Codergirl – Frontend

Unit 2 - Class 3

December 7, 2020

Agenda

- Components and Databinding Deep dive
- Studio – Assignment 4
- Holiday break is 12/28 to 12/30, with the last class being 12/23 and the first of new year being on 1/4/21.

Splitting the app into Components

npm i

ng serve

ng g c cockpit

ng g c server-element

Cockpit component

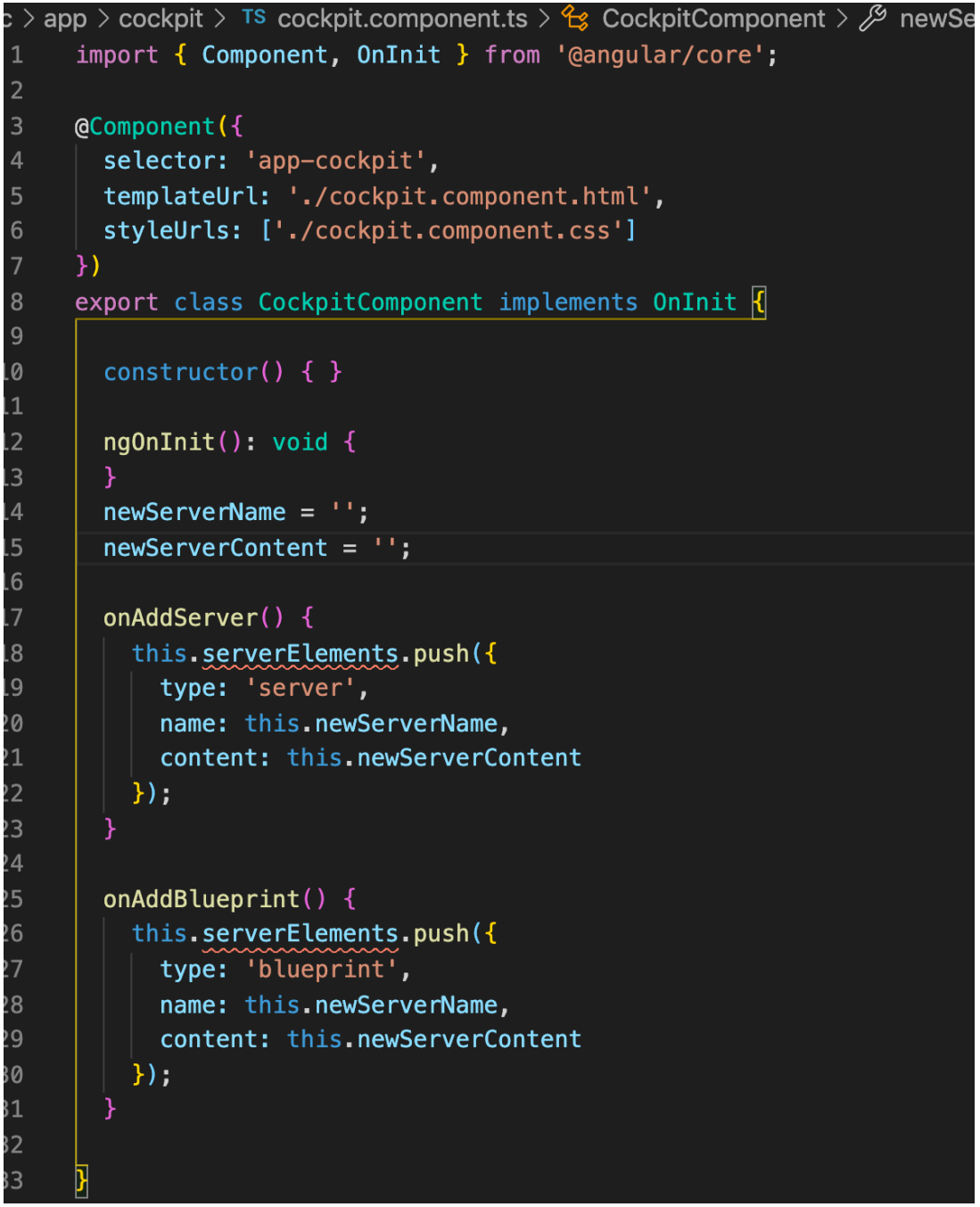
Copy the first row into cockpit.component.html

```
<div class="row">
    <div class="col-xs-12">
      <p>Add new Servers or blueprints!</p>
      <label>Server Name</label>
      <input type="text" class="form-control" [(ngModel)]="newServerName">
      <label>Server Content</label>
      <input type="text" class="form-control" [(ngModel)]="newServerContent">
      <br>
      <button
        class="btn btn-primary"
        (click)="onAddServer()">Add Server</button>
      <button
        class="btn btn-primary"
        (click)="onAddBlueprint()">Add Server Blueprint</button>
    </div>
  </div>
```

related methods in ts
as well as

newServerName = '';

newServerContent = '';



Server-element Component

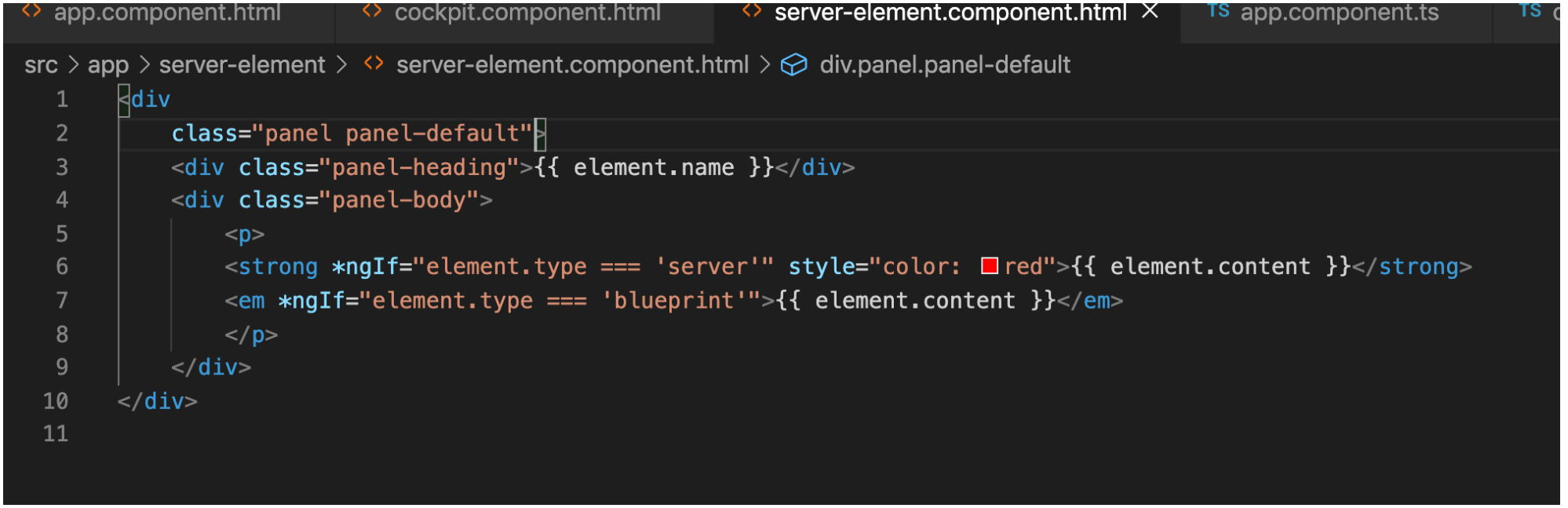
Move the row into server-element.html

app > server-element > server-element.component.html > ...

```
<div
    class="panel panel-default"
    *ngFor="let element of serverElements">
    <div class="panel-heading">{{ element.name }}</div>
    <div class="panel-body">
        <p>
        <strong *ngIf="element.type === 'server'" style="color: red">{{ element.content }}</strong>
        <em *ngIf="element.type === 'blueprint'">{{ element.content }}</em>
        </p>
    </div>
</div>
```

Server-element Component

Since server-element is a single server we remove the ngFor



app-component.html

Add app-cockpit and app-server-element and ngFor for serverElements

```
1   <div class="container">
2     <app-cockpit></app-cockpit>
3     <hr>
4     <div class="row">
5       <div class="col-xs-12">
6         <app-server-element *ngFor="let serverElement of serverElements"></app-server-element>
7       </div>
8     </div>
9   </div>
10
```

App is broke!


TERMINAL
PROBLEMS 2
OUTPUT
DEBUG CONSOLE
1: node
Date: 2020-12-05T01:58:31.560Z - Hash: a24b7dcbabdb5a5df3cb
6 unchanged chunks
Time: 56ms
: Compiled successfully.
ERROR in src/app/cockpit/cockpit.component.ts:18:10 - error TS2339: Property 'serverElements' does not exist on type 'CockpitComponent'.
18 this.serverElements.push({
src/app/cockpit/cockpit.component.ts:26:10 - error TS2339: Property 'serverElements' does not exist on type 'CockpitComponent'.
26 this.serverElements.push({

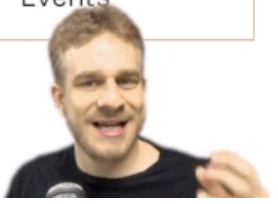
Need to pass data between Components

Property and Event binding to pass data into Elements

They can also be used on directives and Components.

Need to pass data between Components

HTML Elements	Directives	Components
Native Properties & Events	Custom Properties & Events	Custom Properties & Events



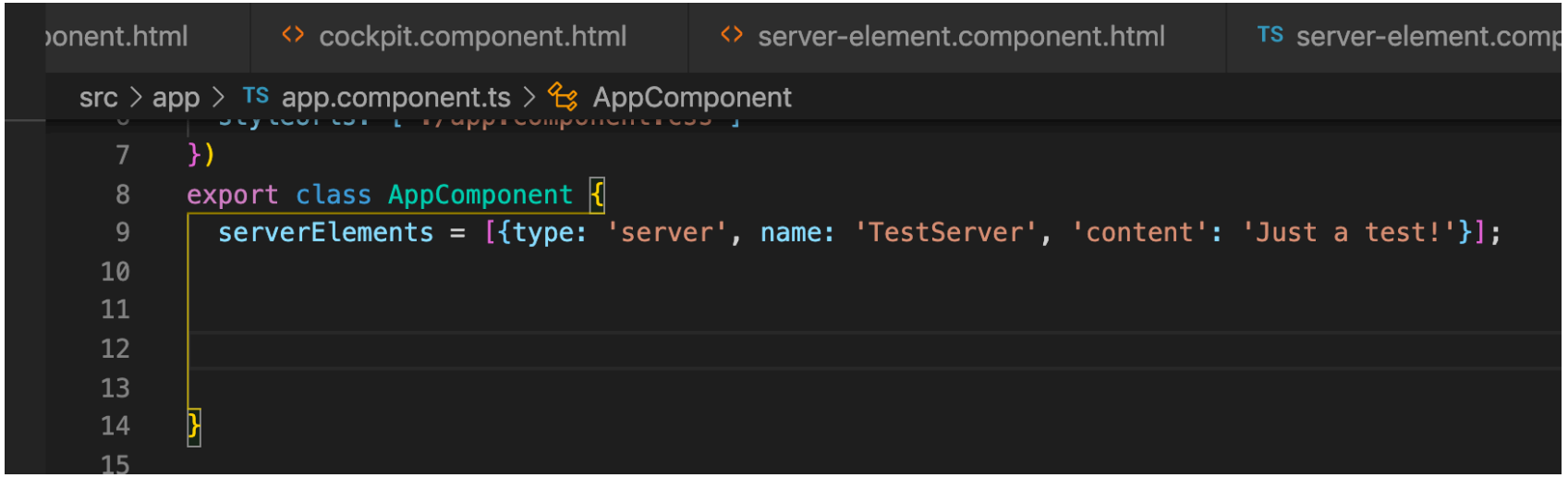
Comment out code that isn’t compiling

```
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    }
34
```

server-element.html has element

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-server-element',
  templateUrl: './server-element.component.html',
  styleUrls: ['./server-element.component.css']
})
export class ServerElementComponent implements OnInit {

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

  ngOnInit(): void {
  }

}
```

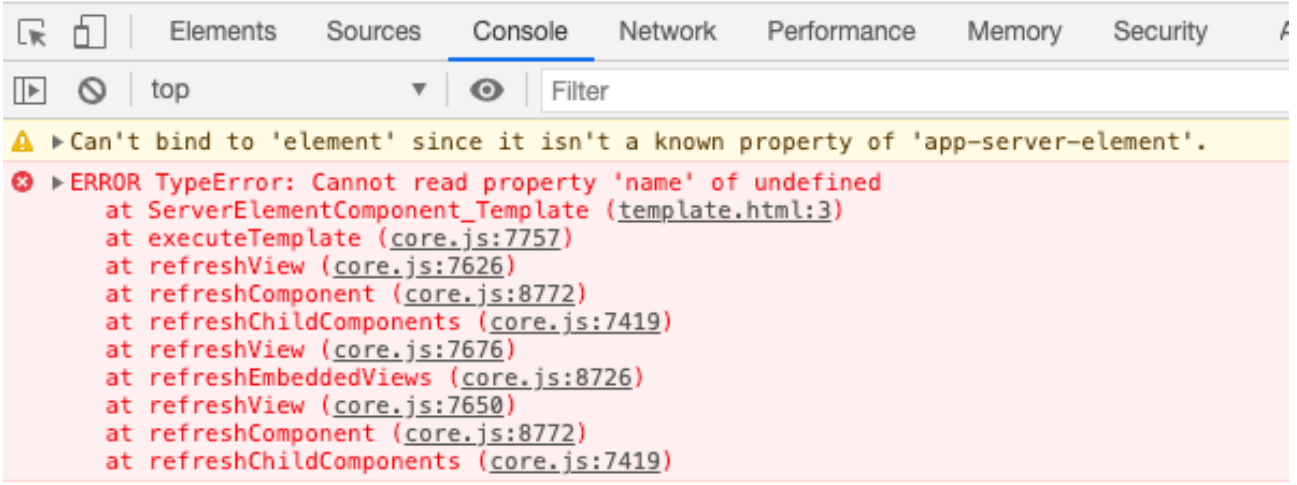
app-component.ts



Properties are not exposed outside of the component

app-component.ts

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



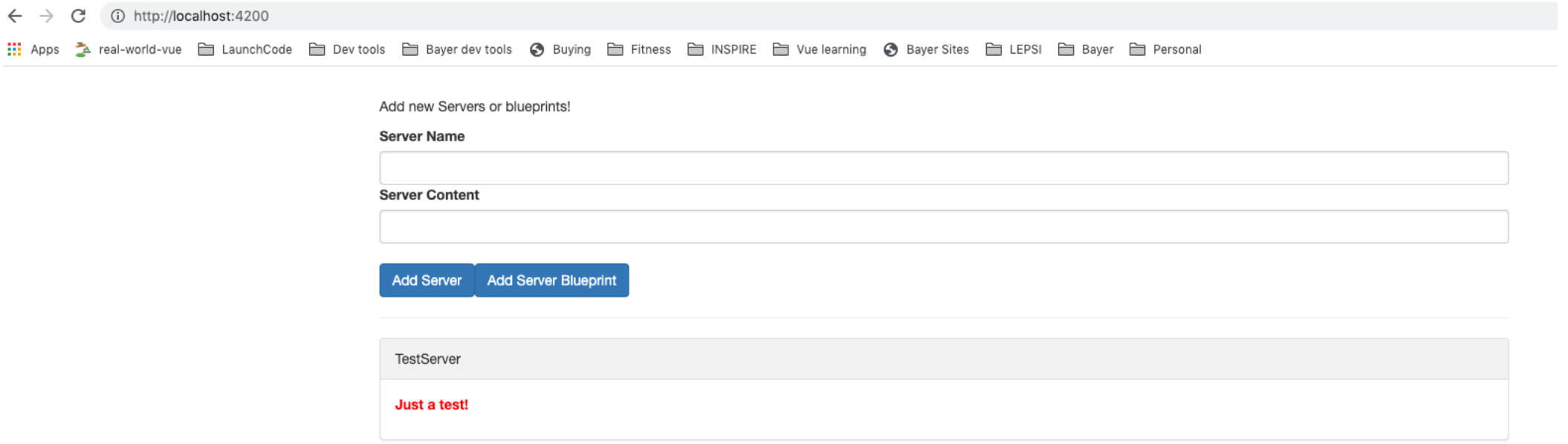
app-component.ts

Need to add a decorator for the property to be accessible outside

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
10    @Input() element: {type: string, name: string, content: string}
11    constructor() { }
12
13    ngOnInit(): void {
14    }
15
16  }
```

Now app loads!



…also supports alias

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

```typescript
import { Component, OnInit, Input } from '@angular/core';

@Component({
  selector: 'app-server-element',
  templateUrl: './server-element.component.html',
  styleUrls: ['./server-element.component.css']
})
export class ServerElementComponent implements OnInit {

  @Input('srvElement') element: {type: string, name: string, content: string}
  constructor() { }

  ngOnInit(): void {
  }

}
```

Informing parent component

app.component.ts

src > app > app.component.ts > AppComponent

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
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
    });
  }
}

```

Informing parent component

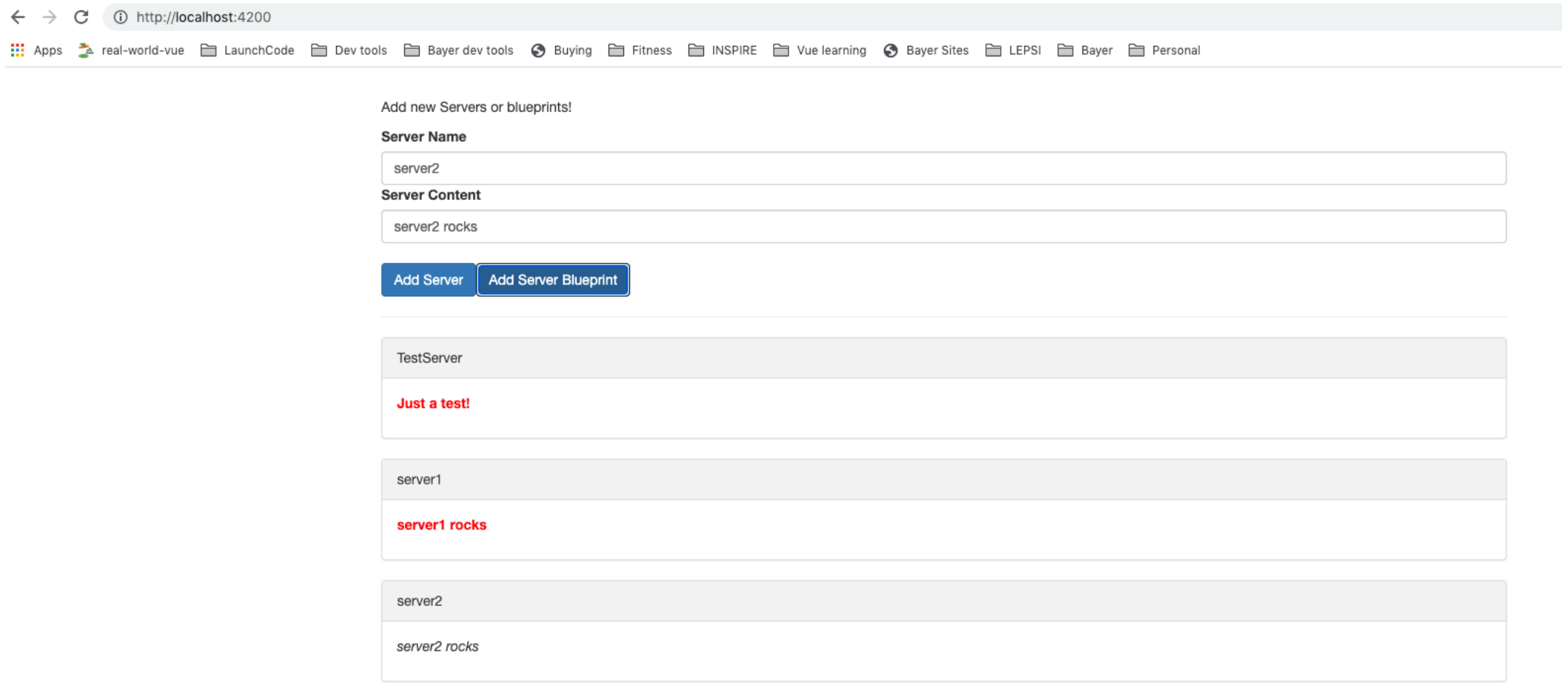
TS cockpit.component.ts ×

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

```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 {
  serverCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  @Output() blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  @Output() newServerName = '';
  newServerContent = '';

  constructor() { }

  ngOnInit(): void {
  }

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

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

}
```

(property) serverName: string

App works!


http://localhost:4200
Apps
real-world-vue
LaunchCode
Dev tools
Bayer dev tools
Buying
Fitness
INSPIRE
Vue learning
Bayer Sites
LEPSI
Bayer
Personal
Add new Servers or blueprints!
Server Name
server2
Server Content
server2 rocks
Add Server
Add Server Blueprint
TestServer
Just a test!
server1
server1 rocks
server2
server2 rocks

Alias works for output too.

cockpit.component.ts

src > app > cockpit > cockpit.component.ts > 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('bpCreated') blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  newServerName = '';
  newServerContent = '';

  constructor() { }

  ngOnInit(): void {
  }

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

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

}
```

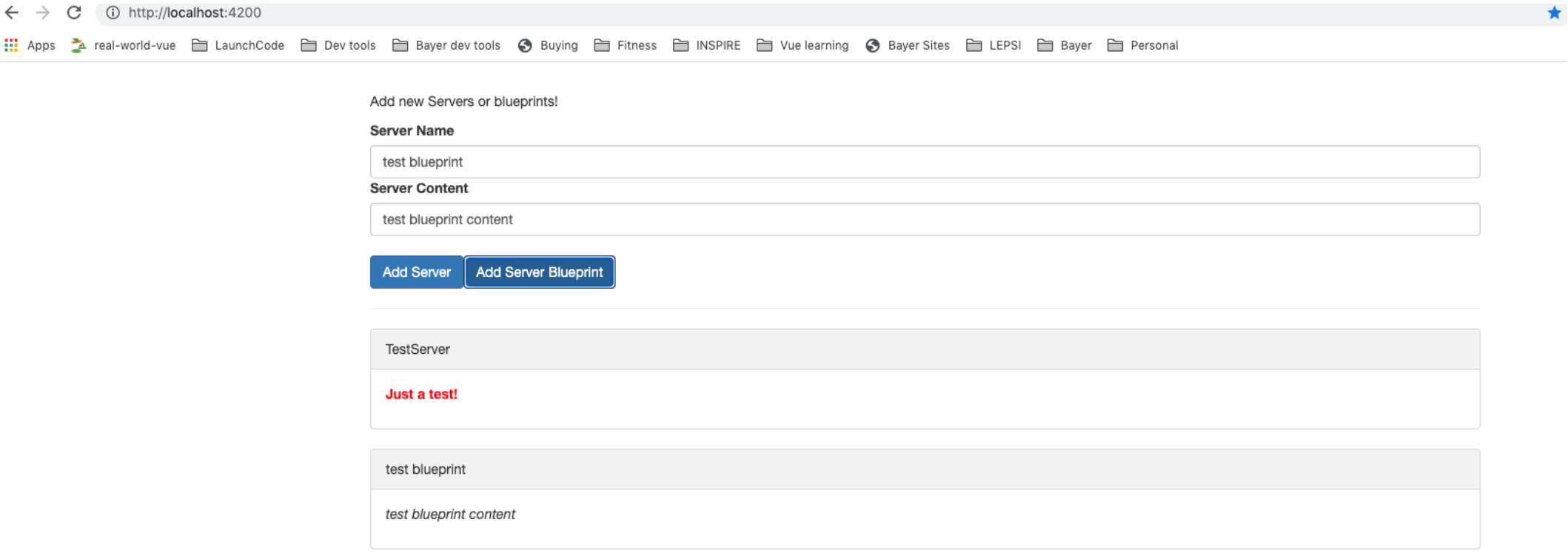
Alias works for output too.

TS cockpit.component.ts | TS app.component.ts

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

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

App works


http://localhost:4200
Apps
real-world-vue
LaunchCode
Dev tools
Bayer dev tools
Buying
Fitness
INSPIRE
Vue learning
Bayer Sites
LEPSI
Bayer
Personal
Add new Servers or blueprints!
Server Name
test blueprint
Server Content
test blueprint content
Add Server
Add Server Blueprint
TestServer
Just a test!
test blueprint
test blueprint content

View Encapsulation

Styles in CSS only get applied to Components they belong to.

Angular enforces style encapsulation.

All elements have a unique attribute names.

Similar to Shadow DOM – each element has it's own. Not supported by all browsers. Hence Angular implements it.

Encapsulation can be overridden. None is used to have global CSS. Default is Emulated and Native isn't supported by all browsersss.

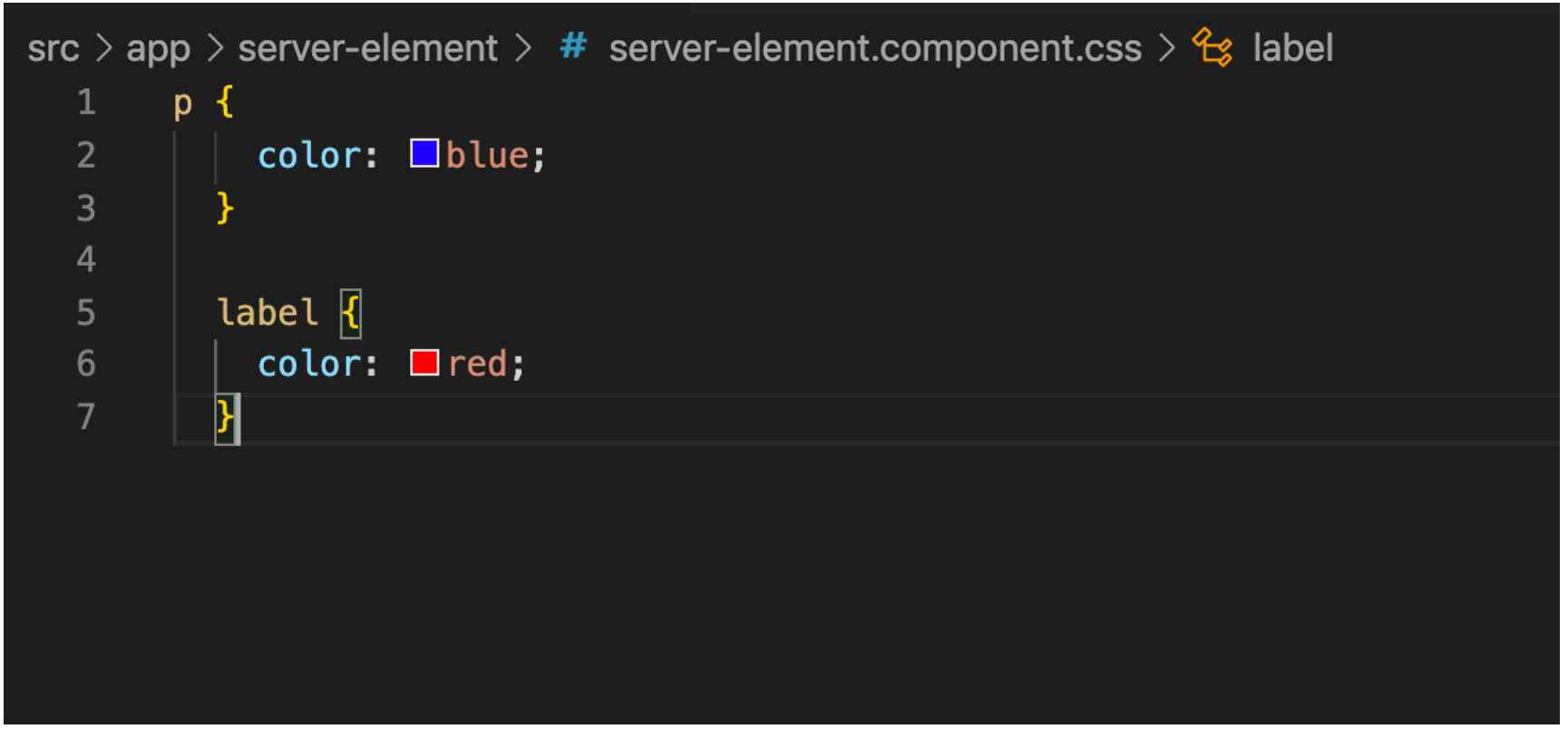
View Encapsulation

TS server-element.component.ts ×

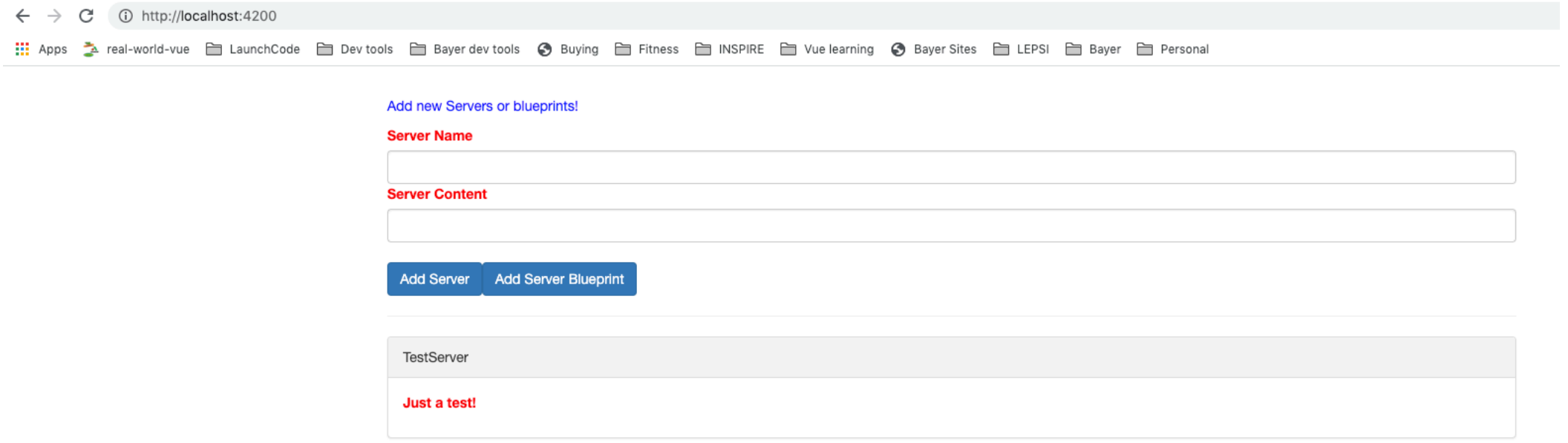
src > app > server-element > TS server-element.component.ts > ...

```
1   import { Component, OnInit, Input, ViewEncapsulation } from '@angular/core';
2
3   @Component({
4     selector: 'app-server-element',
5     templateUrl: './server-element.component.html',
6     styleUrls: ['./server-element.component.css'],
7     encapsulation: ViewEncapsulation.None
8   })
9   export class ServerElementComponent implements OnInit {
10
11    @Input('srvElement') element: {type: string, name: string, content: string}
12    constructor() { }
13
14    ngOnInit(): void {
15    }
16
17  }
18
```

View Encapsulation



View Encapsulation



Local Reference

Instead of two-way data binding, we can place a local reference on an element.

It can be applied to any element.

cockpit.component.html | cockpit.component.ts

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

```
<div class="row">
  <div class="col-xs-12">
    <p>Add new Servers or blueprints!</p>
    <label>Server Name</label>
    <!-- <input type="text" class="form-control" [(ngModel)]="newServerName"> -->
    <input
    type="text"
    class="form-control"
    #serverNameInput
    >
    <label>Server Content</label>
    <input type="text" class="form-control" [(ngModel)]="newServerContent">
    <br>
    <button
      class="btn btn-primary"
      (click)="onAddServer(serverNameInput)">Add Server</button>
    <button
      class="btn btn-primary"
      (click)="onAddBlueprint()">Add Server Blueprint</button>
  </div>
</div>

```

Local Reference

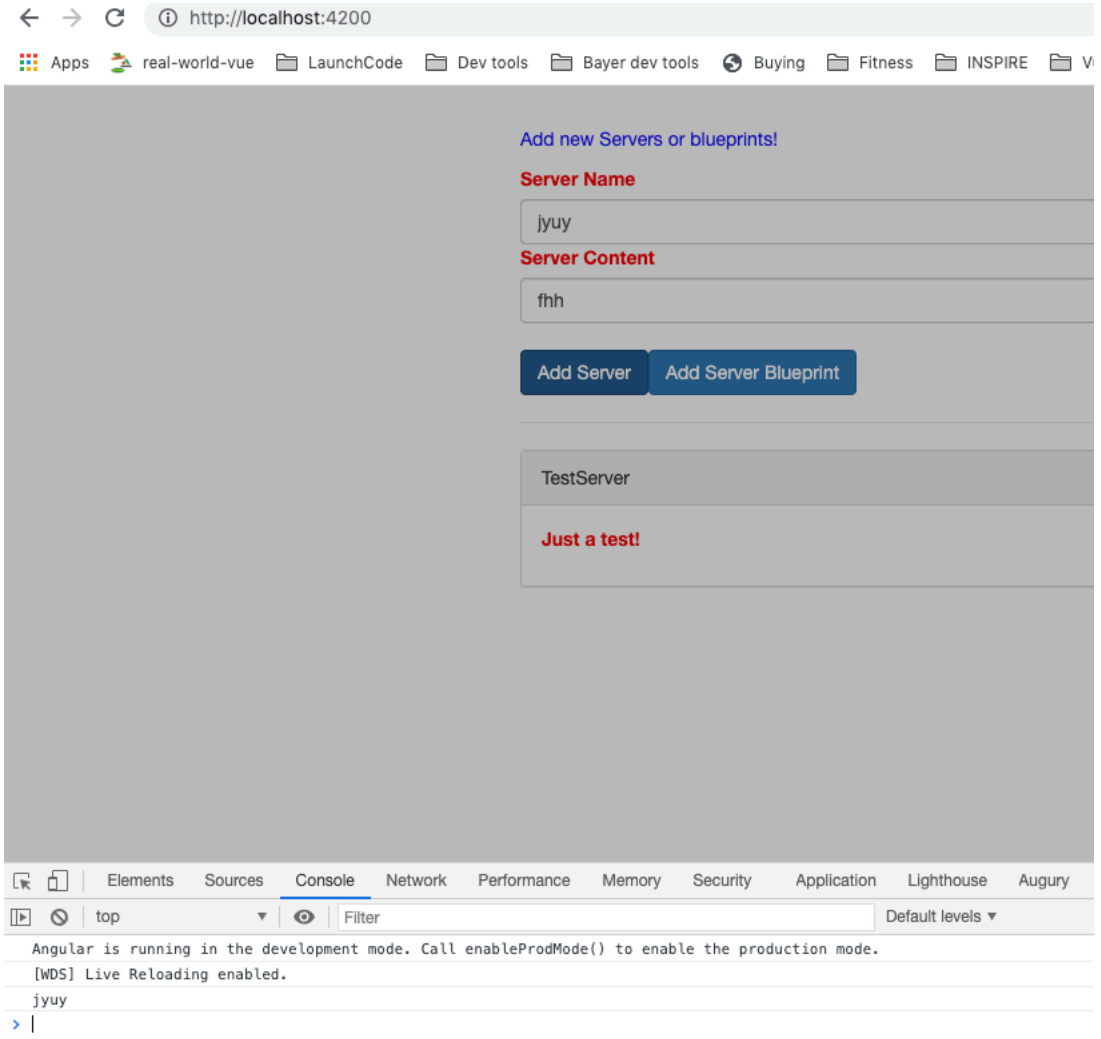
Instead of two-way data binding, we can place a local reference on an element.

It can be applied to any element.

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

```typescript
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('bpCreated') blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  newServerName = '';
  newServerContent = '';

  constructor() { }

  ngOnInit(): void {
  }

  onAddServer(nameInput) {
    console.log(nameInput.value);
    this.serverCreated.emit({serverName: this.newServerName, serverContent: this.newServerContent});
  }

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

}

```

Local Reference



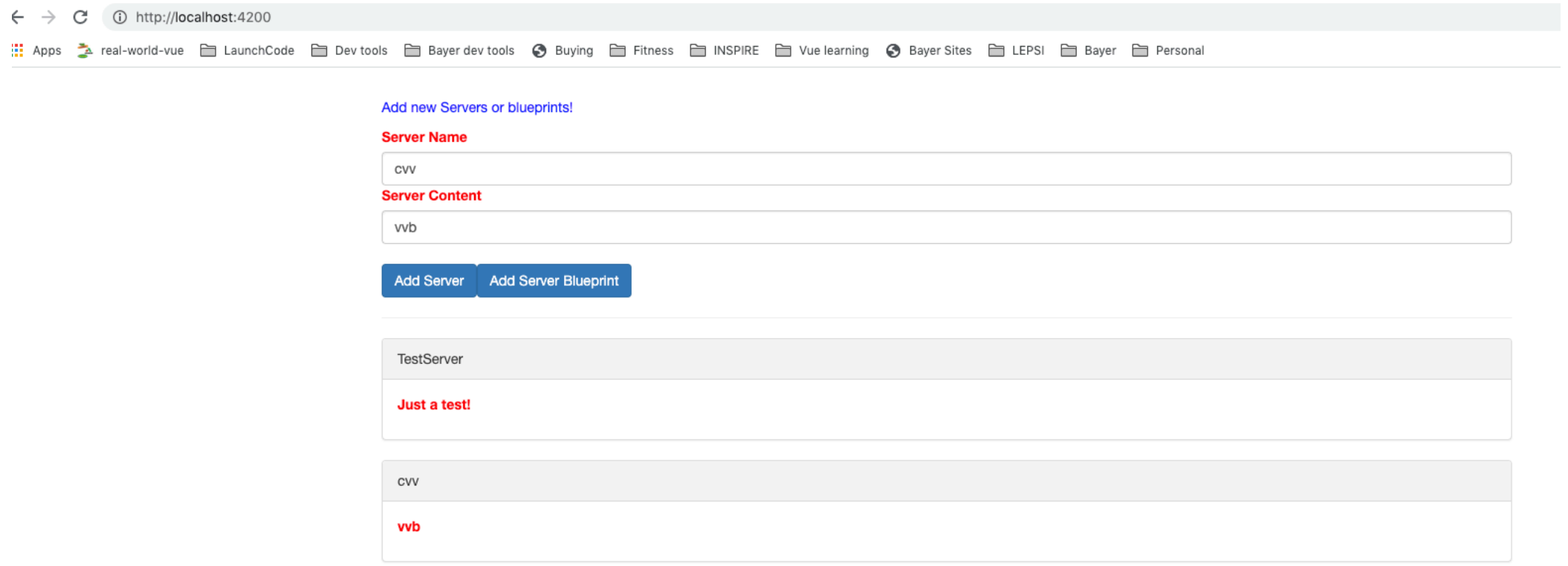
Local Reference

cockpit.component.html | cockpit.component.ts

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

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

Local Reference



ViewChild

cockpit.component.html | cockpit.component.ts

src > app > cockpit > cockpit.component.html > div.row > div.col-xs-12 > button.btn.btn-primary

```
<div class="row">
    <div class="col-xs-12">
      <p>Add new Servers or blueprints!</p>
      <label>Server Name</label>
      <!-- <input type="text" class="form-control" [(ngModel)]="newServerName"> -->
      <input
      type="text"
      class="form-control"
      #serverNameInput
      >
      <label>Server Content</label>
      <!-- <input type="text" class="form-control" [(ngModel)]="newServerContent"> -->
      <input
      type="text"
      class="form-control"
      #serverContentInput
      >
      <br>
      <button
        class="btn btn-primary"
        (click)="onAddServer(serverNameInput)">Add Server</button>
      <button
        class="btn btn-primary"
        (click)="onAddBlueprint(serverNameInput)">Add Server Blueprint</button>
    </div>
  </div>
```

ViewChild

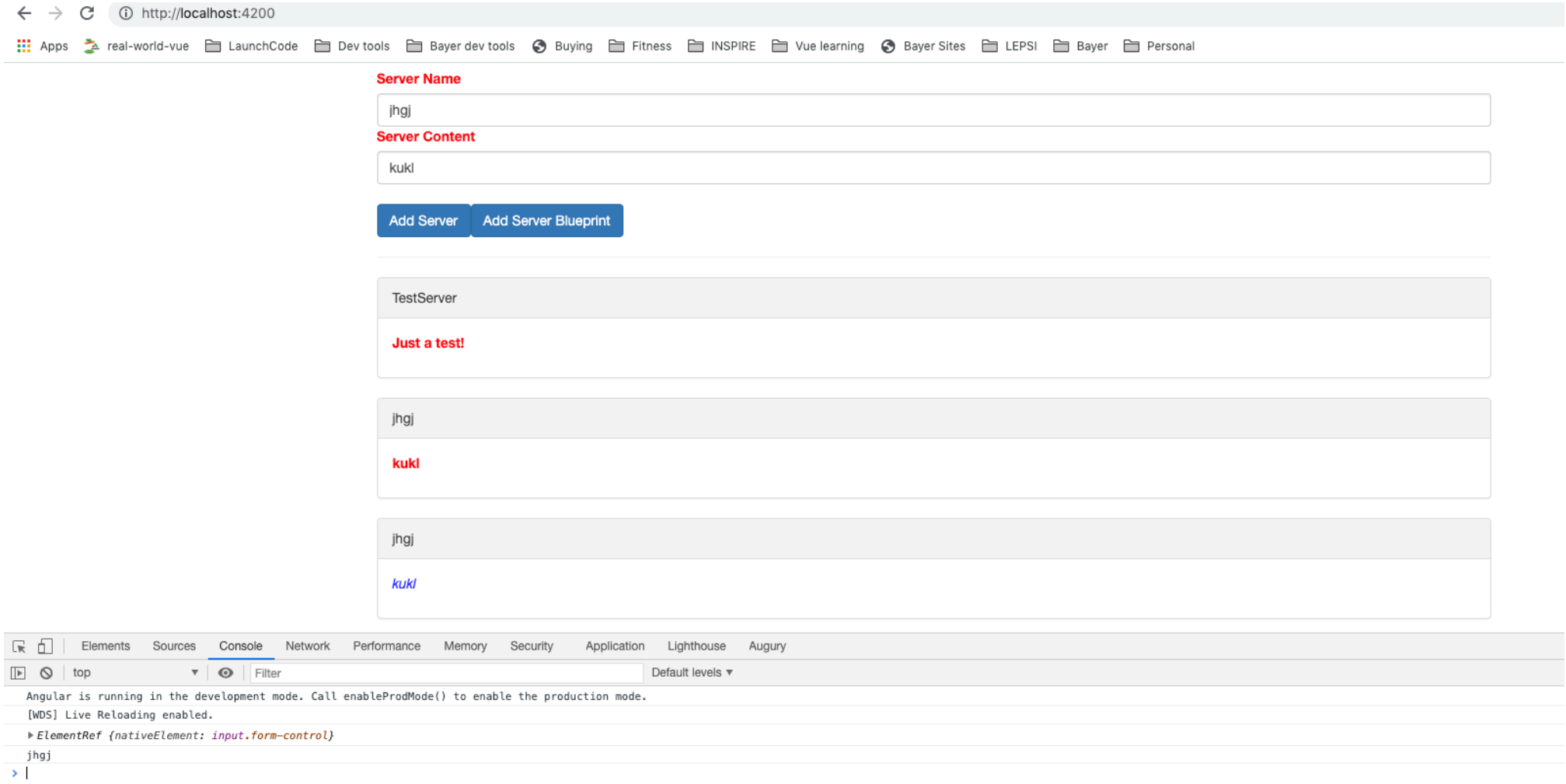
cockpit.component.html | cockpit.component.ts

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

```ts
import { Component, OnInit, EventEmitter, Output, ViewChild, ElementR

@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('bpCreated') blueprintCreated = new EventEmitter<{serverName: string, serverContent: string}>();
  // newServerName = '';
  // newServerContent = '';
  @ViewChild('serverContentInput') serverContentInput: ElementRef;
  constructor() { }

  ngOnInit(): void {
  }

  onAddServer(nameInput: HTMLInputElement) {
    console.log(this.serverContentInput);
    console.log(nameInput.value);
    this.serverCreated.emit({serverName: nameInput.value, serverContent: this.serverContentInput.nativeElement.value});
  }

  onAddBlueprint(nameInput: HTMLInputElement) {
    this.blueprintCreated.emit({serverName: nameInput.value, serverContent: this.serverContentInput.nativeElement.value});
  }

}

```

Find Aa Abl .* ? of 5

ViewChild



Don’t access the DOM and modify using ViewChild!

ng-content

Add the content into a Component using ng-content, that it finds between opening and closing tag.

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

```
1   <div class="container">
2     <app-cockpit
3     (serverCreated)="onServerAdded($event)"
4     (bpCreated)="onBlueprintAdded($event)"
5     ></app-cockpit>
6     <hr>
7     <div class="row">
8       <div class="col-xs-12">
9         <app-server-element
10        *ngFor="let serverElement of serverElements"
11        [srvElement]="serverElement">
12        <p>
13          <strong *ngIf="serverElement.type === 'server'" style="color: red">{{ serverElement.content }}</strong>
14          <em *ngIf="serverElement.type === 'blueprint'">{{ serverElement.content }}</em>
15        </p>
16      </app-server-element>
17      </div>
18    </div>
19  </div>
20
```

Find Aa Abl .* No results

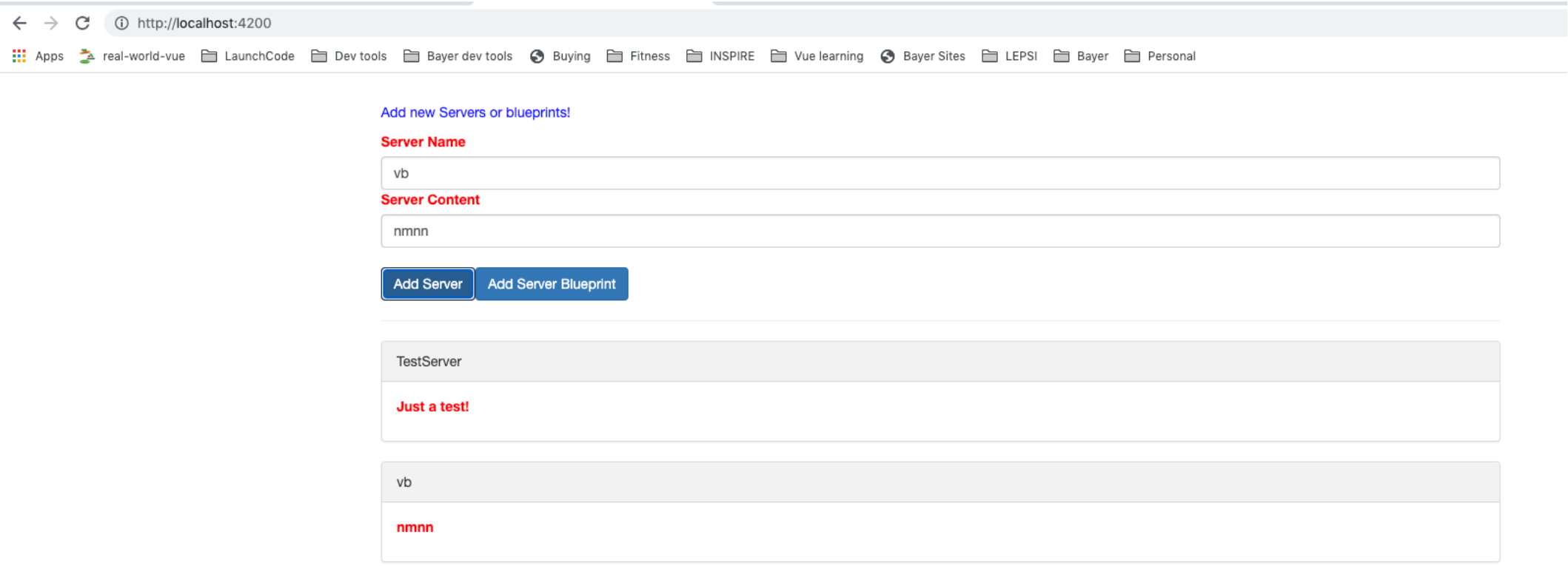
ng-content

server-element.component.html × app.component.html

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

```
1   <div
2       class="panel panel-default">
3       <div class="panel-heading">{{ element.name }}</div>
4       <div class="panel-body">
5           <ng-content></ng-content>
6       </div>
7   </div>
8
```

ng-content


http://localhost:4200
Apps
real-world-vue
LaunchCode
Dev tools
Bayer dev tools
Buying
Fitness
INSPIRE
Vue learning
Bayer Sites
LEPSI
Bayer
Personal
Add new Servers or blueprints!
Server Name
vb
Server Content
nmnn
Add Server
Add Server Blueprint
TestServer
Just a test!
vb
nmnn

Component Lifecycle

Lifecycle

ngOnChanges	Called after a bound input property changes
ngOnInit	Called once the component is initialized
ngDoCheck	Called during every change detection run
ngAfterContentInit	Called after content (ng-content) has been projected into view
ngAfterContentChecked	Called every time the projected content has been checked
ngAfterViewInit	Called after the component's view (and child views) has been initialized
ngAfterViewChecked	Called every time the view (and child views) have bee cked
ngOnDestroy	Called once the component is about to be destroyed

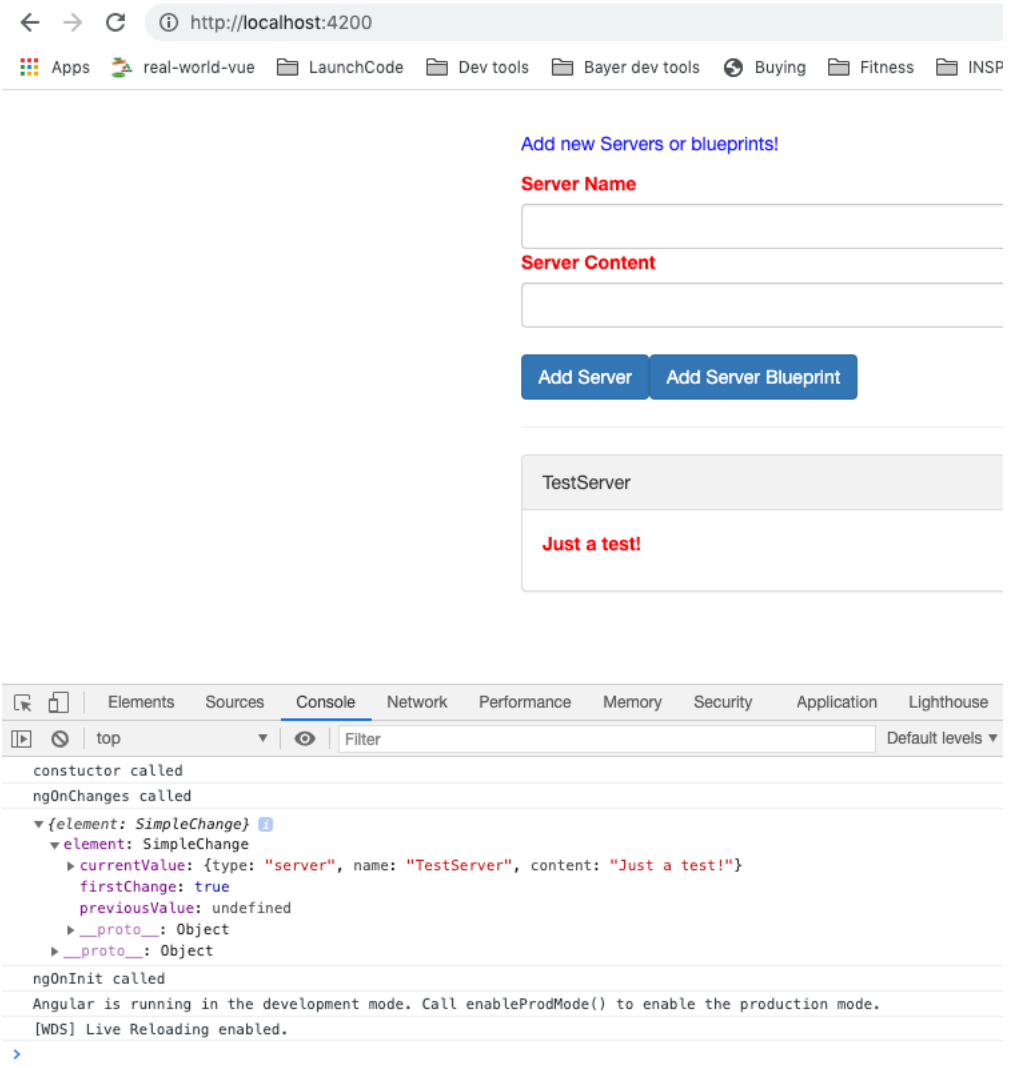
Component Lifecycle

server-element.component.ts

src > app > server-element > server-element.component.ts > ServerElementComponent > con

```
  Component,
  OnInit,
  Input,
  ViewEncapsulation,
  OnChanges,
  SimpleChanges
} from '@angular/core';

@Component({
  selector: 'app-server-element',
  templateUrl: './server-element.component.html',
  styleUrls: ['./server-element.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class ServerElementComponent implements OnInit, OnChanges {

  @Input('srvElement') element: {type: string, name: string, content: string}
  constructor() {
    console.log('constuctor called');
  }

  ngOnChanges(changes: SimpleChanges) {
    console.log('ngOnChanges called');
    console.log(changes);
  }

  ngOnInit(): void {
    console.log('ngOnInit called');
  }

}
```

Component Lifecycle



Component Lifecycle

Review code for server-component.ts

Assignment

npm i

ng serve

ng g c gameControl

ng g c odd

ng g c even

Questions?

Studio!