```typescript
import { Component, OnDestroy, OnInit } from '@angular/core';
import { Category } from '../../shared/models/category.model';
import { Subscription } from 'rxjs/Subscription';
import { WindowSizeService } from '../../shared/services/window-size.service';
import { ChartSelectEvent} from 'ng2-google-charts';
import { TimelineService } from '../timeline.service';
import { CvDataHandlerService } from '../../shared/services/cv-data-handler.service';
import { CoreService } from '../../shared/services/core.service';

@Component({
  selector: 'app-ng2-timeline',
  templateUrl: './ng2-timeline.component.html',
  styleUrls: ['./ng2-timeline.component.scss']
})
export class Ng2TimelineComponent implements OnInit, OnDestroy {
  private _categories: Category[] = null;
  private _residences: any = null;
  catReadySubscription: Subscription;
  resReadySubscription: Subscription;
  minDateChangedSubscription: Subscription;
  maxDateChangedSubscription: Subscription;
  sizeChangedSubscription: Subscription;
  private _currentSize = '';
  public timelineData: any;
  public displayError = false;

  constructor(
    public windowSize: WindowSizeService,
    private _timelineService: TimelineService,
    private _cvDataHandler: CvDataHandlerService,
    public coreService: CoreService
  ) { }

  // subscribes for asynchronous data collection from
      cvData/residenceData/LangChange, min/max date change, and window Size Change
  // data gets retrieved from the server only once in the whole project
  ngOnInit() {
    this._currentSize = this.windowSize.getCurrentSize();

    this.catReadySubscription = this._cvDataHandler.getCategoriesAsync.subscribe(
      (categories: Category[]) => {
        this._categories = categories;
        if (this._residences) {
          this._loadTimelineData();
        }
    });
    if (this._cvDataHandler.isResidences() && this._cvDataHandler.isCategories()) {
      this._categories = this._cvDataHandler.getCategories();
    }

    this.resReadySubscription = this._cvDataHandler.getResidencesAsync.subscribe(
      (residences: any) => {
        this._residences = residences;
        if (this._categories) {
          this._loadTimelineData();
        }
      });
    if (this._cvDataHandler.isResidences() && this._cvDataHandler.isCategories()) {
      this._residences = this._cvDataHandler.getResidences();
      this._loadTimelineData();
    }

    this.minDateChangedSubscription = this._timelineService.minDateChanged.subscribe(
      () => {
        this._loadTimelineData();
      }
    );

    this.maxDateChangedSubscription = this._timelineService.maxDateChanged.subscribe(
      () => {
        this._loadTimelineData();
      }
    );
```

```
73
74        this.sizeChangedSubscription = this.windowSize.sizeChanged.subscribe(
75          (currentSize: string) => {
76            if (this._currentSize !== currentSize || this._currentSize === 'sm') {
77              this._currentSize = currentSize;
78              this._loadTimelineData();
79            }
80          }
81        );
82      }
83
84      // create timeline table options and final table. dataTable is created within the
            timeline service
85      private _loadTimelineData() {
86        this.displayError = false;
87        this.timelineData =  {
88          chartType: 'Timeline',
89          dataTable: this._timelineService.loadTimelineDataTable(this._categories,
                this._residences),
90          options: {
91            colors: ['#ff7200', '#3d3d3d', '#ffba44', '#ffcd77', '#ff6100'],
92            width: (this._currentSize === 'xs') ? 750 : -1,
93            height: 350,
94            tooltip: { trigger: 'none' },
95            timeline: {
96              rowLabelStyle: {fontName: 'Helvetica', fontSize: 16, color: '#000000'},
                    barLabelStyle: {fontName: 'Arial', fontSize: 12}
97            }
98          }
99        };
100     }
101
102     // send category and occurrence index for a timeline element that has been
            clicked, to the timeline service
103     chartSelect(event: ChartSelectEvent) {
104       const categoryId =
              this._timelineService.getDataTabToCateg()[event.row].categoryId;
105       const occurrenceId =
              this._timelineService.getDataTabToCateg()[event.row].occurrenceId;
106
107       if (categoryId !== 'residence') {
108         this._timelineService.setClickedItemOccurrence(
109           this._cvDataHandler.getCategoryByName(categoryId).categoryName,
110           this._cvDataHandler.getOccurrenceByIds(categoryId, occurrenceId)
111         );
112       } else {
113         this._timelineService.setClickedItemResidence(
114           this._cvDataHandler.getResidenceById(occurrenceId)
115         );
116       }
117     }
118
119     // notify that timeline table finished loading
120     ready() {
121       this._timelineService.chartReady.next(true);
122     }
123
124     // execute in case of table display error
125     error() {
126       this.displayError = true;
127       return (this.coreService.getLang() === 'eng') ?
128         'An error was encountered while drawing the table' :
129         'E\' stato incontrato un errore durante il rendering della tabella';
130     }
131
132     // public method returning a private variable
133     getCurrentSize() {
134       return this._currentSize;
135     }
136
137     // free memory from subscriptions on component destruction
138     ngOnDestroy() {
139         this.catReadySubscription.unsubscribe();
```

```
140         this.resReadySubscription.unsubscribe();
141       this.sizeChangedSubscription.unsubscribe();
142     }
143   }
144
```