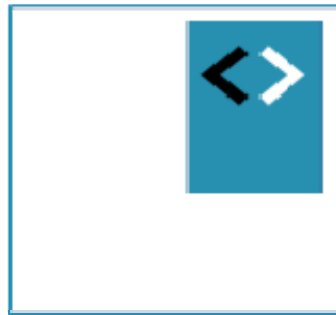




Smart components & View components



Peter Kassenaar –
info@kassenaar.com

“In Angular, all components are equal. There is no sense of different ‘types’ of components”



Smart components / View Components

- Design pattern
- Why ? Separation of concerns
 - **View component** is responsible for **presentation** (and *can* be used in a completely different environment with different component logic)
 - **Smart component** is responsible for **logic** → passing the [calculated] data to the view component.
- Smart components
 - AKA: *Container* components, *controller* components, *statefull* components
- View components
 - AKA: *pure* components, *dumb* components, *stateless*

Characteristics

- Smart components
 - Typically contain big(ger) chunks of logic
 - Typically have a small UI part (reference just the view component)
 - Pass the data to the view component
- View components
 - Typically contain no logic whatsoever
 - Get their data via `@Input()` decorators
 - Submit events via `@Output()` decorators
 - Have large chunks of UI

Example: ../300-smart-view-component

Home (default component)

Smart/View component

Default component- My favorite cities:


1 - Groningen	<input checked="" type="checkbox"/> Visited
2 - Hengelo	<input type="checkbox"/> Visited
3 - Den Haag	<input type="checkbox"/> Visited
4 - Enschede	<input checked="" type="checkbox"/> Visited
5 - Heerlen	<input type="checkbox"/> Visited

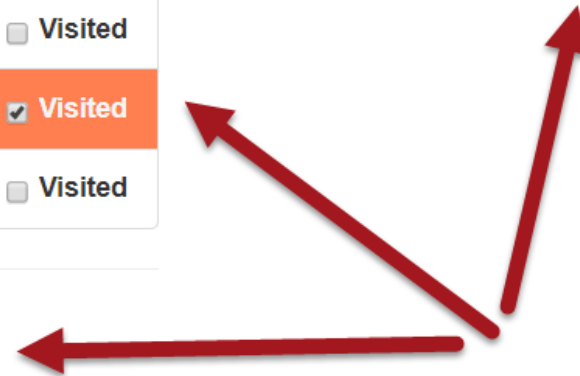
I Visited:

Groningen

Enschede

Current city: Groningen





Home – Default component

- Contains all logic and UI, combined in one component.
 - /home/home.component.html
 - /home/home.component.ts
- This typically works well for smaller applications

```
<div class="row">
  <div class="col-md-6">
    <h2>Default component- My f
    <ul class="list-group">
      <li *ngFor="let city of
        class="list-group-item"
        [class.visited]="city
        (click)="getCity(city
        {{ city.id}} - {{ cit
        <span class="pull-right
          <label>
        </ul>
    ..
  </div>
```

```
@Component({
  selector    : 'app-home',
  templateUrl: './home.component.html'
})
export class HomeComponent implements OnInit {
  ...
  ngOnInit() {
    this.cityService.getCities()
      .subscribe(cities => this.cities = cities);
  }

  getCity(city: City) {
    this.currentCity = city;
    this.cityPhoto    = `assets/img/${this.currentCity.name}.jpg`;
  }
  ...
}
```

Splitting it up in view components

- `../smart-view/smart.component.html | .ts.`
- Passing `[cities]` in
- Getting (events) out.
- `[cities]` can also be an Observable

```
<div class="row">
  <div class="col-md-6">
    <h2>Smart/view component- My favorite cities:</h2>
    <!-- The <city-list>-component is now
         the view component for list of cities -->
    <city-list [cities]="cities"
              (selectCity)="getCity($event)"
              (toggleVisited)="toggleCity($event)">
    </city-list>
    ...
  </div>
</div>
```

<city-list> View component

- Has no logic, just @Input()’s and @Output()’s

```
import {Component, EventEmitter, Input, Output} from '@angular/core';  
import {City} from '../shared/model/city.model';
```

```
@Component({  
  selector    : 'city-list',  
  templateUrl: './city-list.component.html'  
})
```

```
export class CityListComponent {
```

```
  @Input() cities: City[];
```

```
  @Output() selectCity: EventEmitter<City> = new EventEmitter<City>();
```

```
  @Output() toggleVisited: EventEmitter<City> = new EventEmitter<City>();
```

```
}
```


HTML of the view component

- .../smart-view/city-list/city-list-component.html
- Has a nested view component to toggle `visited` state
- Choice: events are directly emitted from the HTML
 - Can also be done via small functions

```
<ul class="list-group">
  <li *ngFor="let city of cities"
    class="list-group-item"
    [class.visited]="city.visited"
    (click)="selectCity.emit(city)">
    {{ city.id }} - {{ city.name }}
    <city-visited [visited]="city.visited"
      (toggle)="city.visited = $event; toggleVisited.emit(city)">
    </city-visited>
  </li>
</ul>
```



<city-visited> View component

Again – just @Input()’s and @Output()’s.

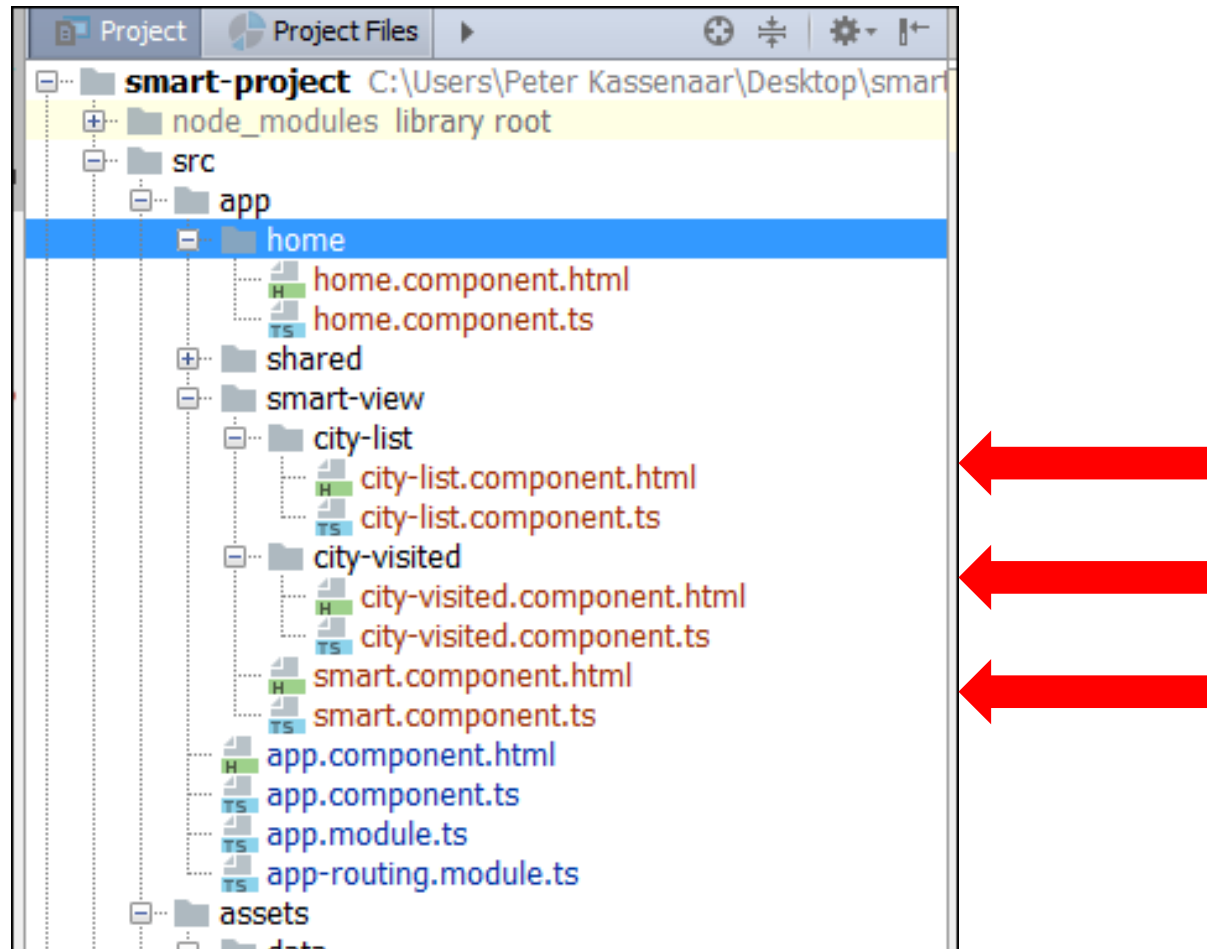
```
export class CityVisitedComponent {  
    @Input() visited: boolean;  
    @Output() toggle: EventEmitter<boolean>= new EventEmitter<boolean>();  
}
```

HTML – again: attribute binding and event binding

```
<span class="pull-right">  
    <label>  
        <input type="checkbox"  
            class="checkbox-inline"  
            [checked]="visited"  
            (change)="visited = !visited; toggle.emit(visited)"> Visited  
    </label>  
</span>
```

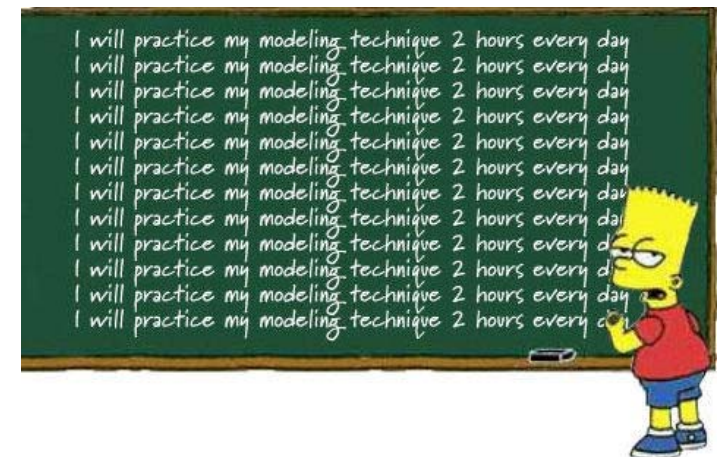
Application structure

- When following this pattern: typically more, smaller components
- The directory tree gets crowded!

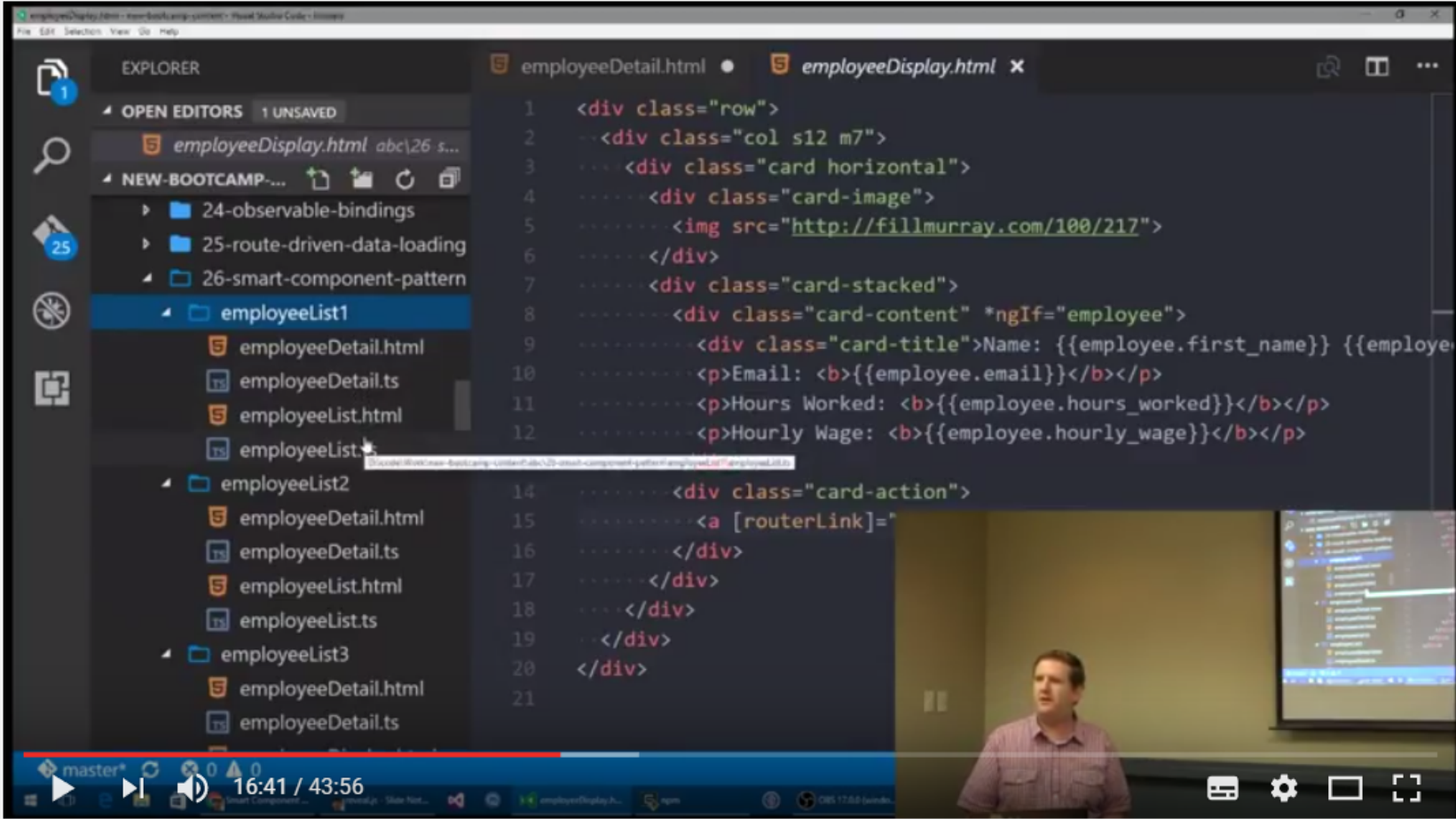


Workshop

- Study the example [../300-smart-view-components](#)
- Create two view additional components:
 - One for displaying the current city
 - One for displaying the list of visited cities
- Optional: use Observables instead of plain arrays.
 - `cities$: Observable<City>`
 - `[cities]="cities$ | async"`



More info on Smart/View components





The screenshot shows a video player interface. The video content is a presentation slide titled "Smart/View Component Patterns - St. Louis Angular Lunch - Paul Spears Jan 2017". The slide features a VS Code editor window. On the left, the Explorer sidebar shows a project structure with folders for "24-observable-bindings", "25-route-driven-data-loading", and "26-smart-component-pattern". Under "26-smart-component-pattern", there are subfolders "employeeList1", "employeeList2", and "employeeList3". The "employeeList1" folder is expanded, showing files like "employeeDetail.html", "employeeDetail.ts", "employeeList.html", and "employeeList.ts". The main editor area displays the "employeeDetail.html" file, which contains HTML code for a card component. The code uses Angular's data binding and structural directives like *ngIf. The video player controls at the bottom show the video is at 16:41 / 43:56. The presenter, Paul Spears, is visible in a small inset window in the bottom right corner of the video frame.

Smart/View Component Patterns - St. Louis Angular Lunch - Paul Spears Jan 2017

Oasis Digital Solutions Inc.

https://www.youtube.com/watch?v=ALm_JVdLT2E

**Todd Motto**
Owner, Ultimate Angular

 Follow @toddmotto 35K followers

≡

Blogs

👤

About

🎤

Speaking events


📺

Online courses


🎓

Workshops


Search posts


 **ULTIMATE ANGULAR™**


Master Angular with my online courses


 Award winning courses trusted by thousands, there's no better place to learn

Categories:




 Angular

 AngularJS

 RxJS

 JavaScript

Stateful and stateless components, the missing manual


 Oct 12, 2016  10 mins read  Edit post


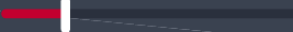
The goals of this article are to define what stateful and stateless components are, otherwise known as smart and dumb - or container and presentational components. For the purposes of the article, we'll be using Angular 2 Components to explain the stateful/stateless concepts. Bear in mind these concepts are not at all limited to Angular, and live in other libs/frameworks such as React.

Table of contents

LATEST COURSE

Angular Fundamentals



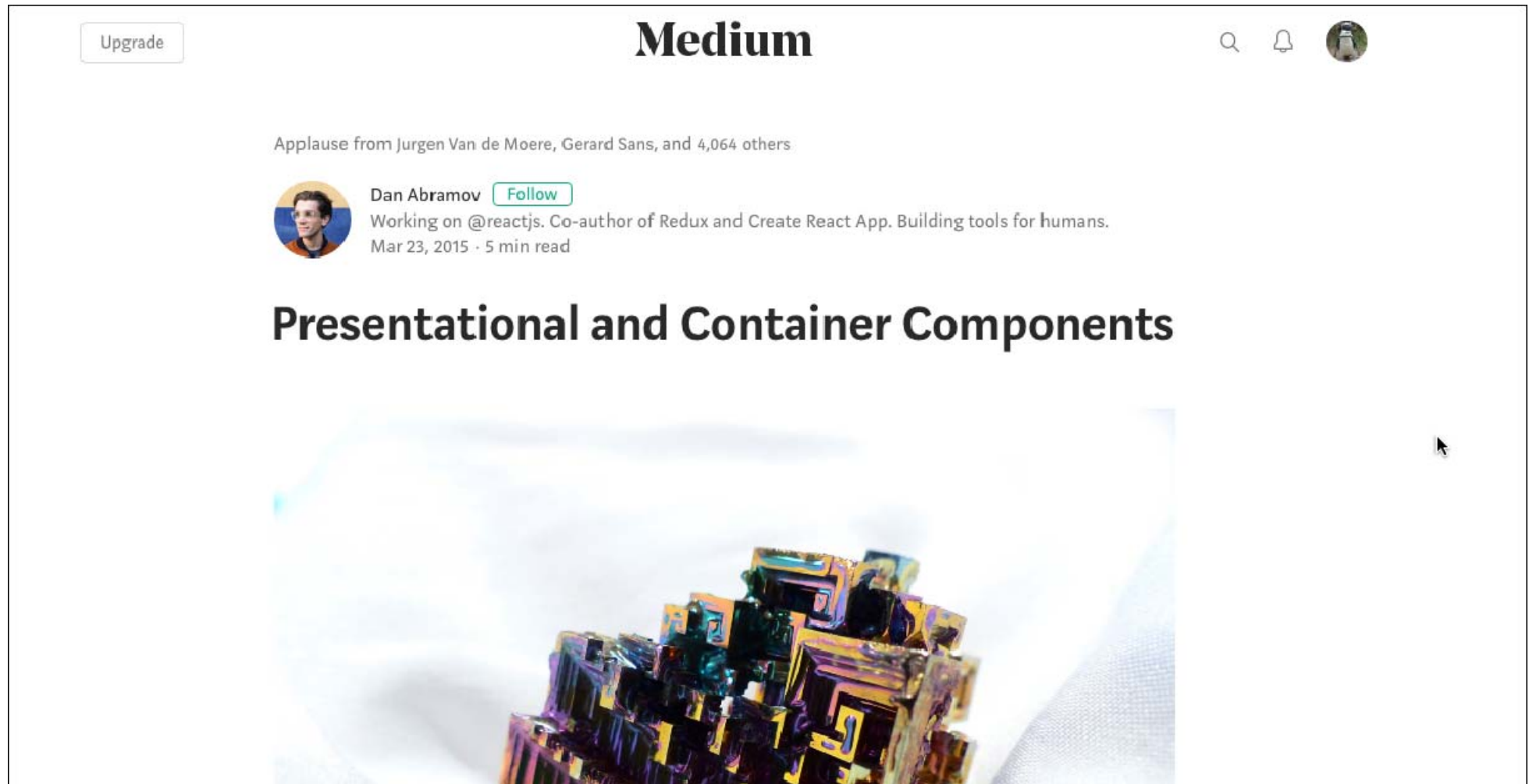
Angular v4+ award-winning courses, learn at your own pace online with me.

<https://toddmotto.com/stateful-stateless-components>

14



<http://blog.angular-university.io/angular-2-smart-components-vs-presentation-components-whats-the-difference-when-to-use-each-and-why/>



https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0