```
1  'use strict';
2
3  var enableAppDirectives = angular.module('EnableAppDirectives', []);
4
5  /**
6   *
7   * @ngdoc directive
8   * @name autoActive
9   * @scope false
10  * @restrict A
11  * @element a
12  * @description
13  * Directive that appends a class to the current element, based on whether it matches
    the current route
14  *
15  */
16 enableAppDirectives.directive('autoActive', ['$location', '$timeout', function ($location,
    $timeout) {
17     return {
18         restrict: 'A',
19         scope: false,
20         link: function (scope, element) {
21
22             /**
23              * @ngdoc function
24              * @name autoActive.setActive
25              * @kind function
26              *
27              * @description
28              * Function that analyses the current route path versus the a link target.
29              * If it matches it adds the class 'sidenavlinksactive', if not it removes it
30              */
31             function setActive() {
32                 var path = $location.path();
33
34                 if (element[0].href.match(path + '(?=\\?|$)')) {
35                     element.addClass('sidenavlinksactive');
36                 } else {
37                     element.removeClass('sidenavlinksactive');
38                 }
39
40
41             }
42
43             //initiate after 500msec to make sure that element is built
44             $timeout(function(){
45                 setActive();
46             }, 500);
47
48             scope.$on('$locationChangeSuccess', setActive);
49         }
50     };
51 }]);
52
53
54 /**
55  *
56  * @ngdoc directive
57  * @name enableSectionHeader
```

```
 58   * @scope true
 59   * @restrict E
 60   * @param {string} picpath The path of the background image for the section header.
 61   * @param {string} picalt The alternative text for the header image, to be read by screenreaders.
 62   * @param {string} title The title of the section
 63   * @description
 64   * Directive that places a section header with a background picture, in alt text and the section title
 65   *
 66   */
 67  enableAppDirectives.directive('enableSectionHeader', ['$sce','$route', function($sce, $route) {
 68      return {
 69          scope:{
 70              picpath: '@',
 71              picalt: '@',
 72              title: '@'
 73          },
 74          restrict: 'E',
 75          replace: 'true',
 76          templateUrl: 'partials/templates/section-header-template.html',
 77
 78          link: function(scope) {
 79              scope.picsrc = $sce.trustAsResourceUrl('partials/'+$route.current.params.level +'/media/pics/'+scope.picpath);
 80          }
 81      };
 82  }]);
 83
 84  /**
 85   *
 86   * @ngdoc directive
 87   * @name enableVideo
 88   * @scope true
 89   * @restrict AE
 90   * @param {string} vididyt The id of the youtube video. This will bring up the youtube video player , as well as the CC attached to that video
 91   * @param {string} vididlc The id of the local video file. This id will be used for the poster image and the subtitles
 92   * @param {string} cclang The current language code. This will load the subtitles in the current portal language.
 93   * @param {boolean} localmode The mode that needs to be inserted in the page, true => local videos, false => youtube videos
 94   * @description
 95   * Directive that creates a local video player based on the localmode paramter with the video id provided and the language for the subtitles.
 96   *
 97   */
 98  enableAppDirectives.directive('enableVideo', ['$sce','$route', function($sce, $route) {
 99
100      return {
101          scope:{
102              vididyt: '@',
103              vididlc: '@',
104              cclang: '@',
105              localmode: '='
106          },
107          restrict: 'AE',
```

```
108         replace: 'true',
109         templateUrl: 'partials/templates/video-template.html',
110 
111         link: function(scope) {
112 
113             if(scope.localmode) {
114                 scope.vidurl = $sce.trustAsResourceUrl('partials/'+$route.current.params.
    level+'/media/vids/'+scope.vididlc+'.mp4');
115                 scope.vidtrack = $sce.trustAsResourceUrl('partials/'+$route.current.params.
    level+'/media/vids/'+scope.vididlc+"_"+scope.cclang+'.srt');
116                 scope.poster = 'partials/'+$route.current.params.level+'/media/pics/'+scope.
    vididlc+'__00_00_00_00.png';
117             }
118             else {
119                 scope.vidurl = $sce.trustAsResourceUrl("https://www.youtube.com/embed
    /"+scope.vididyt+"?html5=1&controls=1&autohide=0&rel=0&showinfo=0&vq=
    small&hl="+scope.cclang+"&cc_load_policy=1");
120             }
121 
122 
123         }
124     };
125 }]);
126 
127 /**
128  *
129  * @ngdoc directive
130  * @name enableAudio
131  * @scope true
132  * @restrict AE
133  * @param {string} sndid The id of the local video file. This id will be used for the
    poster image and the subtitles
134  * @description
135  * Directive that creates an audio player with the audio id provided. The player
    expects and will provide sounds tracks in 2 formats, m4a and ogg.
136  *
137  */
138 enableAppDirectives.directive('enableAudio', ['$sce','$route', function($sce, $route) {
139     return {
140         scope:{
141             sndid: '@'
142         },
143         restrict: 'E',
144         replace: 'true',
145         templateUrl: 'partials/templates/audio-local-template.html',
146         link: function(scope) {
147             scope.audiourlm4a = $sce.trustAsResourceUrl('partials/'+$route.current.
    params.level+'/media/snds/'+scope.sndid+'.m4a');
148             scope.audiourlogg = $sce.trustAsResourceUrl('partials/'+$route.current.params
    .level+'/media/snds/'+scope.sndid+'.ogg');
149         }
150     };
151 }]);
152 
153 
154 /**
155  *
156  * @ngdoc directive
157  * @name enableImage
```

```
158   * @scope true
159   * @restrict AE
160   * @param {string} picname The name of the picture to insert (relative to the media
      folder where the section is loaded from)
161   * @param {string} picalt The alternative text for the picture to be read by
      screenreaders
162   * @description
163   * Directive that creates an audio player with the audio is provided. The player
      expects and will provide sounds tracks in 2 formats, m4a and ogg.
164   *
165   */
166  enableAppDirectives.directive('enableImage', ['$sce','$route', function($sce, $route) {
167     return {
168        scope:{
169           picname: '@',
170           picalt: '@',
171        },
172        restrict: 'E',
173        replace: 'true',
174        template: '<img src="{{picsrc}}" width="100%" alt="{{picalt}}">',
175        link: function(scope) {
176           scope.picsrc = $sce.trustAsResourceUrl('partials/'+$route.current.params.level
      +'/media/pics/'+scope.picname);
177        }
178     };
179  }]);
180
181
182  /**
183   *
184   * @ngdoc directive
185   * @name enableSlideshow
186   * @scope true
187   * @restrict E
188   * @param {string} path Path to the directory where the slideshow is defined. The
      directive expects a JSON file called 'init.json' with an array of images and texts
189   * @description
190   * Directive loads a JSON config file for a slideshow based on the provided path.
191   * The template then loads the pictures and texts specified in that config file and
      creates a self-contained slideshow
192   *
193   */
194  enableAppDirectives.directive('enableSlideshow', ['$http', '$route', function($http,
      $route) {
195     return {
196        scope:{
197           path: '@'
198        },
199        restrict: 'E',
200        replace: 'true',
201        templateUrl: 'partials/templates/slideshow-template.html',
202
203        link: function(scope) {
204           scope.currentIndex = 0; // Initially the index is at the first image
205           scope.images = [];
206
207           scope.abspath = 'partials/'+$route.current.params.level+'/media/pics/slideshows
      /'+scope.path;
208
```

```
209             $http.get(scope.abspath+'/init.json').then(function(res) {
210                 res.data.forEach(function (slide) {
211                     slide.src = scope.abspath+"/"+slide.src;
212                     slide.text = slide["text_"+localStorage.lang];
213                     scope.images.push(slide);
214                 });
215                 console.log('--> json loaded');
216 
217                 scope.$watch('currentIndex', function() {
218                     scope.images.forEach(function(image) {
219                         image.visible = false; // make every image invisible
220                     });
221 
222                     scope.images[scope.currentIndex].visible = true; // make the current
    image visible
223                 });
224             });
225 
226 
227             scope.next = function() {
228                 /*scope.currentIndex < scope.images.length - 1 ? scope.currentIndex++ :
    scope.currentIndex = 0;*/
229 
230                 if(scope.currentIndex < scope.images.length - 1) {
231                     scope.currentIndex++;
232                 }
233                 else {
234                     scope.currentIndex = 0;
235                 }
236             };
237 
238             scope.prev = function() {
239                 /*scope.currentIndex > 0 ? scope.currentIndex-- : scope.currentIndex =
    scope.images.length - 1;*/
240 
241                 if(scope.currentIndex > 0) {
242                     scope.currentIndex--;
243                 }
244                 else {
245                     scope.currentIndex = scope.images.length - 1;
246                 }
247             };
248 
249         }
250     };
251 }]);
252 
253 
254 /**
255  *
256  * @ngdoc directive
257  * @name enableMoreButton
258  * @scope true
259  * @restrict AE
260  * @description
261  * Directive that creates an expandable read more section, with the text wrapped in
262  *
263  */
264 enableAppDirectives.directive('enableReadMore', [function() {
```

```
265     return {
266         scope:{
267             label: '@',
268         },
269         restrict: 'E',
270         replace: 'true',
271         transclude: true,
272         templateUrl: 'partials/templates/read-more-template.html',
273         link: function(scope) {
274             var readMoreInitialStatus = false;
275 
276             if(scope.label === undefined) {
277                 scope.label = 'read more';
278             }
279 
280             scope.isOpen = function() {
281                 if(readMoreInitialStatus === false) {
282                     return false;
283                 }
284                 else {
285                     return true;
286                 }
287             };
288 
289             scope.toggleOpen = function() {
290                 readMoreInitialStatus = !readMoreInitialStatus;
291             };
292         }
293     };
294 }]);
295 
296 
297 
298 /**
299  * @ngdoc directive
300  * @name enableGreybox
301  * @restrict E
302  * @description
303  * Ada this attribute to make 'grey box' element.
304  *   * g-type:   The type of box to create: '' (no icon) 'warning', or 'funfact'
305  * <pre><enable-greybox g-type='warning'> This is the grey box content </enable-
    greybox></pre>
306  */
307 enableAppDirectives.directive("enableGreyBox", function() {
308         var linker = function(scope) {
309             scope.icon = "";
310             if(scope.eType) {
311                 if (scope.eType === 'funfact') {
312                     scope.icon = "870-smile@2x.svg";
313                 }
314                 else if (scope.eType === 'warning') {
315                     scope.icon = "791-warning@2x.svg";
316                 }
317                 else if (scope.eType === 'story') {
318                     scope.icon = "961-book-32@2x.svg";
319                 }
320                 else if (scope.eType === 'quote') {
321                     scope.icon = "quotation.svg";
322                 }
```

```
323             }
324         };
325         return {
326             templateUrl: "partials/templates/enable-greybox-template.html",
327             link: linker,
328             restrict: 'E',
329             transclude : true,
330             scope: {
331                 eType: '@'
332             }
333         };
334     });
335 
336 /**
337  * @ngdoc directive
338  * @name enableQuickQuestion
339  * @restrict E
340  * @description
341  * Add this attribute to make an element (use a div) containing 'did you know?
    comment.
342  * The answer will be shown after clicking anywhere in the box
343  *  * e-question:  The question being asked
344  *  * e-answer:  The answer to the question
345  * <pre><enable-quick-question e-question="Did you know?' e-answer="No I didn't
    !"></enable-quick-question></pre>
346  */
347 enableAppDirectives.directive("enableQuickQuestion", function() {
348     var linker = function(scope) {
349         scope.answerState = false;
350         scope.answer = function() {
351             scope.answerState = !scope.answerState;
352         };
353     };
354     return {
355         templateUrl: "partials/templates/enable-quick-question-template.html",
356         restrict: 'E',
357         transclude : true,
358         link: linker,
359         scope : {
360             eQuestion : '@',
361             eAnswer : '@'
362         }
363     };
364 });
365 
366 /**
367  * @ngdoc directive
368  * @name enable-link
369  * @restrict E
370  * @description
371  * Add this attribute to improve on the <a> link element showing an external link icon
    .
372  * <pre><enable-link href="..."></enable-link></pre>
373  */
374 enableAppDirectives.directive("enableLink", function() {
375     var linker = function(scope) {
376         scope.linkiconpath = "img/icons/directives/link/702-share@2x.svg";
377     };
378     return {
```

```
379         template: '<a class="enablelink" href="{{href}}" target="_blank"><ng-
    transclude></ng-transclude><ng-include src="linkiconpath"></ng-include></a>',
380         restrict: 'E',
381         transclude : true,
382         link : linker,
383         scope : {
384             href : '@'
385         }
386     };
387 });
388
389 /**
390  * @ngdoc directive
391  * @name enableQuotebox
392  * @restrict E
393  * @description
394  * Add this attribute to make 'quote box' element.
395  *  * e-type:  The type of box to create: 'quote' or 'story'
396  * <pre><enable-quotebox e-type="story"> This is the quote box content </enable-
    quotebox></pre>
397  */
398 enableAppDirectives.directive("enableQuotebox", function() {
399     return {
400         templateUrl: "partials/templates/enable-quotebox-template.html",
401         restrict: 'E',
402         transclude : true,
403         scope: {
404             eType: '@'
405         }
406     };
407 });
408
409
410 /**
411  * @ngdoc directive
412  * @name enableQuiz
413  * @restrict E
414  * @description
415  * Add this element anywhere to create a quiz. Quiz questions are taken from
    database using the 'h-id'.
416  *     * e-id:       Must match the ID in the quiz database             ("id")
417  *     * e-shuffle-questions:  Shuffle the questions each time quiz is taken  (true, false)
418  *      <pre><enable-quiz></enable-quiz></pre>
419  */
420 enableAppDirectives.directive("enableQuiz", ['$http', '$route', '$timeout', '$sce',
    function($http, $route, $timeout, $sce) {
421     var linker = function(scope) {
422         var quiz;
423         scope.filePath = "";
424         scope.showLoginButton = false;
425         scope.showAlreadyPassedDownloadButton = false;
426         scope.showLanguageSwitch = false;
427         scope.inSecondLanguage = false;
428         scope.incorrectAnswers = [];
429
430         scope.abspath = 'partials/'+$route.current.params.level+'/quiz/'+scope.path;
431
432         $http.get(scope.abspath+'/init.json').then(function(res) {
433
```

```
434             if(res.data !== {}) {
435                 scope.quizpoll = res.data;
436                 console.log('--> quiz: '+scope.path+' loaded');
437 
438 
439                 // Initialise attribute variables
440                 if (typeof scope.hShuffleQuestions === "undefined") { scope.
    hShuffleQuestions = false; }
441 
442                 scope.trustResource = function getTrustedHtml(resourceUrl) {
443                     return $sce.trustAsHtml(resourceUrl);
444                 };
445 
446                 // The following functions represent a state machine controlling the quiz
    template using 'scope.state'
447 
448                 scope.chooseLanguage = function(toggle) {
449                     scope.inSecondLanguage = toggle;
450                     scope.reload(true);
451                 };
452                 scope.reload = function() {        // Load quiz from JSON, set up data
    structures
453                     quiz = scope.quizpoll;
454                     scope.state = "begin";
455                     scope.type = "radio";
456                     scope.totalPages = quiz.questions.length;
457                     scope.radioTempData = { state : -1};              // Holds the index of the
    selected radio button
458                     scope.title = quiz.title || "(placeholder title)";
459                     scope.intro = quiz.intro;
460                     scope.percentScore = 0;
461                     scope.diploma_link = "";
462                     scope.passPercent = 80;
463                     scope.summarypass = quiz.summarypass;
464                     scope.summaryfail = quiz.summaryfail;
465                     scope.image_url = quiz.image_url;
466                     scope.currentQuestion = {};
467                     scope.data = { answers: [], student_id: '', score: 0 };
468                     scope.filePath = "img/quiz/";
469                 };
470                 scope.check = function(index) {         // Update UI elements after selection
471                     if(scope.state !== 'question') { return; }
472                     if(scope.currentQuestion.type === 'checkbox') {
473                         scope.currentData[index] = !scope.currentData[index];
474                         scope.resultDisabled = true;
475                         for(var j=0; j<scope.currentData.length; j++) {
476                             if (scope.currentData[j]) { scope.resultDisabled = false; }
477                         }
478                     }
479                     else if(scope.currentQuestion.type === 'radio') {
480                         scope.radioTempData.state = index;
481                         for(var i=0; i<scope.currentData.length; i++ ) {
482                             scope.currentData[i] = false;
483                         }
484                         scope.currentData[index] = true;
485                         scope.resultDisabled = false;
486                     }
487                     scope.answer();
488                 };
```

```
489             scope.clickStart = function() {
490                 scope.start();
491             };
492             scope.start = function() {      // Set up data structure for answers
493                 scope.pageIndex = -1;
494                 scope.currentData = null;
495                 scope.responseStatus = "";
496                 scope.resultDisabled = true;
497                 scope.maxscore = 0;
498                 scope.data.score = 0;
499                 scope.data.answers = [];     // Create an array that stores answers for each question
500                 quiz.questions.forEach(function(q) {                    // Set up a 2D array to store answers for each question
501                     var answerPage = [].repeat(false, q.answers.length);
502                     scope.data.answers.push(answerPage);
503                     for(var j=0; j<q.answers.length;j++) {
504                         if (q.answers[j].correct) { scope.maxscore++ ; }          // Total of the correct answers for this quiz
505                     }
506                 });
507                 scope.next();
508             };
509             scope.answer = function() {     // Accumulate the score
510                 if(scope.currentQuestion.type === "radio") {
    // Radio on correct gains a mark. Radio on incorrect scores 0.
511                     if (scope.currentQuestion.answers[scope.radioTempData.state].correct) {
512                         scope.data.score++;                                     // Only one possible correct answer
513                     }
514                     else {
515                         scope.incorrectAnswers.push(scope.currentQuestion.text);
516                     }
517                 }
518                 else if(scope.currentQuestion.type === "checkbox") {
                        // Checking an incorrect box loses a mark. Checking a correct box gains a mark. Not checking a correct or incorrect box does nothing.
519                     for(var j=0; j<scope.currentQuestion.answers.length;j++) {
                      // Multiple possibly correct answers, convert to boolean before comparing
520                         if(scope.currentQuestion.answers[j].correct && scope.currentData[j]) {
521                             scope.data.score++;
522                         }
523                         else if(scope.currentQuestion.answers[j].correct === false && scope.currentData[j] === true) {
524                             scope.data.score--;
525                             scope.incorrectAnswers.push(scope.currentQuestion.text);
526                         }
527                         else {
528                             scope.incorrectAnswers.push(scope.currentQuestion.text);
529                         }
530                     }
531                 }
532                 var theScore = Math.floor(scope.data.score / scope.maxscore * 100);
533                 scope.percentScore = theScore < 0 ? 0 : theScore;
534
535                 $timeout(function() {          // Safari will not reliably update the DOM if
```

```
535  not using $timeout
536                      scope.state = "result";
537                  }, 0);
538
539              };
540
541              scope.next = function() {       // Prepare for the next question
542                  scope.state = "question";
543                  scope.pageIndex++;
544                  scope.resultDisabled = true;
545                  scope.radioTempData.state = -1;
546                  if(scope.pageIndex === scope.totalPages) {
547                      scope.state = "end";
548                  }
549                  else {
550                      scope.currentData = scope.data.answers[scope.pageIndex];
551                      scope.currentQuestion = quiz.questions[scope.pageIndex];
552                      scope.type = scope.currentQuestion.type;
553                      if(scope.currentQuestion.image_url !== "") {
554                          scope.image_url = scope.abspath+'/'+scope.currentQuestion.
     image_url;
555                      }
556                      else {
557                          scope.image_url = "";
558                      }
559                      //scope.image_url = (scope.currentQuestion.image_url !== "") ? scope.
     filePath + scope.currentQuestion.image_url : "";
560                  }
561
562              };
563              scope.reload(false);
564          }
565
566      });
567
568       // true = start test after loading
569  };
570  return {
571      restrict: 'E',
572      link: linker,
573      templateUrl: "partials/templates/enable-quiz-template.html",
574      scope:{
575          path: '@'
576      },
577  };
578 }]);
579
580 /**
581  * @ngdoc directive
582  * @name enable-file-link
583  * @restrict E
584  * @description
585  * Add this attribute to improve on the <a> link element showing an external link icon
     .
586  * <pre><enable-link href="..."></enable-link></pre>
587  */
588 enableAppDirectives.directive("enableFileLink", function() {
589     var linker = function(scope) {
590         scope.linkiconpath = "img/icons/directives/link/702-share@2x.svg";
```

```
591     };
592     return {
593         template: '<a class="enablelink" href="partials/files/{{href}}" target="_blank"><
        ng-transclude></ng-transclude><ng-include src="linkiconpath"></ng-include></a>',
594         restrict: 'E',
595         transclude : true,
596         link : linker,
597         scope : {
598             href : '@'
599         }
600     };
601 });
```