpoints, step-by-step execution, and inspection of variables and call stacks. The extension is compatible with all major browsers and integrates seamlessly with Visual Studio Code.

Disable ▾ **Uninstall**

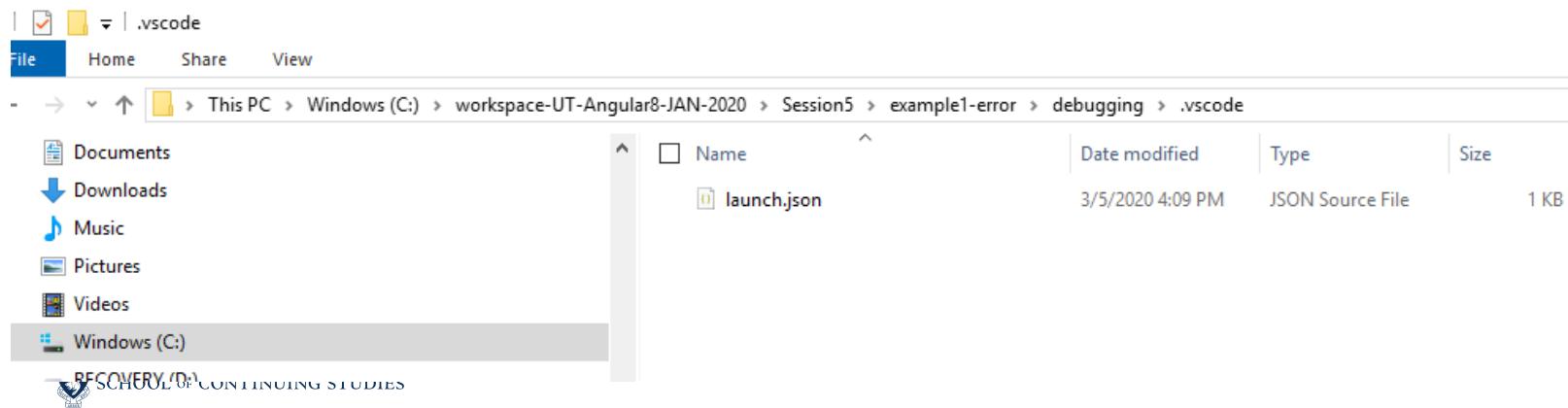
Debugger for Chrome

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



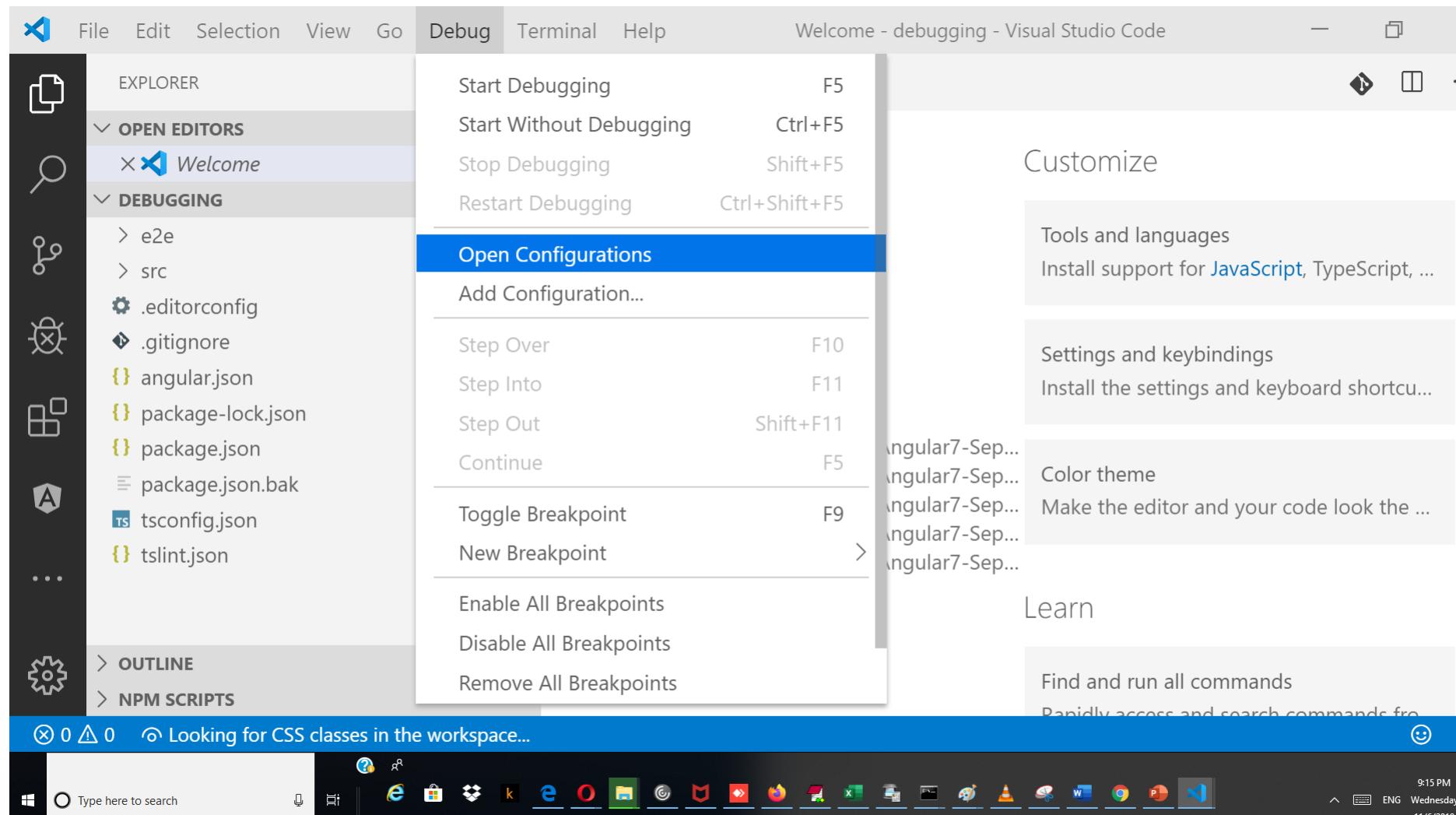
Debugger for Chrome

- 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



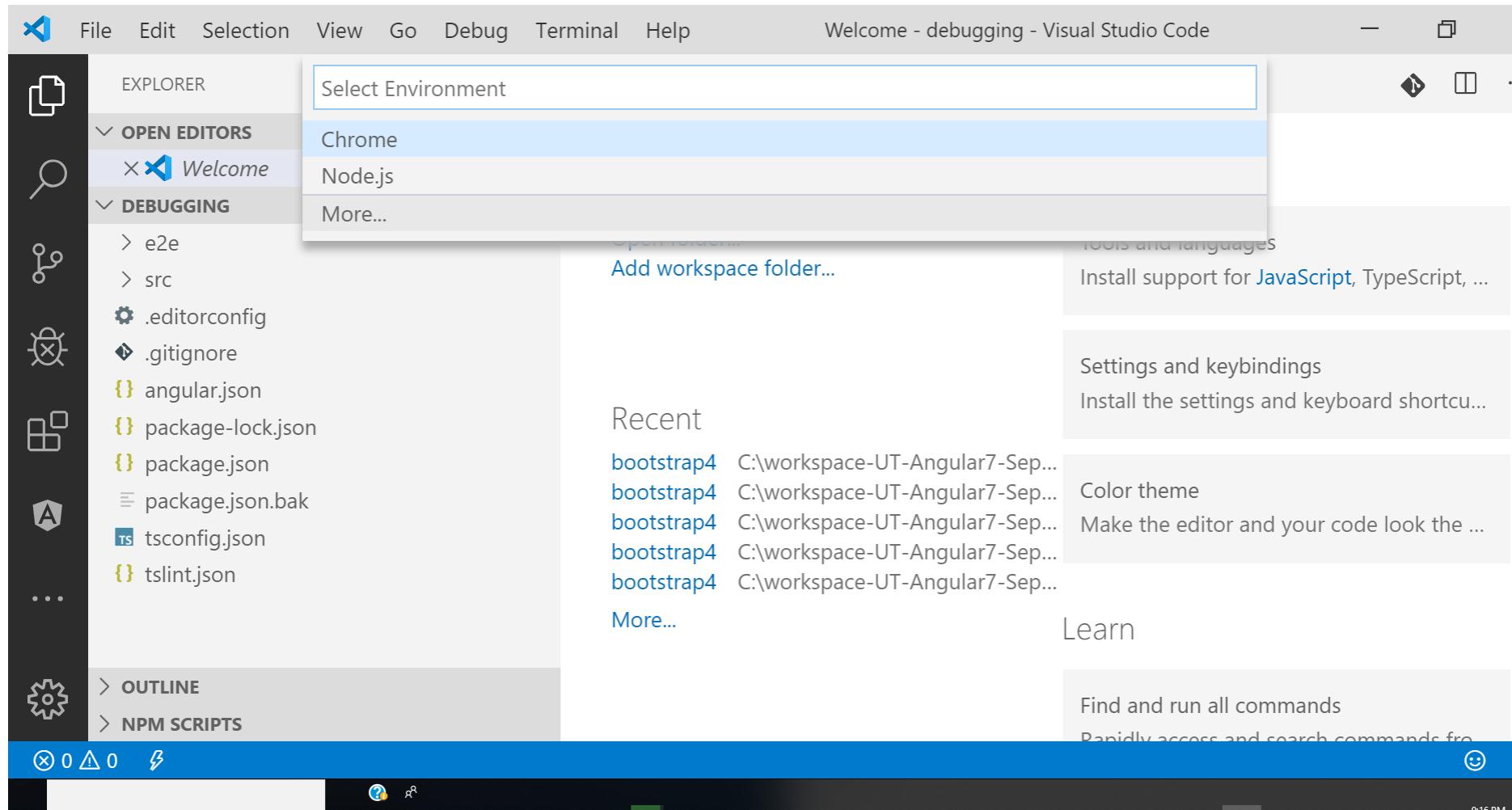
Debugger for Chrome

- **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

The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** launch.json - node_modules - Visual Studio Code.
- EXPLORER Panel:** Shows the project structure with **NODE_MODULES** expanded, containing .bin, .vscode, and several angular-related packages: @angular, @angular-devkit, @babel, @ngtools, and @schematics. The file **{} launch.json** is selected and highlighted with a blue bar.
- Editor Area:** The file **{} launch.json** is open. The code is as follows:

```
// Use IntelliSense to learn about possible attributes.  
// Hover to view descriptions of existing attributes.  
// For more information, visit: https://go.microsoft.com/  
"version": "0.2.0",  
"configurations": [  
  {  
    "type": "chrome",  
    "request": "launch",  
    "name": "Launch Chrome against localhost",  
    "url": "http://localhost:8080",  
    "webRoot": "${workspaceFolder}"  
  }  
]
```

A yellow vertical bar highlights the opening brace of the configurations object. A black arrow points from the text "Change port to 4200" to the number 4200 in the code.

Debugger for Chrome

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

The screenshot shows the Visual Studio Code interface. The top menu bar includes File, Edit, Selection, View, Go, Debug, Terminal, and Help. The title bar indicates the current file is "launch.json - node_modules -".

The Explorer pane on the left shows a tree structure of "NODE_MODULES" containing ".bin", ".vscode", and several "@angular" packages. The file "launch.json" is selected and highlighted.

The Editor pane on the right displays the JSON configuration for launching Chrome. The code is numbered from 1 to 13. The configuration defines a launch target for 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=830302&version=0.2.0
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     }
  ]
```

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
```

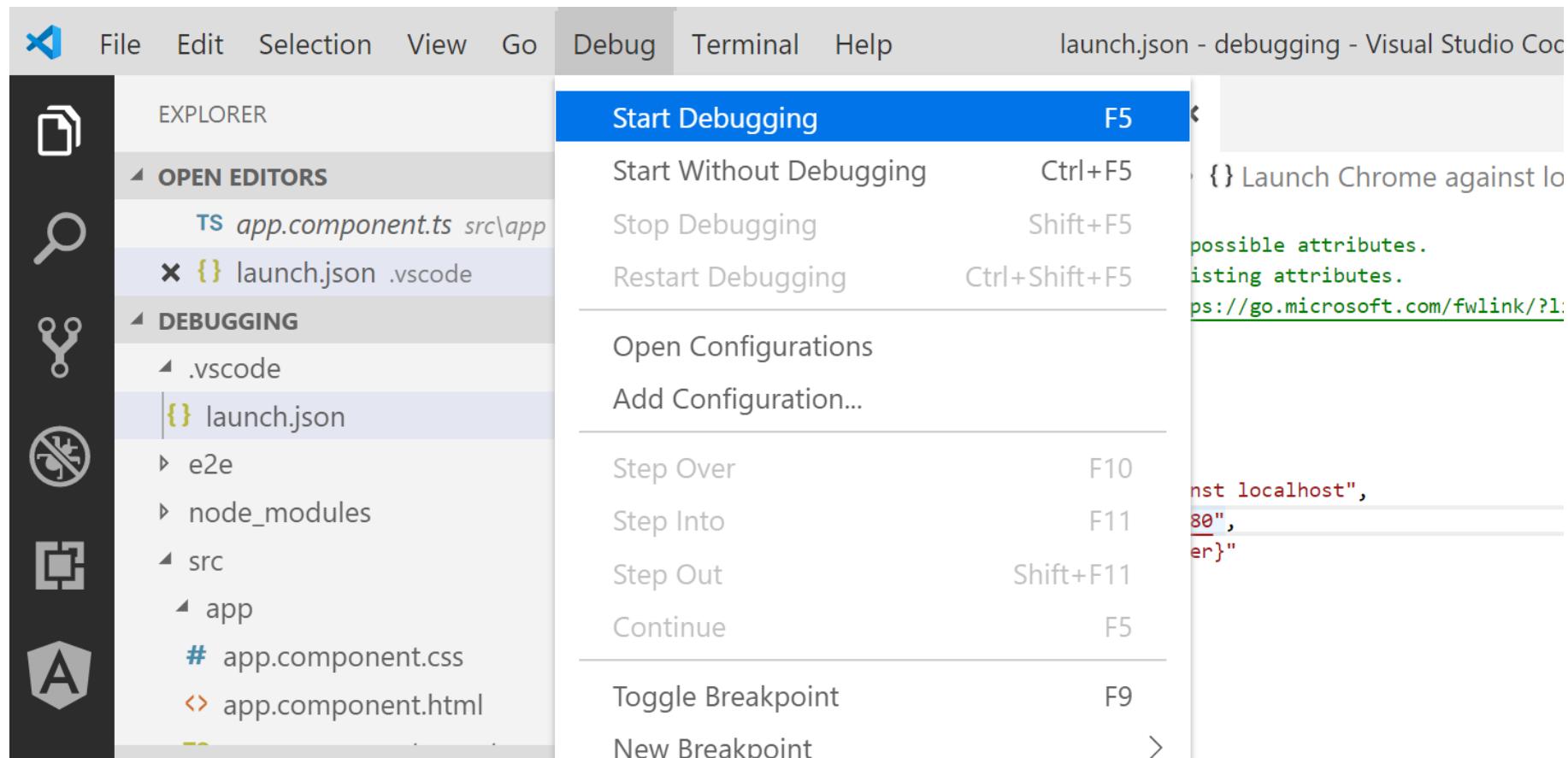
The screenshot shows a browser window with the title "My Servers". A blue button labeled "Add Server" is visible on the left. A black arrow points from this button to the "Console" tab in the DevTools, indicating that clicking it will trigger an error. The DevTools console output is as follows:

```
Angular is running in the development mode. Call enableProdMode() to enable the production mode. core.js:168
✖ ▶ ERROR TypeError: Cannot read property 'push' of undefined AppComponent.html
    at AppComponent.push../src/app/app.component.ts.AppComponent.onAddServer (app.component.ts:12)
    at Object.eval [as handleEvent] (AppComponent.html:5)
    at handleEvent (core.js:23107)
    at callWithDebugContext (core.js:24177)
    at Object.debugHandleEvent [as handleEvent] (core.js:23904)
    at dispatchEvent (core.js:20556)
    at core.js:21003
    at HTMLButtonElement.<anonymous> (platform-browser.js:993)
    at ZoneDelegate.push../node_modules/zone.js/dist/zone.js.ZoneDelegate.invokeTask (zone.js:423)
    at Object.onInvokeTask (core.js:17290)

✖ ▶ ERROR CONTEXT ▶ DebugContext_ {view: {...}, nodeIndex: 5, nodeDef: {...}, elDef: {...}, elView: {...}} AppComponent.html
> |
```

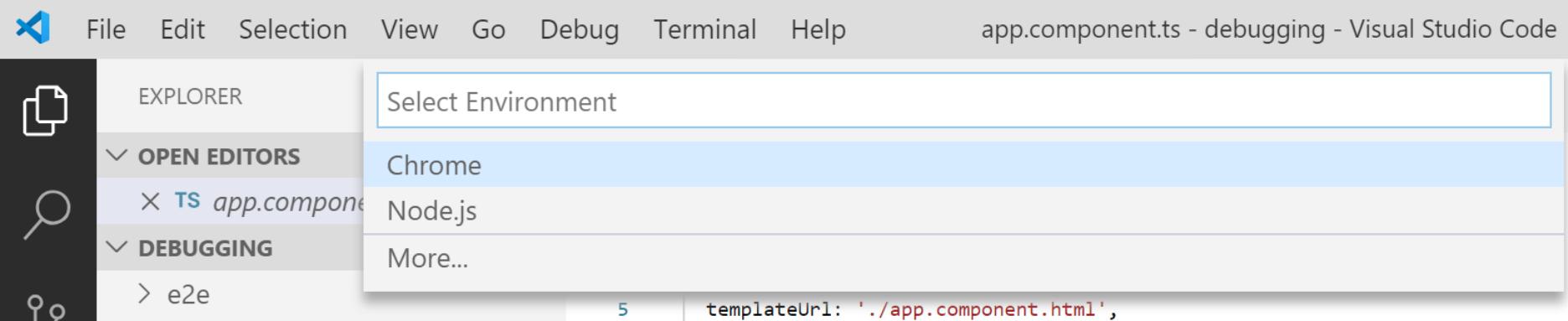
When click "Add Server" get the error in console

Debugger for Chrome

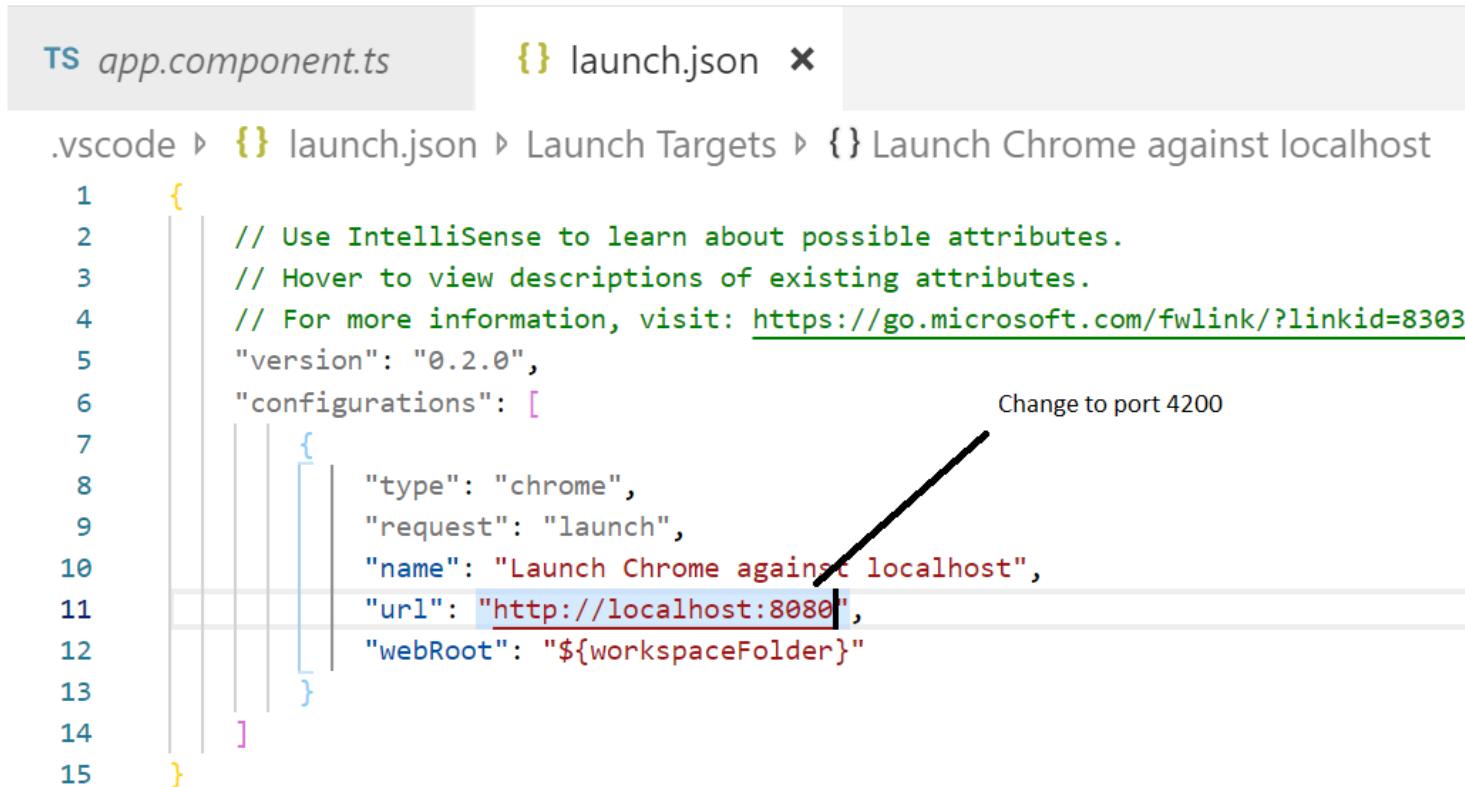


Debugger for Chrome

- Choose “chrome”:



Debugger for Chrome

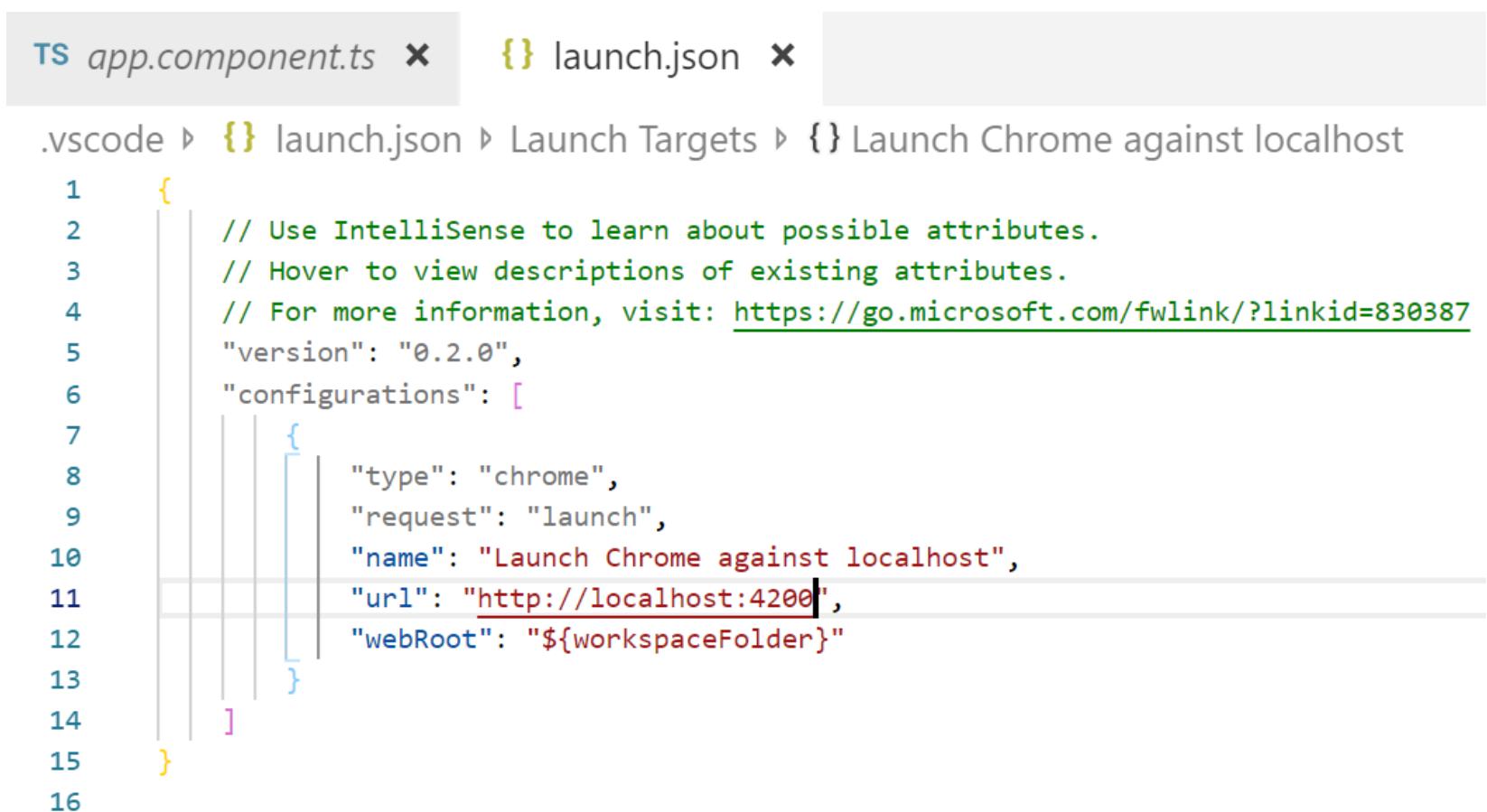


TS app.component.ts {} launch.json x

.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=83038
5      "version": "0.2.0",
6      "configurations": [
7          {
8              "type": "chrome",
9              "request": "launch",
10             "name": "Launch Chrome against localhost",
11             "url": "http://localhost:8080", Change to port 4200
12             "webRoot": "${workspaceFolder}"
13         }
14     ]
15 }
```

Debugger for Chrome



The screenshot shows the VS Code interface with two open files: `app.component.ts` and `launch.json`. The `launch.json` file is the active tab, displaying a JSON configuration for launching a browser. The code is as follows:

```
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
```

The `url` field is highlighted with a red rectangle, indicating it is the current selection.

Debugger for Chrome

- Right click to place breakpoints.

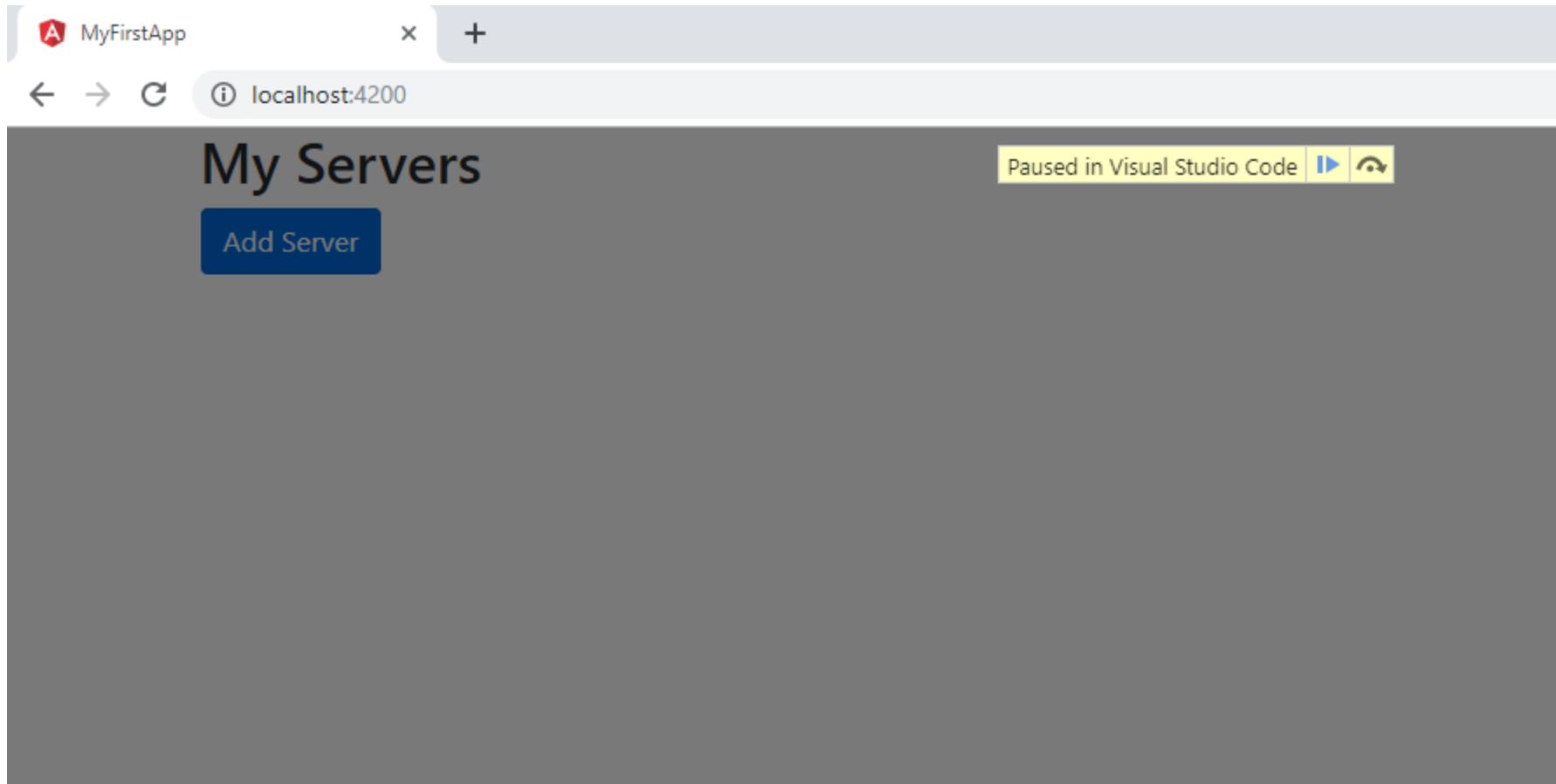
The screenshot shows the Visual Studio Code interface with the 'Debug' tab selected. The code editor displays the file 'app.component.ts' with the following content:

```
src > app > TS app.component.ts > ...
6 |   styleUrls: ['./app.component.css']
7 |
8 | export class AppComponent {
9 |   servers;
10|   ...
11|   onAddServer() {
12|     this.servers.push('Another Server');
13|   }
14|
15|   onRemoveServer(id: number) {
16|     const position = id + 1;
17|     this.servers.splice(position, 1);
18|   }
19| }
```

A red circle marks a breakpoint on line 12. A red circle also marks a breakpoint on line 16. A tooltip 'Left click to create breakpoint' is shown near the line 12 breakpoint. The status bar at the bottom of the screen displays the message: 'Angular is running in the development mode. Call enableProdMode() to enable the production mode.'

Debugger for Chrome

- Click “Add Server”:



Debugger for Chrome

- Get “servers” is undefined.

```
    /      })
8   export class AppComponent {
9     servers;
10    ...
11    onAddServ undefined
12    this.servers.push('Another Server');
13  }
14
```

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

The screenshot shows the VS Code interface with the following details:

- EXPLORER** pane on the left:
 - OPEN EDITORS**: Shows a file named `app.component.ts` with a red error icon and the number 3.
 - DEBUGGING**: Shows configurations for `.vscode`, `e2e`, and `node_modules`.
 - src**:
 - app**:
 - `# app.component.css`
 - `< app.component.html`
 - `TS app.component.spec.ts`
- Editor** pane on the right:
 - Title bar**: `TS app.component.ts`
 - File path**: `src > app > TS app.component.ts`
 - Content**:

```
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   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
```

A red squiggly underline is under the line `//servers = [];`. A tooltip says `Add this line`.

How to debug through Google Chrome

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

The screenshot shows the Google Chrome DevTools interface with the 'Sources' tab selected. On the left, the file tree shows the project structure under 'localhost:4200'. The 'app.component.ts' file is open in the main pane. A black arrow points from the bottom-left towards the file tree. A callout bubble with the text 'Click "." folder inside webpack' points to the root folder icon in the file tree.

```
+ selector: 'app-root',
5 templateUrl: './app.component.html',
6 styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   servers: string[] = [];
10
11   onAddServer() {
12     this.servers.push('Another Server');
13   }
14
15   onRemoveServer(id: number) {
16     const position = id + 1;
17     this.servers.splice(position, 1);
18   }
19 }
20
```

Click "." folder inside webpack

How to debug through Google Chrome

My Servers

Add Server

Another Server

The screenshot shows the Google Chrome DevTools interface with the 'Sources' tab selected. The left sidebar displays a tree view of files under 'webpack://'. The 'app.component.ts' file is selected in the tree, and its code is shown in the main pane. The code defines an AppComponent class with methods for adding and removing servers. A tooltip indicates that the code is source-mapped from 'main.js'. The status bar at the bottom shows 'Line 12, Column 10'.

```
4 selector: 'app-root',
5 templateUrl: './app.component.html',
6 styleUrls: ['./app.component.css']
7 }
8 export class AppComponent {
9   servers: string[] = [];
10
11   onAddServer() {
12     this.servers.push('Another Server');
13   }
14
15   onRemoveServer(id: number) {
16     const position = id + 1;
17     this.servers.splice(position, 1);
18   }
19 }
```

{ } Line 12, Column 10
(source mapped from main.js)

How to debug through Google Chrome

The screenshot shows the Google Chrome DevTools interface with the 'Sources' tab selected. A file tree on the left lists files like 'index.html', 'main.js', 'polyfills.js', etc. The main area shows the code for 'app.component.ts'. A red arrow points from the text 'CLICK here to deactivate breakpoints' to a small icon in the top right corner of the code editor.

CLICK here to deactivate
breakpoints

Console Elements Sources Network Performance Memory Application Security Audits Augury

filesystem Overrides » :

index.html
localhost:4200
node_modules/bootstrap
src
(index)
main.js
polyfills.js
runtime.js
styles.js
vendor.js
Augury
://
:backpack:/:

app.component.ts

```
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   servers: string[] = [];
10
11   onAddServer() {
12     this.servers.push('Another Server');
13   }
14
15   onRemoveServer(id: number) {
16     const position = id + 1;
17     this.servers.splice(position, 1);
18   }
19 }
```

▶ Watch
▼ Call Stack
Not paused
▼ Scope
Not paused
▼ Breakpoints

- app.component.ts:8
export class AppComponent {
- app.component.ts:9
servers: string[] = [];
- app.component.ts:12
this.servers.push('Another Server');
- app.component.ts:16

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

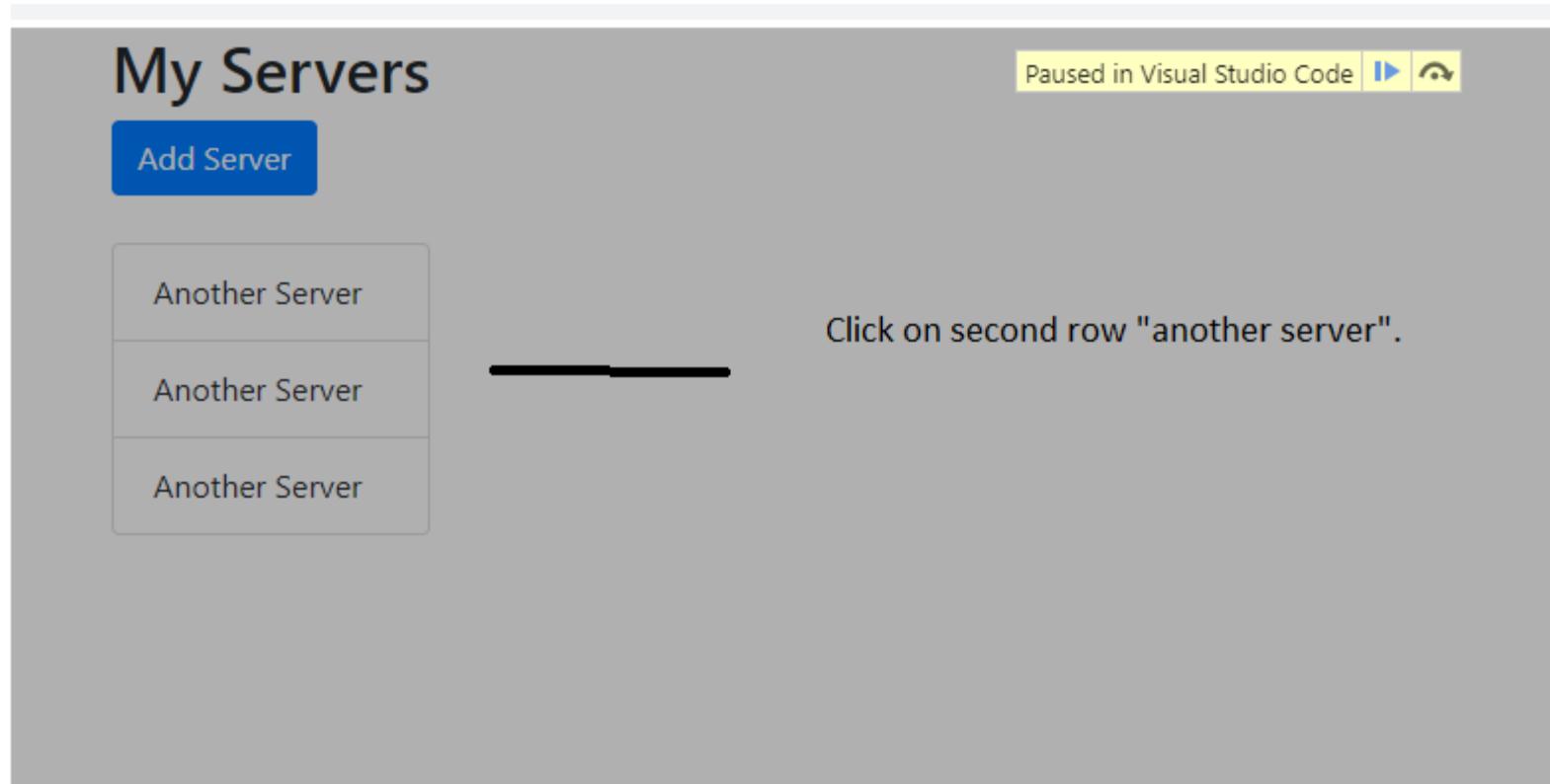
```
1  <div class="container">
2    <div class="row">
3      <div class="col-xs-12">
4        <h2>My Servers</h2>
5        <button class="btn btn-primary" (click)="onAddServer()">Add Server</button>
6        <br><br>
7        <ul class="list-group">
8          <li
9            class="list-group-item"
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.

My Servers

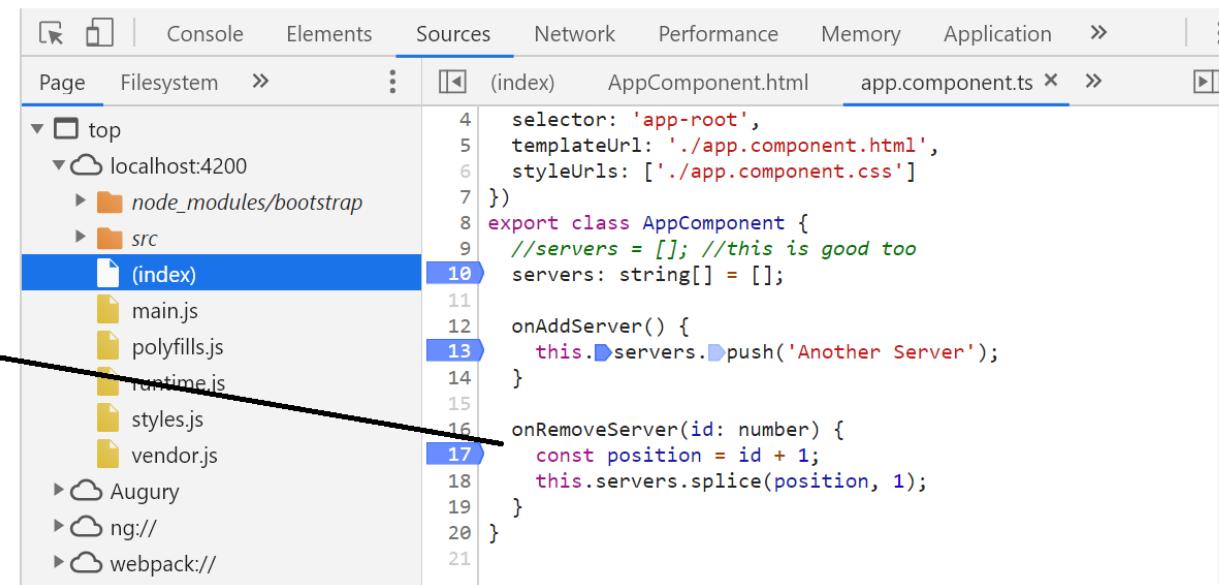
Add Server

Another Server

Another Server

Another Server

position 3 means 3 items in
the array. Second
parameter 1 means remove
1 item from the item



```
4 selector: 'app-root',
5 templateUrl: './app.component.html',
6 styleUrls: ['./app.component.css']
7 }
8 export class AppComponent {
9 //servers = []; //this is good too
10 servers: string[] = [];
11
12 onAddServer() {
13     this.servers.push('Another Server');
14 }
15
16 onRemoveServer(id: number) {
17     const position = id + 1;
18     this.servers.splice(position, 1);
19 }
20 }
21 }
```

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

My Servers

Add Server

Another Server

Another Server

example2-solution(how to debug using browser)

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

The screenshot shows the Chrome DevTools Sources tab open to the file `app.component.ts`. The left sidebar displays the project structure:

- ng://
- webpack://
 - (webpack)-dev-server
 - (webpack)
- .
- node_modules
- src
 - app
 - app.component.css
 - app.component.html
 - app.component.ts
 - app.module.ts
 - environments
- \$ lazy route resource la

The `app.component.ts` file content is shown:

```
templateUrl: './app.component.html',
styleUrls: ['./app.component.css']
})
export class AppComponent {
//servers = []; //this is good too
servers: string[] = [];

onAddServer() {
    this.servers.push('Another Server');
}

onremoveServer(id: number) { id = 1
    const position = id + 1; position = 2
    this.servers.splice(position, 1);
}
}
```

Line 13 (`this.servers.push('Another Server');`) has a blue background highlight. Line 17 (`this.servers.splice(position, 1);`) has a light blue background highlight. A black arrow points from the text "Put breakpoint on removeServer" to the word "splice" in line 17. The code editor shows a cursor at the end of line 17.

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.

The screenshot shows a browser window with developer tools open. The left side displays a list of servers: "Another Server", "Another Server", and "Another Server". A callout points from the third "Another Server" entry to the code in the debugger, explaining that position 2 means it's looking at the 3rd element but there are only 2 elements in the array. The code in the debugger highlights the problematic line: `this.servers.splice(position, 1);`. The developer tools tabs at the top are: Paused in debugger, Console, Elements, Sources, Network, Performance, Memory, and App.

My Servers

Paused in debugger

Add Server

Another Server

Another Server

Another Server

Position 2 means it is looking at 3rd element but there is only 2 elements. 1st element start with index 0. 2nd element start with index 1.

Console Elements Sources Network Performance Memory App

Page Filesystem >:

top

localhost:4200

- node_modules/bootstrap
- src
- (index)
- main.js
- polyfills.js
- runtime.js
- styles.js
- vendor.js

Augury

ng://

webpack://

app.component.ts x main.ts >

```
import { Component } from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  //servers = []; //this is good too
  servers: string[] = [];
  onAddServer() {
    this.servers.push('Another Server')
  }
  onRemoveServer(id: number) {
    const position = id + 1;
    this.servers.splice(position, 1);
  }
}
```

example2-solution(how to debug using browser)

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

EXPLORER

OPEN EDITORS

- TS app.component.ts...
- <> app.component.html...

DEBUGGING

- # app.component.css
- <> app.component.html
- TS app.component.sp...
- TS app.component.ts
- TS app.module.ts
- > assets
- > environments
- browserslist
- ★ favicon.ico
- <> index.html

TS app.component.ts × <> app.component.html

```
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   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
17    //const position = id + 1;
18    const position = id;
19    this.servers.splice(position, 1);
20  }
}
```

example2-solution(how to debug using browser)

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

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows files like app.component.css, app.component.html, app.module.ts, assets, environments, browserslist, favicon.ico.
- OPEN EDITORS**: Shows app.component.ts and app.component.html.
- DEBUGGING**: Shows app.component.css, app.component.html, app.component.sp..., app.component.ts, app.module.ts.

The **app.component.ts** editor has the following code:

```
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   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
17    //const position = id + 1;
18    const position = id;
19    this.servers.splice(position, 1);
  
```

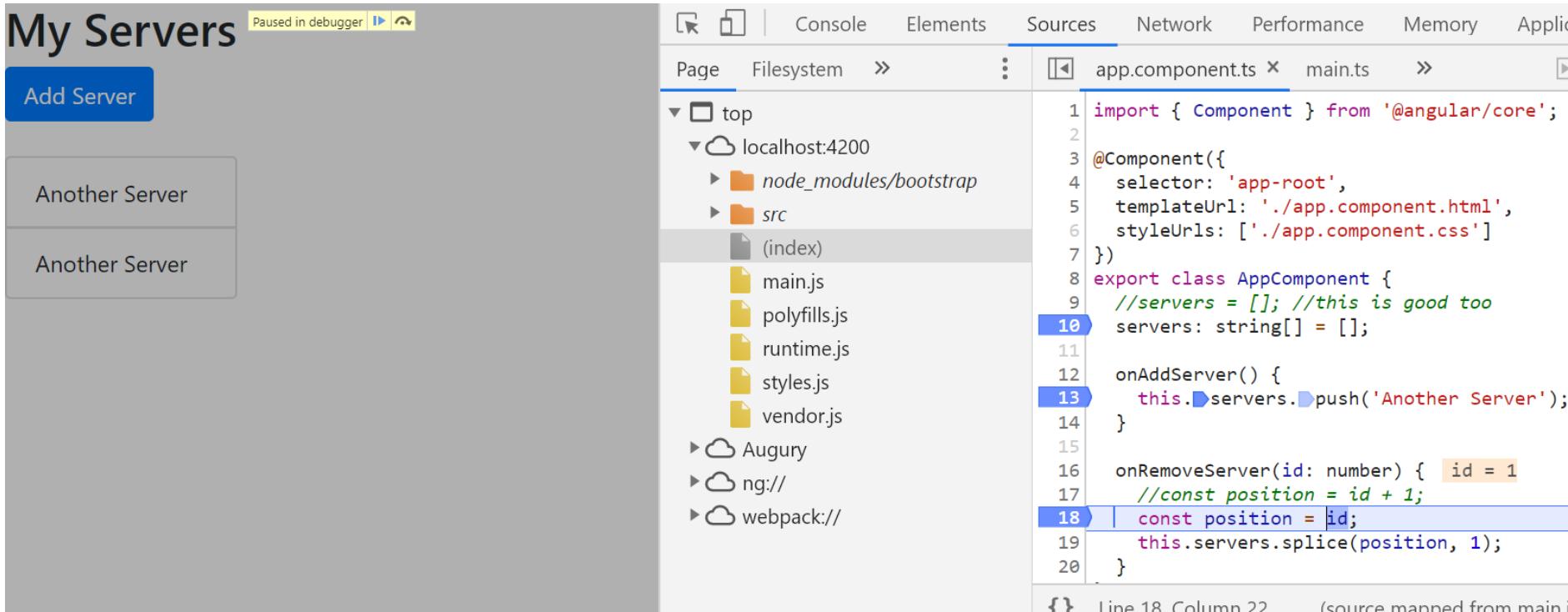
Annotations and comments in the code:

- Line 9: `//servers = [];` is underlined in red with a comment: `//this is good too`.
- Line 13: A red circle is at the start of the `onAddServer()` method.
- Line 17: `//const position = id + 1;` is underlined in red with a comment: `Delete the '1'.` A black line points from this annotation to the `1` in `id + 1`.
- Line 17: A red circle is at the start of the `onRemoveServer()` method.
- Line 19: `this.servers.splice(position, 1);` is underlined in purple.

SCHOOL OF CONTINUING STUDIES

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”.

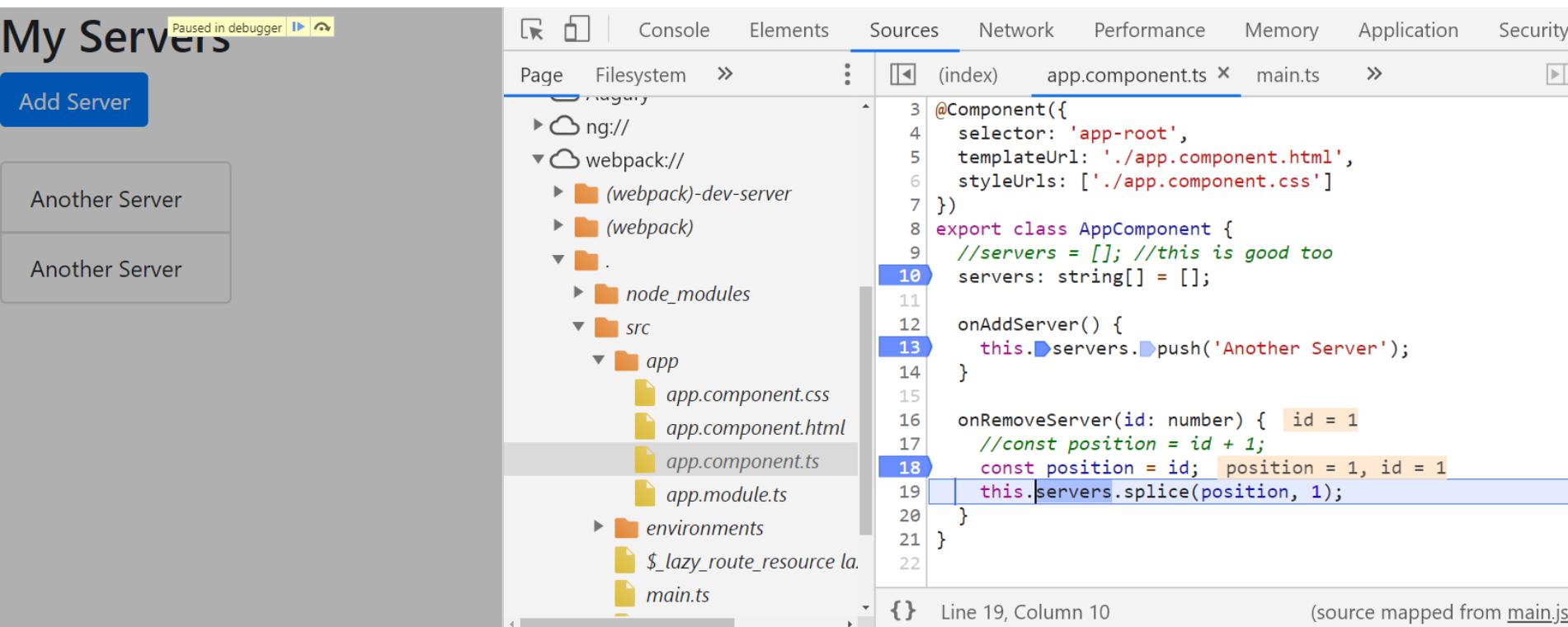


The screenshot shows a browser window with developer tools open. The left pane displays a list of servers under 'My Servers' with two 'Another Server' entries highlighted. The right pane shows the 'Sources' tab of the developer tools, with the file 'app.component.ts' selected. The code is as follows:

```
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   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
17    //const position = id + 1;
18    const position = id;
19    this.servers.splice(position, 1);
20  }
}
```

The line 'this.servers.push('Another Server')' is highlighted with a blue selection, indicating it is the current line of execution. The status bar at the bottom of the developer tools indicates 'Line 18, Column 22 (source mapped from main.js)'.

example2-solution(how to debug using browser)



The screenshot shows a browser developer tools interface with the "Sources" tab selected. The left sidebar displays a file tree with the following structure:

- ng://
- webpack://
 - (webpack)-dev-server
 - (webpack)
 - .
 - node_modules
 - src
 - app
 - app.component.css
 - app.component.html
 - app.component.ts
 - app.module.ts
 - environments
 - \$_lazy_route_resource.la.
 - main.ts

The "app.component.ts" file is open in the main editor area. The code is as follows:

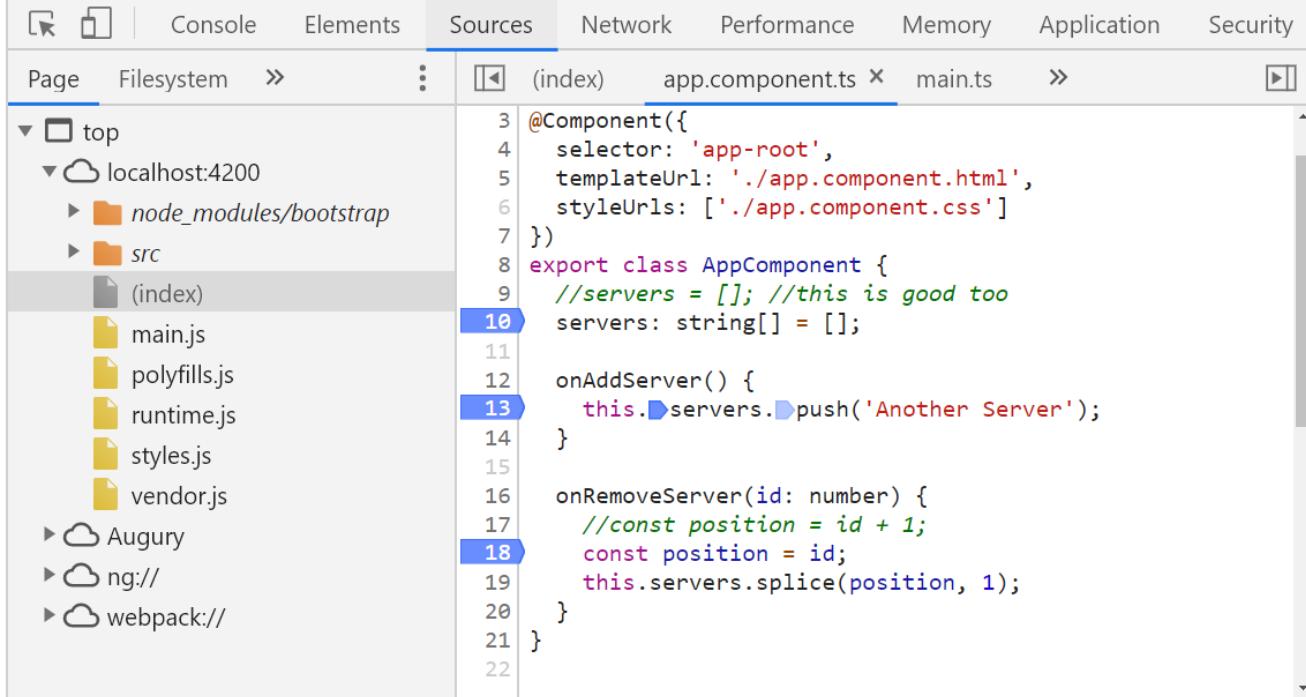
```
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
17    //const position = id + 1;
18    const position = id; position = 1, id = 1
19    this.servers.splice(position, 1);
20  }
21}
```

A blue highlight is present on line 18, specifically on the line `this.servers.splice(position, 1);`. The status bar at the bottom indicates "Line 19, Column 10". A note at the bottom right says "(source mapped from main.js)".

example2-solution(how to debug using browser)

My Servers

Add Server



The screenshot shows a browser's developer tools open to the Sources tab. The left sidebar displays a file tree for the application. Under the 'localhost:4200' section, the 'src' folder is expanded, showing files like 'main.js', 'polyfills.js', 'runtime.js', 'styles.js', and 'vendor.js'. The '(index)' file is selected in the tree. The right pane shows the code for 'app.component.ts'. The code defines an Angular component named 'AppComponent' with a selector of 'app-root'. It has a template URL pointing to 'app.component.html' and style URLs pointing to 'app.component.css'. The component has a property 'servers' of type 'string[]' initialized to an empty array. It includes two methods: 'onAddServer' which adds a new server to the array, and 'onRemoveServer' which removes a server from the array at a specified index. Lines 10, 13, and 18 are highlighted with blue boxes.

```
3 @Component({
4   selector: 'app-root',
5   templateUrl: './app.component.html',
6   styleUrls: ['./app.component.css']
7 })
8 export class AppComponent {
9   //servers = []; //this is good too
10  servers: string[] = [];
11
12  onAddServer() {
13    this.servers.push('Another Server');
14  }
15
16  onRemoveServer(id: number) {
17    //const position = id + 1;
18    const position = id;
19    this.servers.splice(position, 1);
20  }
21}
22}
```

Augury

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

Extensions

2 of 2 extensions



Augury

Offered by: Rangle.io

Extends the Developer Tools, adding tools for debugging and profiling /

★★★★★ 215 Developer Tools

Add to Chrome

Click "Add to Chrome"



Augury (Canary Build)

Offered by: Rangle.io

Alpha release version for Augury. Updated frequently with experimental

★★★★★ 2 Developer Tools

Add to Chrome

Augury

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

core/search/augury

chrome web store

augury

Extensions

It can:

Add "Augury (Canary Build)"?

Read and change all your data on the websites you visit

Add extension Cancel

jadeite1000@gmail.com

2 of 2 extensions

« Home

Extensions Themes

Features

Runs Offline By Google Free Available for Android Works with Google Drive

Augury

Offered by: Rangle.io

Extends the Developer Tools, adding tools for debugging and profiling /

Click "Add Extension"

Add to Chrome

Augury (Canary Build)

Offered by: Rangle.io

Alpha release version for Augury. Updated frequently with experimental fea

Checking...

★★★★★ 2 Developer Tools

★★★★★ & up

Click "Add Extension"

Augury (Canary Build)

Offered by: Rangle.io

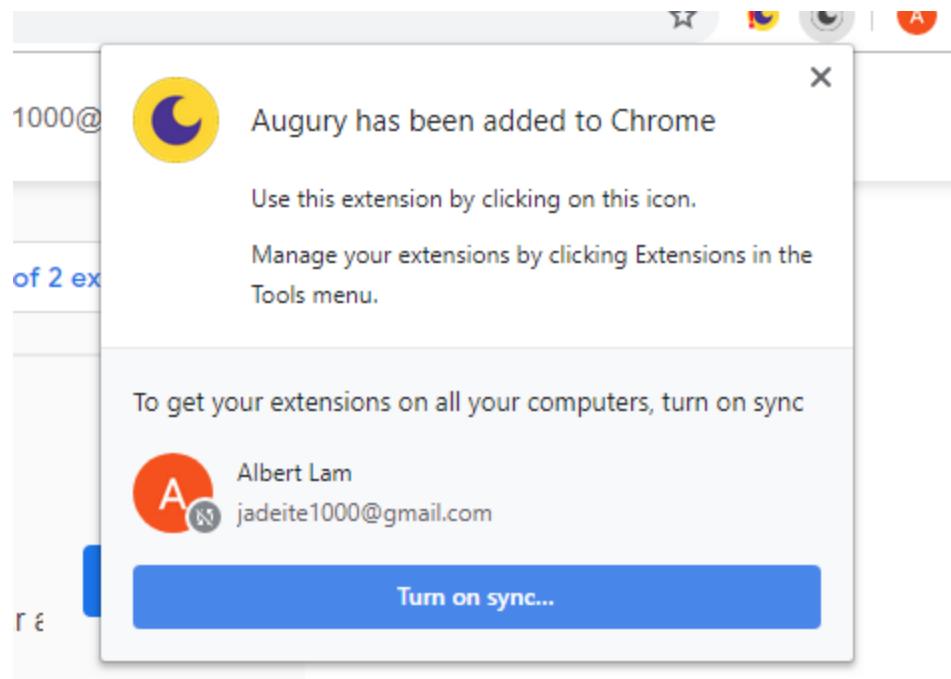
Alpha release version for Augury. Updated frequently with experimental fea

Checking...

★★★★★ 2 Developer Tools

Augury

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



Click "Turn on sync".

Augury

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

localhost:4200

Google Chrome isn't your default browser Set as default

My Servers

Add Server

Another Server

Another Server

Another Server

Another Server

AppComponent

Console Elements Sources Network Performance Memory Application Augury

Angular Version: 7.2.15

AppComponent (View Source) (\$\$el in Console)

Change Detection: Default

State

servers: Array[4]

- 0: Another Server
- 1: Another Server
- 2: Another Server
- 3: Another Server

Click "Augury" tab.
Click "Properties" tab and click on "servers".

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

The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help.
- Title Bar:** app.component.html - component-interaction - Visual Studio Code.
- Left Sidebar (EXPLORER):**
 - OPEN EDITORS: app.component.html (highlighted)
 - COMPONENT-INTERACTION:
 - > e2e
 - > node_modules
 - > src
 - > app
 - app.component.html (highlighted)
 - 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
 - Editor Area:** The code for app.component.html is displayed, showing examples of component communication. The code includes sections for "parent-to-child", "parent-to-child-setter", "parent-to-child-on-changes", and "child-to-parent".

```
<h1 id="top">Component Communication Cookbook</h1>
<a href="#parent-to-child">Pass data from parent to child with input binding ("Heroes")</a>
<a href="#parent-to-child-setter">Intercept input property changes with a setter ("Name")</a>
<a href="#parent-to-child-on-changes">Intercept input property changes with <i>ngOnChanges</i> ("Version")</a>
<a href="#child-to-parent">Parent listens for child event ("Colonize Universe")</a>
<div id="parent-to-child">
  <app-hero-parent></app-hero-parent>
</div>
<a href="#top" class="to-top">Back to Top</a>
<hr>
<div id="parent-to-child-setter">
  <app-name-parent></app-name-parent>
</div>
<a href="#top" class="to-top">Back to Top</a>
<hr>
<div id="parent-to-child-on-changes">
  <app-version-parent></app-version-parent>
</div>
<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).

EXPLORER

OPEN EDITORS

hero-child.component.ts 1

COMPONENT-...

app.component.ts
app.module.ts
astronaut.component.ts
countdown-parent.compo...
countdown-timer.compon...
hero-child.component.ts 1
hero-parent.component.ts
hero.ts

TS hero-child.component.ts ×

src > app > TS hero-child.component.ts > HeroChildComponent > hero

```
1 import { Component, Input } from '@angular/core';
2
3 import { Hero } from './hero';
4
5 @Component({
6   selector: 'app-hero-child',
7   template:
8     `<h3>{{hero.name}} says:</h3>
9     <p>I, {{hero.name}}, am at your service, {{masterName}}.</p>
10    `
11 })
12 export class HeroChildComponent {
13   @Input() hero: Hero;
14   @Input('master') masterName: string;
15 }
16
```



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

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists several files: app.component.ts, app.module.ts, astronaut.component.ts, countdown-parent.compo..., countdown-timer.compon..., hero-child.component.ts (which is currently selected and has a blue background), hero-parent.component.ts, and hero.ts. The Open Editors sidebar shows the hero-child.component.ts file is open. The main editor area displays the code for hero-child.component.ts:

```
TS hero-child.component.ts ×
src > app > TS hero-child.component.ts > HeroChildComponent > hero
1 import { Component, Input } from '@angular/core';
2
3 import { Hero } from './hero';
4
5 @Component({
6   selector: 'app-hero-child',
7   template:
8     `
9       <h3>{{hero.name}} says:</h3>
10      <p>I, {{hero.name}}, am at your service, {{masterName}}.</p>
11    `
12 })
13 export class HeroChildComponent {
14   @Input() hero: Hero;
15   @Input('master') masterName: string;
16 }
```

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

The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar lists several files under 'OPEN EDITORS' and 'COMPONENT-INTERACTION'. In the center, the 'hero.ts' file is open in the editor. The code defines a class 'Hero' with a string property 'name' and an array 'HEROES' containing three hero objects.

```
TS hero.ts      ×  
src > app > TS hero.ts > [ HEROES  
1  export class Hero {  
2    name: string;  
3  }  
4  
5  export const HEROES = [  
6    {name: 'Dr IQ'},  
7    {name: 'Magneta'},  
8    {name: 'Bombasto'}  
9  ];  
10
```

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.
-

Master controls 3 names

"Dr IQ"

"<no name set>"

"Bombasto"

[Back to Top](#)

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 }
```

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

EXPLORER

OPEN EDITORS

src > app > **version-child.component.ts** ×

COMPONENT-INTERACTION

- countdown.directive.component.ts
- hero-child.component.ts
- hero-parent.component.ts
- hero.ts
- mission.service.ts
- missioncontrol.component.ts
- name-child.component.ts
- name-parent.component.ts
- version-child.component.ts**
- version-parent.component.ts
- voter.component.ts
- votetaker.component.ts

> assets

> environments

OUTLINE

NPM SCRIPTS

```
/* tslint:disable:forin */
import { Component, Input, OnChanges, SimpleChange } from '@angular/core';
@Component({
  selector: 'app-version-child',
  template: `
    <h3>Version {{major}}.{{minor}}</h3>
    <h4>Change log:</h4>
    <ul>
      | <li *ngFor="let change of changeLog">{{change}}</li>
    </ul>
  `
})
export class VersionChildComponent implements OnChanges {
  @Input() major: number;
  @Input() minor: number;
  changeLog: string[] = [];
  ngOnChanges(changes: {[propKey: string]: SimpleChange}) {
    let log: string[] = [];
    for (let propName in changes) {
      let changedProp = changes[propName];
      let to = JSON.stringify(changedProp.currentValue);
      if (changedProp.isFirstChange()) {
        log.push(`Initial value of ${propName} set to ${to}`);
      } else {
        let from = JSON.stringify(changedProp.previousValue);
        log.push(`${propName} changed from ${from} to ${to}`);
      }
    }
  }
}
```

Implements onChanges interface

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

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

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER**: Shows a tree view of files. The "version-parent.component.ts" file is currently selected, indicated by a blue bar at the bottom of its row.
- version-child.component.ts**: An editor tab showing the template for the child component, which includes an "app-version-child" element with attributes [major] and [minor].
- version-parent.component.ts**: An editor tab showing the code for the parent component. It defines a "VersionParentComponent" class with properties "major" and "minor", and methods "newMinor()" and "newMajor()".

```
src > app > TS version-parent.component.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-version-parent',
5   template: `
6     <h2>Source code version</h2>
7     <button (click)="newMinor()">New minor version</button>
8     <button (click)="newMajor()">New major version</button>
9     <app-version-child [major]="major" [minor]="minor"></app-version-child>
10   `
11 })
12 export class VersionParentComponent {
13   major = 1;
14   minor = 23;
15
16   newMinor() {
17     this.minor++;
18   }
19
20   newMajor() {
21     this.major++;
22     this.minor = 0;
23   }
24 }
```

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

Click "New minor version".

~~Source code version~~

New minor version New major version

Version 1.23

Change log:

Before clicking "New minor version".

- Initial value of major set to 1, Initial value of minor set to 23

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

The screenshot shows the Chrome DevTools interface with the Sources tab selected. The left sidebar shows the project structure under 'app'. The main pane displays the code for 'version-child.component.ts'. The code defines a component with a selector and template, and implements the `OnChanges` interface. The `ngOnChanges` method takes an array of changes, where each change is an object with a `propKey` and a `SimpleChange` value. The method logs the initial value or change for each propKey. The right sidebar shows the 'Watch' panel with the `changes` object expanded, showing its `minor` and `changedProp` properties.

```
5   selector: 'app-version-child',
6   template:
7     <h3>Version {{major}}.{{minor}}</h3>
8     <h4>Change log:</h4>
9     <ul>
10       <li *ngFor="let change of changeLog">{{change}}</li>
11     </ul>
12   )
13 }
14 export class VersionChildComponent implements OnChanges {
15   @Input() major: number;
16   @Input() minor: number;
17   changeLog: string[] = [];
18
19   ngOnChanges(changes: {[propKey: string]: SimpleChange}) {
20     let log: string[] = [];
21     for (let propName in changes) {
22       changes = {minor: SimpleChange};
23       let changedProp = changes[propName];
24       let to = JSON.stringify(changedProp.currentValue);
25       if (changedProp.isFirstChange()) {
26         log.push(`Initial value of ${propName} set to ${to}`);
27       } else {
28         let from = JSON.stringify(changedProp.previousValue);
29         log.push(`${propName} changed from ${from} to ${to}`);
30       }
31     }
32     this.changeLog.push(log.join(', '));
33   }
34 }
```

Debugger paused

Watch

changes: Object

- minor: SimpleChange
 - currentValue: 24
 - firstChange: false
 - previousValue: 23
 - __proto__: Object
 - __proto__: Object
- propName: "minor"
- minor.previousValue: <not available>

changedProp: SimpleChange

- currentValue: 24
- firstChange: false
- previousValue: 23
- __proto__: Object
- __proto__: Object
- to: "24"

Call Stack

- ngOnChanges
- version-child.component.ts:24
- checkAndUpdateDirectiveInline
- core.js:21905

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

Source code version

New minor version

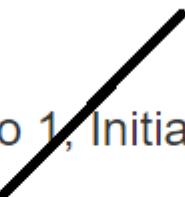
New major version

Version 1.24

After clicking "New minor version".

Change log:

- Initial value of major set to 1, Initial value of minor set to 23
- minor changed from 23 to 24



[Back to Top](#)

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

- Click “New Major Version”

Source code version

Before Click "Major version".

New minor version New major version

Version 1.24

Change log:

- Initial value of major set to 1, Initial value of minor set to 23
- minor changed from 23 to 24



[Back to Top](#)

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

The screenshot shows a browser developer tools debugger interface. On the left, there's a sidebar with navigation links like "source code version", "New minor version", "New major version", "version 1.24", "change log", and "Should mankind colonize the universe?". Below these are buttons for "Agree: 0, Disagree: 0" and "Disco". The main area is a debugger window with tabs for Page, Filesystem, Sources, Network, Performance, Memory, Application, Security, and more. The Sources tab is active, showing the file "version-child.component.ts". The code is as follows:

```
<h4>Change log:</h4>
<ul>
  <li *ngFor="let change of changeLog">{{change}}
</ul>
}
export class VersionChildComponent implements OnInit {
  @Input() major: number;
  @Input() minor: number;
  changeLog: string[] = [];

  ngOnChanges(changes: {[propKey: string]: SimpleChange}) {
    let log: string[] = [];
    for (let propName in changes) {
      changes = changes[propName];
      let changedProp = changes[ propName ];
      let to = JSON.stringify(changedProp.currentValue);
      if (changedProp.isFirstChange()) {
        log.push(`Initial value of ${propName} is ${to}`);
      } else {
        let from = JSON.stringify(changedProp.previousValue);
        log.push(`${propName} changed from ${from} to ${to}`);
      }
      this.changeLog.push(log.join(', '));
    }
  }
}
```

The right side of the debugger shows the execution context with the variable "changes" expanded. It contains two properties: "major" and "minor", both of which are instances of "SimpleChange". The "major" object has properties: "currentValue: 2", "firstChange: false", and "previousValue: 1". The "minor" object has properties: "currentValue: 0", "firstChange: false", and "previousValue: 24". The "changedProp" object also has properties: "currentValue: 2", "firstChange: false", and "previousValue: 1". The "this.changeLog" array contains two entries: "Initial value of major is 2" and "minor changed from 24 to 2".

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

- After clicking “New Major Version”:
-

Source code version

New minor version New major version

Version 2.0

Change log:

- Initial value of major set to 1, Initial value of minor set to 23
- minor changed from 23 to 24
- major changed from 1 to 2, minor changed from 24 to 0

[Back to Top](#)

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-Flow\Version1.txt

Component Interaction(Parent listens for child event).

- The child is voter.component.ts

The screenshot shows the Angular IDE interface. On the left, the Explorer sidebar lists various component files. In the center, two code editors are open: 'version-child.component.ts' and 'voter.component.ts'. The 'voter.component.ts' editor is active and displays the following TypeScript code:

```
src > app > TS voter.component.ts > ...
1 import { Component, EventEmitter, Input, Output } from '@angular/core';
2
3 @Component({
4   selector: 'app-voter',
5   template: `
6     <h4>{{name}}</h4>
7     <button (click)="vote(true)" [disabled]="didVote">Agree</button>
8     <button (click)="vote(false)" [disabled]="didVote">Disagree</button>
9   `
10})
11export class VoterComponent {
12  @Input() name: string;
13  @Output() voted = new EventEmitter<boolean>();
14  didVote = false;
15
16  vote(agreed: boolean) {
17    this.voted.emit(agreed);
18    this.didVote = true;
19    console.log("voter this.name:" + this.name);
20    console.log("voter this.voted:" + this.voted);
21  }
22}
23
```

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 votetaker.component.ts

The screenshot shows an IDE interface with two tabs open: 'version-child.component.ts' and 'votetaker.component.ts'. The 'votetaker.component.ts' tab is active, displaying the following code:

```
src > app > TS votetaker.component.ts > VoteTakerComponent
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-vote-taker',
5   template: `
6     <h2>Should mankind colonize the Universe?</h2>
7     <h3>Agree: {{agreed}}, Disagree: {{disagreed}}</h3>
8     <app-voter *ngFor="let voter of voters"
9       [name]="voter"
10      (voted)="onVoted($event)">
11    </app-voter>
12  `
13})
14 export class VoteTakerComponent {
15   agreed = 0;
16   disagreed = 0;
17   voters = ['Narco', 'Celeritas', 'Bombasto'];
18
19   onVoted(agreed: boolean) {
20     agreed ? this.agreed++ : this.disagreed++;
21     console.log("votetaker agreed:" + agreed);
22   }
23}
```

The 'OPEN EDITORS' sidebar on the left lists several components and services, with 'votetaker.component.ts' currently selected. The 'EXPLORER' sidebar shows the project structure under 'src > app'.

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).

Should mankind colonize the Universe?

Agree: 0, Disagree: 0

Before clicking "Agree"

Narco



Agree Disagree

Celeritas

Agree Disagree

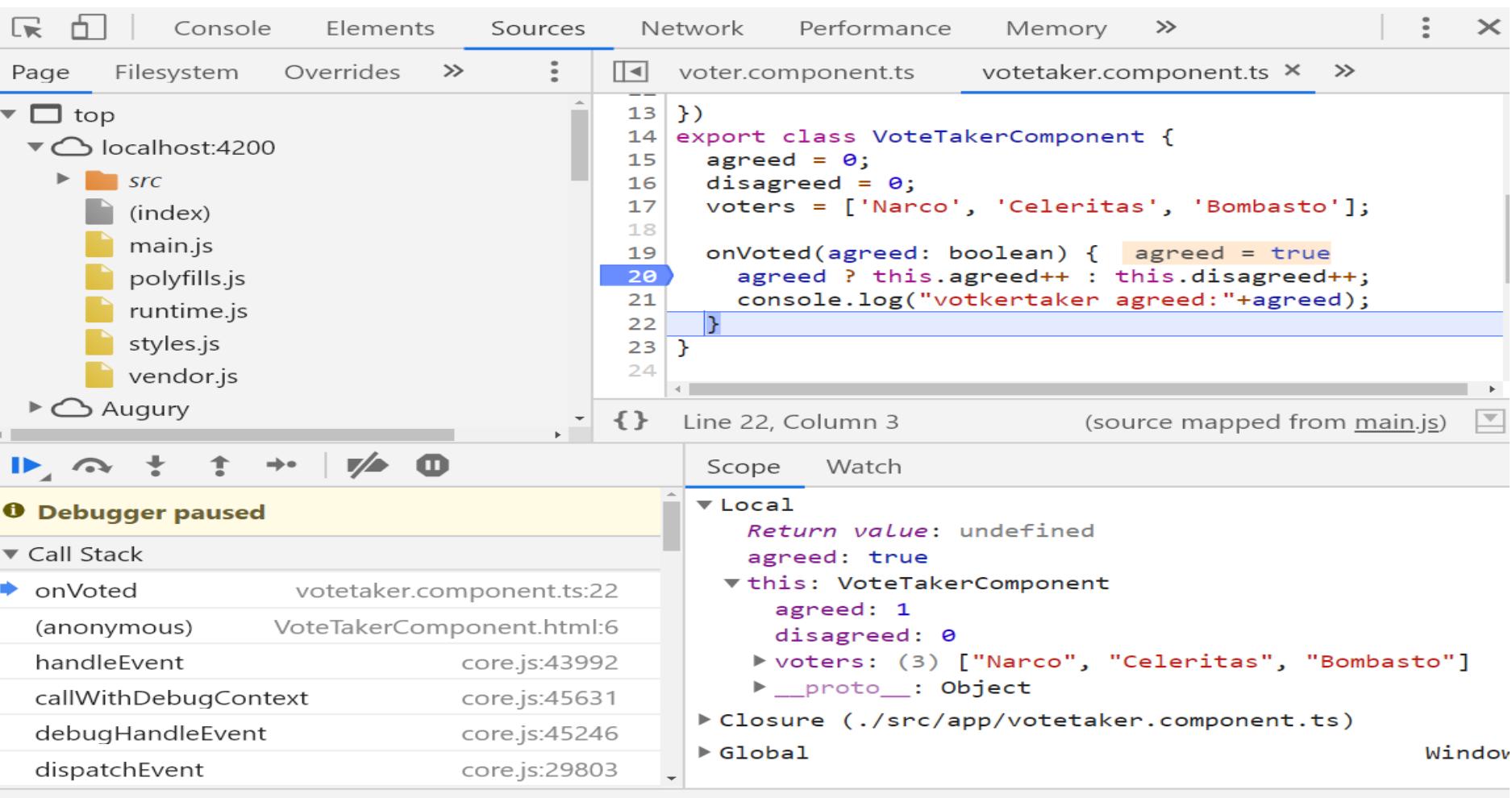
Bombasto

Agree Disagree

[Back to Top](#)

Component Interaction(Parent listens for child event).

- Parent Component votetaker.component.ts



```
13 })
14 export class VoteTakerComponent {
15   agreed = 0;
16   disagreed = 0;
17   voters = ['Narco', 'Celeritas', 'Bombasto'];
18
19   onVoted(agreed: boolean) { agreed = true
20     agreed ? this.agreed++ : this.disagreed++;
21     console.log("votetaker agreed:" + agreed);
22   }
23 }
```

Line 22, Column 3 (source mapped from main.js)

Scope Watch

Local

- Return value: undefined
- agreed: true
- this: VoteTakerComponent
 - agreed: 1
 - disagreed: 0
 - voters: (3) ["Narco", "Celeritas", "Bombasto"]
 - __proto__: Object
- Closure (../src/app/votetaker.component.ts)
- Global

Call Stack

- onVoted votetaker.component.ts:22
- (anonymous) VoteTakerComponent.html:6
- handleEvent core.js:43992
- callWithDebugContext core.js:45631
- debugHandleEvent core.js:45246
- dispatchEvent core.js:29803

Component Interaction(Parent listens for child event).

- Child Component voter.component.ts

The screenshot shows the Chrome DevTools interface with the Sources tab selected. On the left, the file tree shows the project structure with files like index.html, main.js, polyfills.js, runtime.js, styles.js, and vendor.js. The voter.component.ts file is open, and a breakpoint is set at line 20, column 5, which corresponds to the `this.voted.emit(agreed);` line. The code editor highlights this line. The Call Stack on the left shows the execution flow through voter.component.ts, VoterComponent.html, core.js, and other Angular framework files. The Watch pane on the right shows the properties of the `this.voted` EventEmitter instance, including `closed`, `hasError`, `isStopped`, `observers`, `thrownError`, `_isAsync`, `_isScalar`, and `_proto_`.

```
voter.component.ts
11 export class VoterComponent {
12   @Input() name: string;
13   @Output() voted = new EventEmitter<boolean>();
14   didVote = false;
15
16   vote(agreed: boolean) { agreed = true
17     this.voted.emit(agreed);
18     this.didVote = true;
19     console.log("voter this.name:" + this.name);
20     console.log("voter this.voted:" + this.voted);
21   }
22 }
```

Line 20, Column 5 (source mapped from main.js)

Scope Watch

`this.voted: EventEmitter`

- `closed: false`
- `hasError: false`
- `isStopped: false`
- `observers: [Subscriber]`
- `thrownError: null`
- `_isAsync: false`
- `_isScalar: false`
- `_proto_: Subject`

Component Interaction(Parent listens for child event).

- After clicking “Agree”.

The screenshot shows a browser window with the URL `localhost:4200/#parent-to-child-setter`. The page content includes a header "Version 1.23", a "Change log" section with a bullet point about initial values, a "Back to Top" link, and three user profiles: "Narco", "Celeritas", and "Bombasto". Each profile has an "Agree" and "Disagree" button. A callout from the Narco profile points to the "didVote" variable in the code, which is highlighted in red. The developer tools sidebar shows the file structure and the code for `voter.component.ts`.

```
Component({
  selector: 'app-voter',
  template: `
    <h4>{{name}}</h4>
    <button (click)="vote(true)" [disabled]="didVote">Agree</button>
    <button (click)="vote(false)" [disabled]="didVote">Disagree</button>
  `
})
export class VoterComponent {
  @Input() name: string;
  @Output() voted = new EventEmitter<boolean>();
  didVote = false;

  vote(agreed: boolean) {
    this.voted.emit(agreed);
    this.didVote = true;
    console.log("voter this.name:" + this.name);
    console.log("voter this.voted:" + this.voted);
  }
}
```

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-Flow\Version1.txt
- Look at C:\AlbertLam\Angular8-UT-Jan-2020\Session5\Flow\ComponentInteraction-Parent-Interacts-With-Child-Via-Local-Variable-ClickStart--Flow\Version1.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

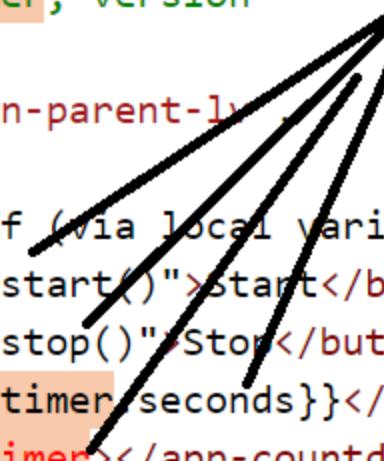
```
- 6  ///// Local variable, #timer
  7  <@Component({
  8    selector: '',
  9    template:
10      <h3>Countdown to Liftoff (via local variable)</h3>
11      <button (click)="timer.start()">Start</button>
12      <button (click)="timer.stop()">Stop</button>
13      <div class="seconds">{{timer.seconds}}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

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).

```
-  
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  })
```

declare timer variable
and used in html template



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:

The screenshot shows a web application interface. At the top, there is a horizontal line separator. Below it, the title "Mission Control" is displayed. Underneath the title, there is a button labeled "Announce mission". Below this button, there are three entries, each consisting of a name followed by a message and a "Confirm" button. The first entry is "Lovell: <no mission announced> Confirm". The second entry is "Swigert: <no mission announced> Confirm". The third entry is "Haise: <no mission announced> Confirm". Below these entries, the word "History" is displayed. At the bottom of the page, there is a link labeled "Back to Top". To the right of the main content area, there is a vertical scroll bar.

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

- Announce Mission:

Mission Control

[Announce mission](#)

Lovell: **Fly to the moon!** [Confirm](#)

Swigert: **Fly to the moon!** [Confirm](#)

Haise: **Fly to the moon!** [Confirm](#)

History

- Mission "Fly to the moon!" announced

[Back to Top](#)

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

- Click First “Confirm”.

Mission Control

[Announce mission](#)

Lovell: **Fly to the moon!** [Confirm](#)

Swigert: **Fly to the moon!** [Confirm](#)

Haise: **Fly to the moon!** [Confirm](#)

History

- Mission "Fly to the moon!" announced
- Lovell confirmed the mission

[Back to Top](#)

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

- Click Second “Confirm”.

Mission Control

[Announce mission](#)

Lovell: **Fly to the moon!** [Confirm](#)

Swigert: **Fly to the moon!** [Confirm](#)

Haise: **Fly to the moon!** [Confirm](#)

History

- Mission "Fly to the moon!" announced
- Lovell confirmed the mission
- Swigert confirmed the mission

[Back to Top](#)

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

- Click Third “Confirm”.

Mission Control

Announce mission

Lovell: **Fly to the moon!**

Swigert: **Fly to the moon!**

Haise: **Fly to the moon!**

History

- Mission "Fly to the moon!" announced
- Lovell confirmed the mission
- Swigert confirmed the mission
- Haise confirmed the mission

[Back to Top](#)

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

lient.ts

TS countdown-parent.component.ts

TS mission.service.ts X

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

```
1 import { Injectable } from '@angular/core';
2 import { Subject } from 'rxjs';
3
4 @Injectable()
5 export class MissionService {
6
7     // Observable string sources
8     private missionAnnouncedSource = new Subject<string>();
9     private missionConfirmedSource = new Subject<string>();
10
11    // Observable string streams
12    missionAnnounced$ = this.missionAnnouncedSource.asObservable();
13    missionConfirmed$ = this.missionConfirmedSource.asObservable();
14
15    // Service message commands
16    announceMission(mission: string) {
17        this.missionAnnouncedSource.next(mission);
18    }
19
20    confirmMission(astronaut: string) {
21        this.missionConfirmedSource.next(astronaut);
22    }
```

Inside mission.service.ts file



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 |
43
```

The MissionControlComponent injects the MissionService instance through its constructor.



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).

```
4 import { Subscription }    from 'rxjs';
5
6 @Component({
7   selector: 'app-astronaut',
8   template: `
9     <p>
10       {{astronaut}}: <strong>{{mission}}</strong>
11       <button
12         (click)="confirm()"
13         [disabled]="!announced || confirmed">
14         Confirm
15       </button>
16     </p> `           This is
17   `                   astronaut.component.ts
18 })
19 export class AstronautComponent implements OnDestroy {
20   @Input() astronaut: string;
21   mission = '<no mission announced>';
22   confirmed = false;
23   announced = false;
24   subscription: Subscription;
25
26   constructor(private missionService: MissionService) {
27     this.subscription = missionService.missionAnnounced$.subscribe(
28       mission => {
29         this.mission = mission;
30         this.announced = true;
31         this.confirmed = false;
32       });
33   }
34
35   confirm() {
36     this.confirmed = true;
37     this.missionService.confirmMission(this.astronaut);
38   }
39
40   ngOnDestroy() {
41     // prevent memory leak when component destroyed
42     this.subscription.unsubscribe();
43   }
44 }
```

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.

Add new Servers or blueprints!

Server Name

Server Content

Add Server

Add Server Blueprint

Testserver

Just a test!



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

The screenshot shows a web-based application interface for managing servers or blueprints. At the top, there are two browser tabs, both labeled "localhost:4200". The left tab shows a header with "Apps" and several folder icons, followed by "Sunnybrook", "typescript", "JackNathan", "Typescript", "UT-ANGULAR7-MA...", and "Test-Internet-Speed". The right tab has a similar structure.

The main content area contains the following elements:

- A heading: "Add new Servers or blueprints!"
- A "Server Name" input field containing "server name1". A large black arrow points from this field towards the "Add Server Blueprint" button.
- A "Server Content" input field containing "server content1". Another large black arrow points from this field towards the "Add Server Blueprint" button.
- Two blue buttons at the bottom: "Add Server" and "Add Server Blueprint".

To the right of the "Add Server Blueprint" button, there is a note: "Type the following and click \"Add Server\" button".

Testserver

Just a test!



Splitting Apps into components

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

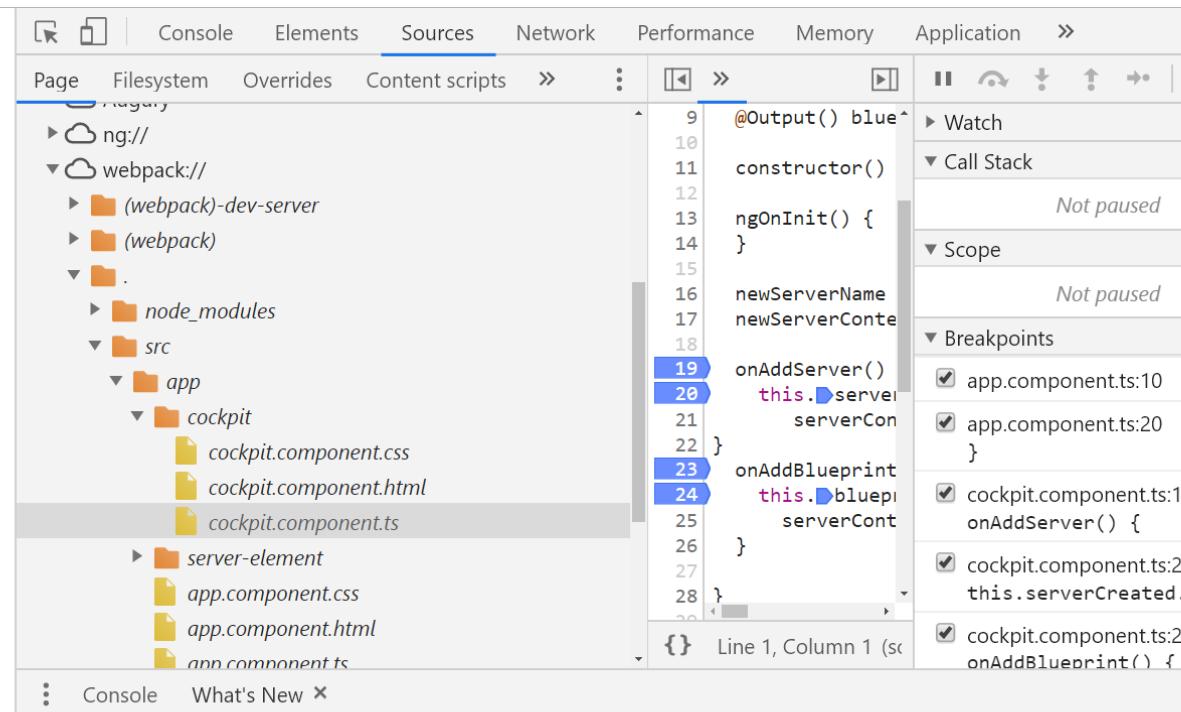
Add new Servers or blueprints!

Server Name

Server Content

Add Server **Add Server Blueprint**

Testserver
Just a test!



The screenshot shows the Chrome DevTools Sources tab open. The left sidebar lists file paths: ng://, webpack://, ., node_modules, src, app, cockpit, server-element, and app. The file cockpit.component.ts is selected and highlighted with a gray background. The right panel displays the code for this file:

```
9  @Output() blue
10 constructor()
11 ngOnInit() {
12     newServerName
13     newServerContent
14 }
15 onAddServer()
16     this.server
17     serverCon
18 }
19 onAddBlueprint()
20     this.bluep
21     serverCont
22 }
23 onAddBlueprint()
24     this.bluep
25     serverCont
26 }
27
28 }
```

The code includes logic for adding servers and blueprints, utilizing the Angular framework's @Output decorator and constructor injection.

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)
```

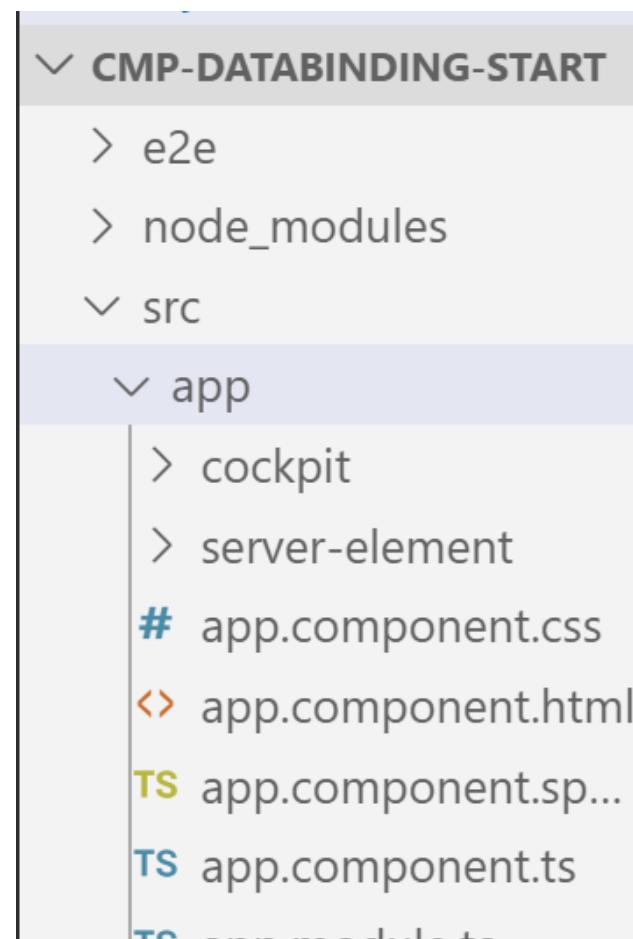
```
CREATE src/app/server-element/server-element.component.ts (300  
bytes)
```

```
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



New file
Open folder...
Add workspace folder...

Recent

debugging C:\workspace-UT-Angular7-Sept-2019\S...
debugging C:\workspace-UT-Angular7-Sept-2019\S...
node_modules C:\workspace-UT-Angular7-Sept-201...
debugging C:\workspace-UT-Angular7-May-2019\s...
bootstrap4 C:\workspace-UT-Angular7-Sept-2019\S...

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 x 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      /* Begin cockpit */
12
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

Code editor showing the migration of code from `app.component.html` to `server-element.component.html`.

The code is as follows:

```
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-->| Move from app.component.html to server-
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-->
21         </div>
22       </div>
23     </div>
24   </div>
```

A large black arrow points from the line `<!-- begin server element-->` in the `cockpit.component.ts` file to the corresponding line in the `app.component.html` file.

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 ▶ serverelement.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 }}</strong>
9         <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
10      </p>
11    </div>
12  </div>
13 <!-- end server element-->
```

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 }}</strong>
8        <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
9      </p>
10     </div>
11   </div>
12   <!-- end server element-->
13
```

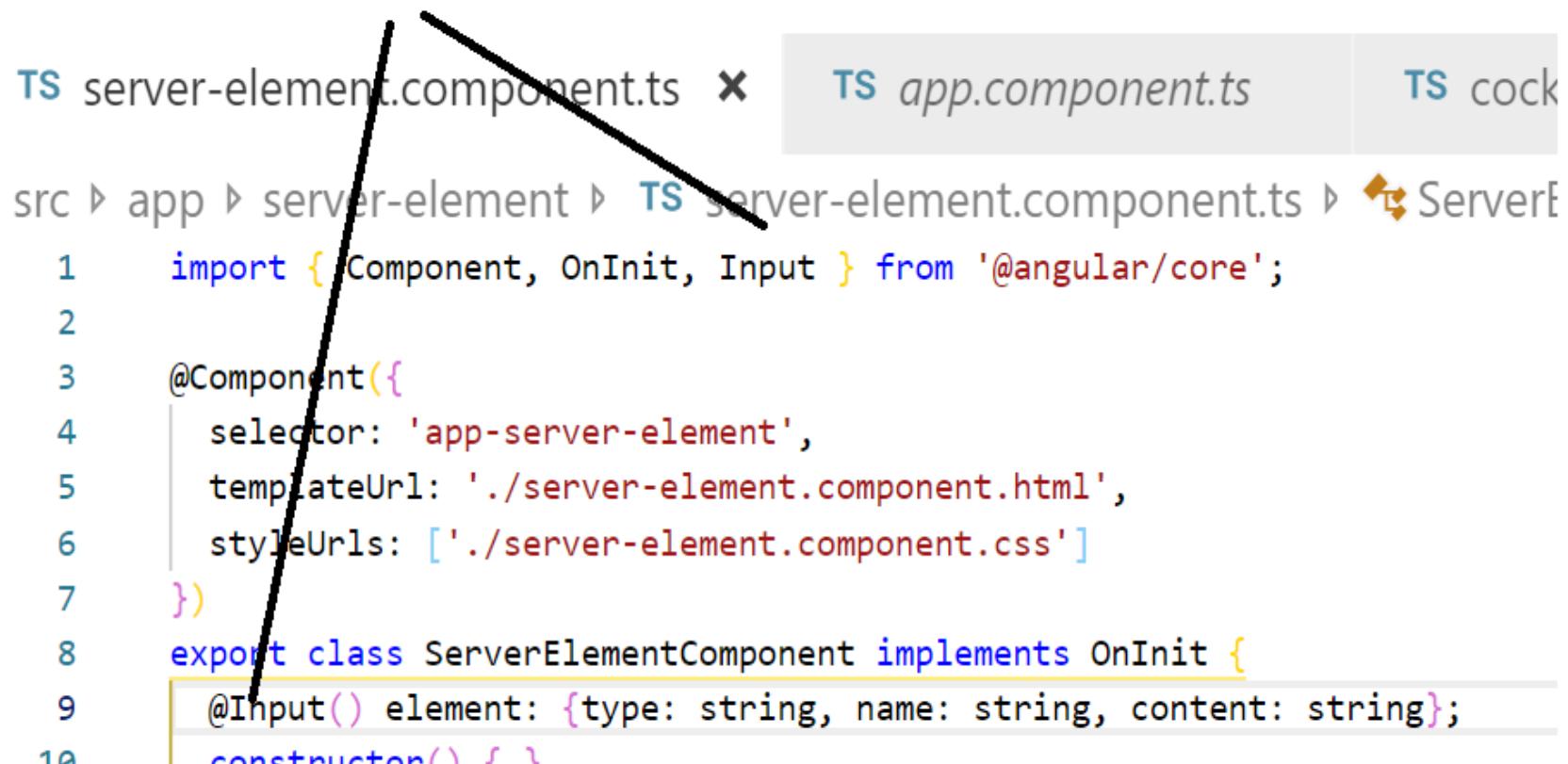
example4-component-and-databinding-Version3

- 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"



The screenshot shows a code editor with three tabs at the top: "TS server-element.component.ts" (crossed out), "TS app.component.ts" (selected), and "TS cock". Below the tabs, the file structure is shown: "src" > "app" > "server-element" > "TS server-element.component.ts". A red line with an arrow points from the "server-element.component.ts" tab towards the code below. The code itself is as follows:

```
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() element: {type: string, name: string, content: string};
10  constructor() { }
```

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

```
src ▶ app ▶ cockpit ▶ TS cockpit.component.ts ▶ CockpitComponent
```

```
16
17     /* Begin cockpit */
18
19     onAddServer() {
20         /*
21             this.serverElements.push({
22                 type: 'server',
23                 name: this.newServerName,
24                 content: this.newServerContent
25             });
26         */
27     }
28
29
30     onAddBlueprint() {
31         /*
32             this.serverElements.push({
33                 type: 'blueprint',
34                 name: this.newServerName,
35                 content: this.newServerContent
36             });
37         */
38     }
39
40     /* End cockpit */
```

For now, Comment out the following code

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

component.ts TS cockpit.component.ts TS app.component.ts <> app.component.html X

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

```
1  <div class="container">
2    <!-- Begin cockpit-->
3    <app-cockpit></app-cockpit>    Add these two new lines
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  
```

example4-component-and-databinding-Version3

Add new Servers or blueprints!

Server Name

Server Content

Add Server

Add Server Blueprint

Testserver

Just a test!

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

The screenshot shows a code editor with two tabs: 'server-element.component.ts' and 'app.component.html'. The 'app.component.html' tab is active, displaying the following code:

```
<div class="container">
  <!-- Begin cockpit-->
  <app-cockpit></app-cockpit>
  <!-- End cockpit-->
  <hr>
  <div class="row">
    <div class="col-xs-12">
      <!-- begin server element-->
      <app-server-element *ngFor="let serverElement of serverElements"
        [srvElement]="serverElement"></app-server-element>
      <!-- end server element-->
    </div>
  </div>
```

A tooltip is displayed over the line '[srvElement]="' with the text: 'Change from element to 'srvElement' for the new alias.' A large black 'X' is drawn over the entire code block.

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)

app.component.html TS cockpit.component.ts TS app.component.ts X

src > app > TS app.component.ts > AppComponent > onServerAdded

```
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  onServerAdded(serverData: {serverName: string, serverContent: string}) {
12
13    this.serverElements.push({
14      type: 'server',
15      name: serverData.serverName,
16      content: serverData.serverContent
17    });
18
19  }
20
21
22  onBlueprintAdded(blueprintData: {serverName: string, serverContent: string}) {
23    this.serverElements.push({
24      type: 'blueprint',
25      name: blueprintData.serverName,
26      content: blueprintData.serverContent
27    });
28 }
```

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 X

TS app.component.ts

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

```
1  <div class="container">                                add this line
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 }
```

Make sure these 2 variables exist

Add these import
Add these two lines
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)

Add new Servers or blueprints!

Server Name

Server Content

Type the following
and click "Add
Server".

Add Server **Add Server Blueprint**

Testserver

Just a test!

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

Add new Servers or blueprints!

Server Name

Type the following
and click "Add
Server".

Server Content

Add Server Add Server Blueprint

Testserver

Just a test!

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

Add new Servers or blueprints!

Server Name

A new Server

Server Content

Some testcontent

Add Server

Add Server Blueprint

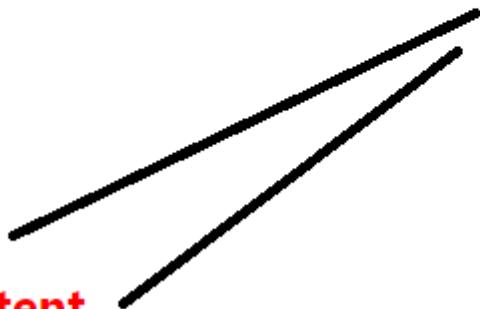
Testserver

Just a test!

A new Server

Some testcontent

new two lines
added



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

Add new Servers or blueprints!

Server Name

A new Server

Server Content

Some testcontent

Add Server

Add Server Blueprint

Testserver

Just a test!

A new Server

Some testcontent

A new Server

Some testcontent

After clicking "Add
Server Blueprint"

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

TS cockpit.component.ts X

src > app > cockpit > TS 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',      Add this alias "bpCreated" in cockpit.component.ts
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 = '';
17 newServerContent = '';
```



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

In app.component.html modify to "bpCreated".

package-lock.json

{ } tslint.json

TS cockpit.component.ts

↔ app.component.html

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

```
1   <div class="container">
2     <!-- Begin cockpit-->
3     <app-cockpit (serverCreated)="onServerAdded($event)"
4       (bpCreated)="onBlueprintAdded($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>
17
```



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

- Run command:ng serve -o

```
C:\workspace-UT-Angular7-Sept-2019\Session5\example4-component-and-databinding-Version6\cmp-databinding-start>ng serve -o
```

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

Add new Servers or blueprints!

Server Name
test server name

Server Content
test server content

Add Server Add Server Blueprint

Testserver
Just a test!
est server name
test server content

type above and click "Add Server".

alias "bpCreated"

Line 17, Column 25 (source mapped from main.js)

```
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}>()
9   @Output('bpCreated') blueprintCreated = new EventEmitter<{serverName: 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 }
```

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

The screenshot shows a browser developer tools interface with the 'Sources' tab selected. The left sidebar displays a file tree under the 'app' directory, including 'cockpit', 'server-element', 'environments', and other files like 'app.component.css' and 'main.ts'. The right panel shows the content of 'app.component.html'. A red arrow points from the text 'In "app.component.html" changed to "bpCreated"' to the word 'bpCreated' in the code.

Add new Servers or blueprints!

Server Name
test server name

Server Content
test server content

Add Server Add Server Blueprint

Testserver
Just a test!

test server name

Console Elements Sources Network Performance Memory Application Security

Page Filesystem >:

(index) app.component.html X cockpit.component.ts >

Pretty-print this minified file? more never show x

```
ded($event)\n  (bpCreated)="onBlueprintAdded($event)\n  >
```

In "app.component.html" changed to "bpCreated"

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.