React and Angular

Side-by-side examples

John Haldeman – Hackforge – January 2018

This Talk

We'll look at impressions on ***first use*** using ***examples*** from side projects

The perspective on this talk is "Someone who is new trying to get things done"

We won't talk about:

- Theory
- "The React/Angular Way"
- Idioms
- Best Practices

Before We Begin

Some unfortunate terminology:

This presentation deals with Angular (that is Angular 2 and higher)

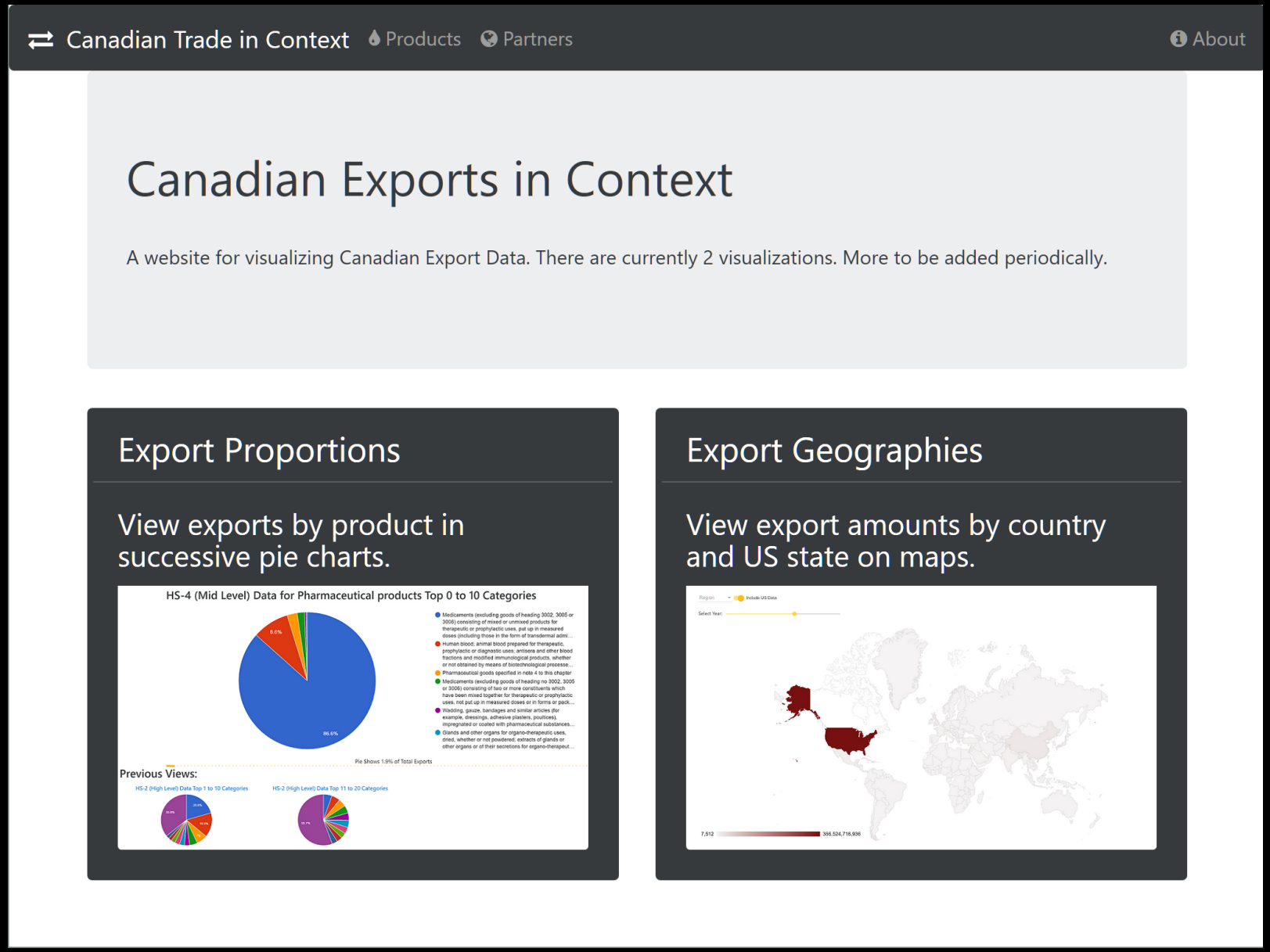
Angular 1 (AngularJS) != Angular 2+ (Angular):

- Similar in ***Spirit***
- Supported but the same people and company
- Very different from one another

Application 1

www.tradecontext.ca

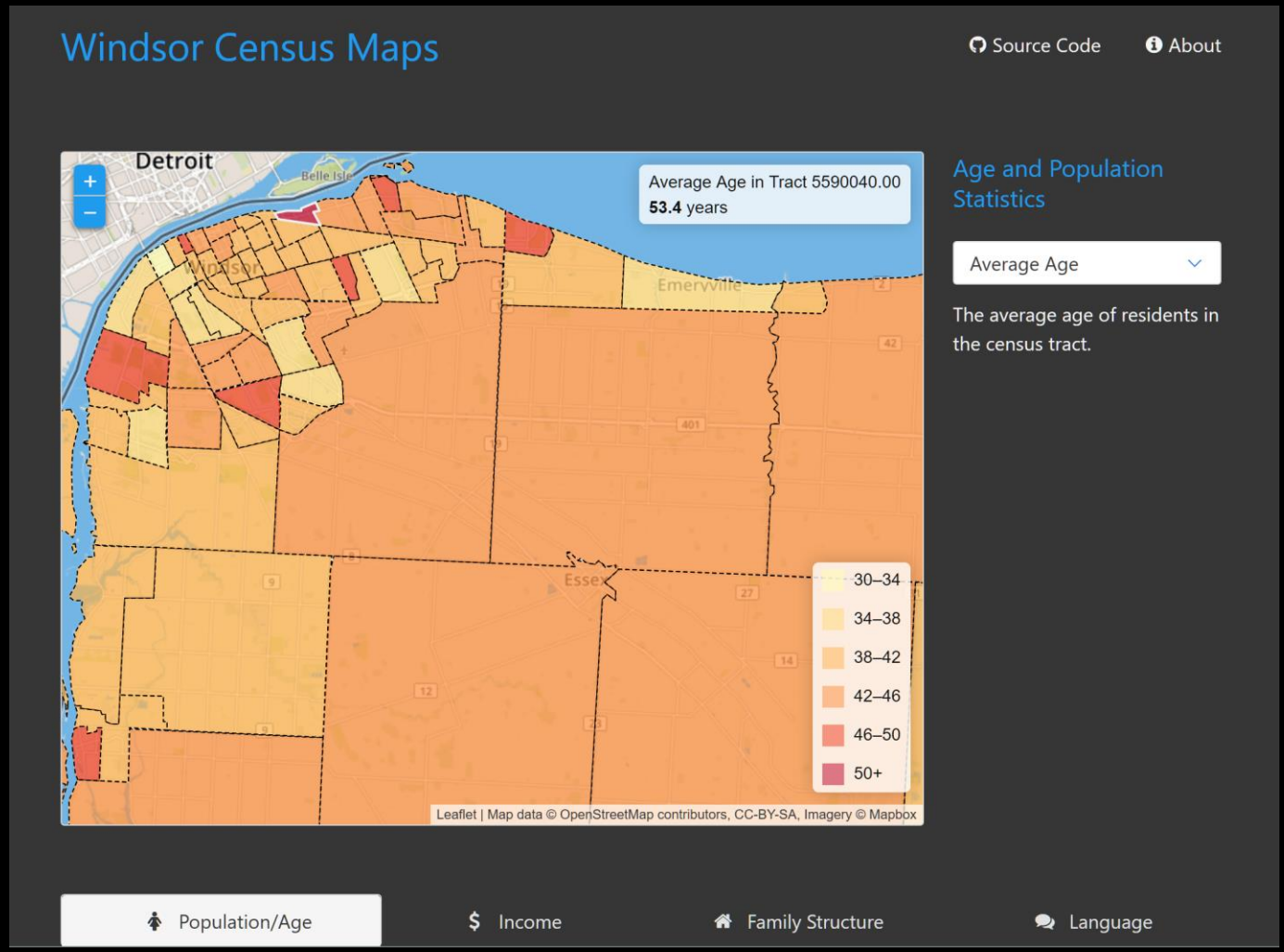
Built with Angular to visualize Canadian trade data



Application 2

www.windsorcensusmaps.com

Built with React to Display a Census Data on a Map or Windsor



Things to Look at

- Creating Components
- Defining a Component’s HTML
- Iterating through data to generate HTML
- Passing Data into a Component
- Responding to Events in a Component/Getting data out of a component
- Calling services
- Integrating with third party libraries

Creating a Component

```
import {Component,[…]} from '@angular/core’;

@Component({
  selector: 'pie-chart-list',
  templateUrl: '/app/components/pie-chart-list/templates/pie-chart-list-template.html’
})

export class PieChartList{

  constructor(public element: ElementRef) {
  }

[…]

}
```

Creating a Component

```
import React, { Component } from 'react’;
[…]

export default class Map extends Component {

  constructor(props) {
    super(props);
    this.state = {
      […]
    };
  }
[…]

}
```

Defining a Component's HTML – In a Template File (or String)

```
<row>
  <div class='col-xs-12 ellipsis'>
    <button (click)="focusChart(chartIndex)"
                type="button" class="btn btn-link ellipsis">
          <span class="ellipsis">{{chart.title}}</span>
    </button>
  </div>
 </row>
```

Defining a Component's HTML – In JSX

```
function FooterButton(props){
  return(
    <li className={getClassNameForFooterButtons(props.is_active)}>
      <a href="#" onClick={props.onClick}>
        <span className="icon">
          <i className={"fa " + props.icon}></i>
        </span>
        {props.text}
      </a>
    </li>
  )
}
```

Iterating through data to generate HTML

```
<row>
  <div class='col-lg-3 col-md-6 col-xs-12'
       *ngFor="let chart of charts;
               let chartIndex = index;
               let isFirst = first">

         […]

  </div>
</row>
```

Iterating through data to generate HTML

```
export default function Footer(props){
  return(
    <div className="hero-foot">
      <nav className="tabs is-boxed is-fullwidth">
        <div className="container">
          <ul>
            {props.footer_items().map((item, i) =>
              <FooterButton
                     onClick={props.onClick} key={i}
                     icon={item.icon} text={item.category}
                     is_active={item.category === props.selected_item.category}
              />)
            }
          </ul>
        </div>
      </nav>
    </div>
  )
}
```

Passing Data Into a Child from a Parent

```
<container>
 <row>
     […]
       <div GoogleChart
            [id]="id"
            [chartData]="data"
            [chartOptions]="chartOptions"
            [chartType]="chartType"
            [actions]="chartActions" >
        </div>
      […]
  </row>
</container>
```

Passing Data Into a Child from a Parent

```
render() {
  return (
    […]
     <div className="hero-body">
      <div className="container">
       <div className="columns">
        <div className="column is-three-quarters">
         <Map  measure={this.state.curr_measure.measure}
               measure_name={this.state.curr_measure.measure_name}
               measure_units={this.state.curr_measure.measure_units}
               measure_type={this.state.curr_measure.measure_type}
               measure_detail={this.state.curr_measure.measure_detail}
               colours={this.state.curr_measure.colours}
               />
        </div>
       </div>
     </div>
    […]
}
```

Responding to Events in a Component: Child

```
<row>
  <div class='col-lg-3 col-md-6 col-xs-12' *
        ngFor="let chart of charts; let chartIndex = index, let isFirst = first">
    <div *ngIf="chart != undefined">
      <row>
        <div class='col-xs-12 ellipsis'>
          <button (click)="focusChart(chartIndex)"
                  type="button"
                  class="btn btn-link ellipsis">
            <span class="ellipsis">{{chart.title}}</span>
          </button>
        </div>
      </row>
    </div>
  </div>
</row>
```

```
@Component({
  selector: 'pie-chart-list',
  templateUrl: '[...]pie-chart-list-template.html'
})
export class PieChartList{
  @Output() onChartFocus = new EventEmitter();

  private focusChart(index){
    this.onChartFocus.emit(this.charts[index]);
    this.charts = this.charts.slice(0, index);
  }
```

Responding to Events in a Component: Parent

```
<row>
 <div class='col-xs-12'>
    <pie-chart-list (onChartFocus)="onNewChartFocus($event)" ></pie-chart-list>
 </div>
</row>
```

```
onNewChartFocus(chart: ChartDefinition){
    this.router.navigate(["proportions", chart.url]);
 }
```

Responding to Events in a Component

Grandparent:

```
<Footer footer_items={getMeasures} selected_item={this.state.curr_category}
onClick={this.handleCategoryChange.bind(this)} />
```

Footer:

```
<FooterButton onClick={props.onClick} key={i}
  icon={item.icon} text={item.category}
  is_active={item.category === props.selected_item.category} />
```

```
function FooterButton(props){
 return(
  <li className={getClassNameForFooterButtons(props.is_active)}>
   <a href="#" onClick={props.onClick}>
    {props.text}
   </a>
  </li>
 )}
```

Note: Looking at this again, there's probably a better way to do it that would avoid having to have the parent component know to bind it's context to the subcomponent.

See: https://reactjs.org/tutorial/tutorial.html#lifting-state-up

Calling Services (1/2)

```
import { Injectable } from '@angular/core'; import { Http, Response} from '@angular/http';
import { Observable ] from 'rxjs/Observable'; import 'rxjs/add/operator/map'; import 'rxjs/add/operator/catch';

@Injectable()
export class ExportGeoService {
 […]
 getGeoData(urlPostfix:string): Observable<ExportGeoData> {
  return this.http.get(`${this.url}/${urlPostfix}`)
       .map(this.extractData)
       .catch(this.handleError);
 }
 […]
 private extractData(res: Response) {
    let body = res.json();
    return body || { };
 }
 private handleError (error: Response | any) {
  […]
 }
}
```

Calling Services (2/2)

```
import {ExportGeoService} from '/services/export_geos.service’
[…]
constructor(private exportGeoService: ExportGeoService){ […] }
[…]
getGeoData() {
    this. exportGeoService.getGeoData(this.currentURL)
    .subscribe(
      geoData => { […] }   ,
      error => { […] } ;
  }
[…]
```

Calling Services

React is a “View Library” only – Not a framework

It has no HTTP client or client library functions

The upshot: use a third party library to interact with services or built in browser capabilities

Integrating with 3rd Party Libraries

```
declare var google:any;
export class GoogleChart implements OnChanges
{
  public _element:any;
  constructor(public element: ElementRef) {
    this._element =
          this.element.nativeElement;
  }
  ngOnChanges(){
    [...]
    this.drawGraph([…]);
    [...]
  }
  ngAfterViewInit(){
    [...]
    this.drawGraph([…]);
    [...]
  }
}
```

```
drawGraph([…]) {
    function drawChart() {
      let wrapper = new
google.visualization.ChartWrapper({
          [...]
        });
        wrapper.draw();
       [...]
    }
    google.charts
          .setOnLoadCallback(drawChart);
  }
```

Integrating with 3rd Party Libraries

```
import L from 'leaflet';
import './lib/leaflet.css';
export default class Map extends Component {
  init() {
      [...]
      let map = L.map(this.mapref, config.params);
      [...]
  }
  [...]
  componentWillUnmount() {
    this.state.map.remove();
  }
  componentDidMount() {
    if (!this.state.map) this.init();
  }
  render(){
    return <div className="is-bordered" id="mapid" ref={el => this.mapref = el}> </div>
  }
}
```

Summary

- At a base level, these are pretty similar
- Angular has more in it – Tries to be a full JS framework
- React is a view library – Tried not be a framework

- Both are powerful, help you organize your javascript, and get you out of manipulating the DOM