

Video Player Documentation

Files: script2.js

(Note: the line numbers may not be exactly accurate. When searching for a section of code it's probably best to Ctrl+f).

This file included all of the front-end functionality for the player. This includes the video player and controls, all even listeners for the creation of labels and annotations.

Video player functionality: Lines 141 – 310

Create New Annotation: Lines 311 – 321 & 449 – 498

Within lines 311 – 321 exist the event listener for the CREATE NEW LABEL BUTTON once it is pressed these lines hear it and call the function **createNewLabel()** which is located on 449.




Once CREATE NEW LABEL pressed code below executes

```
311     newLabel = document.getElementById('create-new');
312
313     //Create New Label is pressed.
314     newLabel.addEventListener('click', function() {
315
316         if(busy1 == false)
317         {
318             busy1 = true;
319             createNewLabel(); //Creates box to select color and
320         }
321     });
```

Submit Button Pressed: Lines 324 – 340 & 502 - 561

Once label name and color selected press SUBMIT to execute creation of the new label. The event listener on 329 will get label name and color, then pass them to the function **submitLabel()** on line 502.



```
324     newBox = document.getElementById('new-new'); //container holding all new labels
325
326     //The next few event listeners use "event delegation" since the items created are
    //done so dynamically within the browser. if(e.target && e.target.id == "<id of element
    clicked>" signifies event delegation);
327
328     //Once name is entered, color is picked and Submit is pressed
329     newBox.addEventListener('click', function(e){
330         if(e.target && e.target.id == "submit") //Event delegation
331         {
332
333             color = document.getElementById("colorPick").value;
334             label = document.getElementById("namebox").value;
335
336             loadlock = false;
337             submitLabel(color, label); //New label is made using color and label name
338
339         }
340     });
```

Create new annotation: Lines 345 – 370 & 564 – 676

This event listener waits for the CREATE NEW ANNOTATION button to be pressed, which exists on the newly created label. Once the button is pressed the code below is executed. The event listener will call the **createNewAnnotation(color, label, kids, Grandparent Element)** on line 564, which will create the box which allows you to create annotation data.



```
345 ▼ labelContainer.addEventListener('click', function(e) {
346     var num = 0;
347     if(e.target && e.target.id == "addNew") //Event delegation
348     {
349         color = e.target.parentElement.style.backgroundColor;
350         label = e.target.parentElement.id;
351         var boxInd;
352
353     if(busy2 == false) {
354         busy2 = true;
355         var numibox = document.getElementsByTagName('tot');
356
357         //Loop is for the numbers on the annotation box: skin:1, skin:2, etc.
358     for(var i = 0; i < numibox.length; i++) {
359     if(e.target.parentElement == numibox[i]) {
360         boxInd = i;
361     }
362     }
363
364     var kids = e.target.parentElement.parentElement.children;
365     kids = kids.length;
366     //console.log(kids);
367     createNewAnnotation(color, label, kids,
368         e.target.parentElement.parentElement);
369     }
370     });
```

The if statement on 353 exists in this and other functions so that the user must create one annotation before they move onto the next

```
353 ▼ if(busy2 == false) {
354     busy2 = true;
```

The code from 355 – 365 exists to determine the indices of the annotation that is created from the push of the CREATE NEW ANNOTATION button.

```
355         var numibox = document.getElementsByTagName('tot');
356
357         //Loop is for the numbers on the annotation box: skin:1, skin:2, etc.
358         for(var i = 0; i < numibox.length; i++) {
359             if(e.target.parentElement == numibox[i]) {
360                 boxInd = i;
361             }
362         }
363
364         var kids = e.target.parentElement.parentElement.children;
365         kids = kids.length;
```

Most of the **createNewAnnotation()** function sets up the new annotation box, it also creates the box under the correct label. It does so using the GRAND PARENT ELEMENT passed to the function (the 4th parameter). On line 673 the function **drawEditAnnotation(color, label)** is called. This is the function that allows the user to create the annotation rectangle as well as set begin and end times.

The **drawEditAnnotation()** function is very important as this it contains the main functionality of the application.

drawEditAnnotation(): Lines 679 – 717

Lines 682 – 688: Event listener which allows user to set BEGIN time



```
682         //Set begin time
683         btAssign.addEventListener('click', function(){
684
685             getStartTime = video.currentTime;
686             getStartTime = getStartTime.toFixed(2);
687             stringStartTime = getStartTime.toString();
688             beginTime.innerHTML = stringStartTime;
```

Lines 693 – 701 Do the same thing but for the end time.

```
693 ▼ endAssign.addEventListener('click', function(){  
694  
695  
696     getEndTime = video.currentTime;  
697     getEndTime = getEndTime.toFixed(2);  
698     stringEndTime = getEndTime.toString();  
699     endTime.innerHTML = stringEndTime;  
700  
701 });
```

Before the user can finalize the annotation they must draw the annotation box on the canvas. Look at the example below.



Once the BEGIN and END times are set and the user has drawn the annotation box, they can then finalize the annotation.

Lines 703 – 715

These lines provide event listeners that actually draw and resize the annotation box.

```
703 //Mouse on canvas will show cursor as crosshair
704 ▼ canvas.addEventListener('mouseover', function(){
705     canvas.style.cursor = "crosshair";
706 } );
707
708 //On canvas click to begin drawing annotation box
709 canvas.addEventListener('mousedown', mouseDown);
710
711 //End drawing annotation box
712 canvas.addEventListener('mouseup', mouseUp, false);
713
714 //Size annotation box
715 canvas.addEventListener('mousemove', mouseMove, false);
```

Finalize Annotation: Lines 373 – 387

Once the user has set the begin and end times and drawn the annotation box they will press the *FINALIZE* button which will store the data by executing the code below. On line 381 the function **finalizeNewAnno(color)** is called.

```
373 //Finalize is pressed
374 ▼ labelContainer.addEventListener('click', function(e) {
375
376 ▼     if(e.target && e.target.id == "finalize") {
377         busy1 = false;
378         busy2 = false;
379
380         //Data is stored in database
381         finalizeNewAnno(e.target.parentElement.style.backgroundColor);
382         var dad = e.target.parentElement;
383         dad.parentElement.removeChild(nAB);
384         discardAnnotation();
385     }
386
387 });
```

The **finalizeNewAnno()** function updates the database with the annotation data the user just created.

```
719 //Finalize annotation by storing data in database
720 ▼ function finalizeNewAnno(color) {
721     var x, y, w, h;
722     x = currentAnnotation.x;
723     y = currentAnnotation.y;
724     w = currentAnnotation.w;
725     h = currentAnnotation.h;
726
727     var newAnnotationKey = database.ref().child('annotations').push().key;
728     console.log(newAnnotationKey);
729     var newAnnotationStatus = firebaseUpdate(newAnnotationKey, newAnnotationBox.id,
        getStartTime, getEndTime, x, y, w, h, videoName, color);
730 }
```

Delete Annotation: Lines 389 – 407

This event listener deletes an annotation. It does so when the user clicks the “X” on the annotation.



```
389 //Delete annotation
390 ▼ LabelContainer.addEventListener('click', function(e) {
391 ▼     if(e.target && e.target.id == "delete") {
392         var thisid = e.target.parentElement.id;
393         var kids = e.target.parentElement.parentElement.children;
394
395 ▼         for(var i = 0; i < kids.length; i++) {
396 ▼             if(kids[i].id == thisid) {
397                 kids[i].parentElement.removeChild(kids[i]);
398             }
399 ▼             else if(kids[i].id == "nAB") {
400                 kids[i].parentElement.removeChild(kids[i]);
401             }
402             busy2 = false;
403         }
404
405         removeAnnotation(thisid);
406     }
407 });
```

Display finalized annotations:

Finalized annotations will minimize to reduce clutter. Once they have been minimized if the user wishes to see the annotations created they can click on the annotation (anywhere but the X on the example below) and the annotation box will display on the canvas where it was drawn. Also, the video current time will jump to the BEGIN time that the user set for the annotation.



```
411 ▼      labelContainer.addEventListener('click', function(e) {
412          if(e.target && e.target.name == "nabox")
413          {
414
415              if(busy1 == false && busy2 == false)
416              {
417                  var myid = e.target.id;
418
419                  var annotationPromise = database.ref().child(videoName + '/annotations
+ myid).once("value")
420                  .then(function(snapshot) {
421                      return snapshot.val();
422                  });
423                  annotationPromise.then(function(key) {
424                      var annotationDetails = database.ref().child('/annotations/' +
key).once("value")
425                      .then(function(snapshot) {
426                          return snapshot.val();
427                      })
428                      .then(function(details) {
429                          showAnno(details.color, details.x, details.y, details.w,
details.h, ctx);
430                          video.currentTime = details.start;
431                      });
432                  });
433              }
434          }
435      });
```