Working with Data Continued

App.component.css

```
.container {
  margin-top: 30px;
}

p {
  color: blue;
}
```

App.component.html

```
<div class="container">
  <app-cockpit
    (serverCreated)="onServerAdded($event)"
    (bpCreated)="onBlueprintAdded($event)"
  ></app-cockpit>
  <hr>
  <div class="row">
    <div class="col-xs-12">
      <button class="btn btn-primary" (click)="onChangeFirst()">Change first Element</button>
      <button class="btn btn-danger" (click)="onDestroyFirst()">Destroy first Component</
button>
      <app-server-element
        *ngFor="let serverElement of serverElements"
        [srvElement]="serverElement"
        [name]="serverElement.name">
        <p #contentParagraph>
          <strong *ngIf="serverElement.type === 'server'" style="color:
red">{{ serverElement.content }}</strong>
          <em *ngIf="serverElement.type === 'blueprint'">{{ serverElement.content }}</em>
        </p>
      </app-server-element>
    </div>
```

```
  </div>
</div>
```

App.component.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
    });
  }

  onChangeFirst() {
    this.serverElements[0].name = 'Changed!';
  }
```

```
  onDestroyFirst() {
    this.serverElements.splice(0, 1);
  }
}
```

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';



import { AppComponent } from './app.component';
import { CockpitComponent } from './cockpit/cockpit.component';
import { ServerElementComponent } from './server-element/server-element.component';

@NgModule({
  declarations: [
    AppComponent,
    CockpitComponent,
    ServerElementComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

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">-->
    <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>
```

cockpit.component.ts

```
import { Component, OnInit, EventEmitter, Output, ViewChild, ElementRef } 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 = '';
  @ViewChild('serverContentInput', { static: false }) serverContentInput: ElementRef;

  constructor() { }

  ngOnInit() {
  }

  onAddServer(nameInput: HTMLInputElement) {
    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
    });
  }

}
```

sever-element.component.css

```
p {
  color: blue;
}

label {
  color: red;
}
```

sever-element.component.HTML

```
div
  class="panel panel-default">
  <!--<div class="panel-heading">{{ element.name }}</div>-->
  <div class="panel-heading" #heading>{{ name }}</div>
  <div class="panel-body">
    <ng-content></ng-content>
  </div>
</div>
```

sever-element.component.ts

```
mport {
  Component,
  OnInit,
  Input,
  ViewEncapsulation,
```

```
  OnChanges,
  SimpleChanges,
  DoCheck,
  AfterContentInit,
  AfterContentChecked,
  AfterViewInit,
  AfterViewChecked,
  OnDestroy,
  ViewChild,
  ElementRef,
  ContentChild
} from '@angular/core';

@Component({
  selector: 'app-server-element',
  templateUrl: './server-element.component.html',
  styleUrls: ['./server-element.component.css'],
  encapsulation: ViewEncapsulation.Emulated // None, Native
})
export class ServerElementComponent implements
  OnInit,
  OnChanges,
  DoCheck,
  AfterContentInit,
  AfterContentChecked,
  AfterViewInit,
  AfterViewChecked,
  OnDestroy {
  @Input('srvElement') element: {type: string, name: string, content: string};
  @Input() name: string;
  @ViewChild('heading', {static: true}) header: ElementRef;
  @ContentChild('contentParagraph', {static: true}) paragraph: ElementRef;

  constructor() {
    console.log('constructor called!');
  }
```

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

ngOnInit() {
  console.log('ngOnInit called!');
  console.log('Text Content: ' + this.header.nativeElement.textContent);
  console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);
}

ngDoCheck() {
  console.log('ngDoCheck called!');
}

ngAfterContentInit() {
  console.log('ngAfterContentInit called!');
  console.log('Text Content of paragraph: ' + this.paragraph.nativeElement.textContent);
}

ngAfterContentChecked() {
  console.log('ngAfterContentChecked called!');
}

ngAfterViewInit() {
  console.log('ngAfterViewInit called!');
  console.log('Text Content: ' + this.header.nativeElement.textContent);
}

ngAfterViewChecked() {
  console.log('ngAfterViewChecked called!');
}

ngOnDestroy() {
  console.log('ngOnDestroy called!');
```

```
  }

}
```