

Overview

For the first practical we were asked to build a website that presents a hitlist of popular Youtube videos. The user can select a video returned from the search that will play embedded player. The data about the videos is stored on the server side in Apache Solr, which provides the search functions.

The code we wrote had to be written as a single-page application that refreshes data dynamically from the server using AJAX calls.

Design Basic

A template was provided which contained three files: index.html, hitlist.js and hitlist.css, which we had to change and extend. The main task was to extend the hitlist.js file to add the required functionality to our website.

The main feature was implement the search feature, which sends a request to the server then binds the results to elements in the html document. So that the webpage **doesn't** have to be refreshed after its loaded, we were required to make AJAX calls, allowing the JavaScript code in the web browser to access data on the server and display it. In order to implement this functionality, I decided to use the AngularJS framework which makes making AJAX calls easier.

I decided to create an application that has an explicit controller, this is created in the context of an Angular Module. The controller is where the majority of the JavaScript code lies. The Angular controller allows the data in the Angular application to be controlled as the controllers are regular JavaScript Objects.

The Angular controller contains various variables which are set to the attributes of the videos returned and functions. The first function in the controller is the **"submitSearch"** function. This function first of all call an external function **"stopVideo"** which clears the **"src"** attribute of the iframe tag in the HTML file. This ensures that no videos continue playing after new searches are made. **The "stopVideo" function was found at <http://stackoverflow.com/questions/15424910/angularjs-access-scope-from-outside-js-function>**. Next the function checks that there the search input has not been left empty, if it has an alert is sent to the user. Otherwise if a valid string has been entered, the solr URL query is constructed with the input string and the sorting parameters. The program then preforms various checks on the date inputs fields.

First the **"angular.isDate"** function is used to check if either the startDate, endDate or both fields contain valid dates.

- If an end date and a start date have been entered a check is preformed to ensure the start date is not after the end date and the appropriate parameters are appended to the query URL.
- If a start date is entered and no end date is entered the start date is appended to the URL and **"TO NOW"** is also appended to the URL. This means all the videos posted from the start date to the current date will be returned.
- If an end date is entered and no start date is entered the start date is set to the date **"2000-01-01"** which is before **YouTube's** creation and therefore will encamp all videos the end date is then appended to the end of the URL.
- Finally, the end of the URL is added.

Once the URL has been created the AJAX call is made and the array that is sent back is filtered and duplicates are removed. Using the ng-repeat directive the list in the HTML document is populated with the elements of the filtered array.

The next function is the **"playVideo"** function, which is called when a user click on a video from the list, using the ng-click directive. When this function is called, the JavaScript access the DOM and changes the display properties of the elements, allowing for the video section to be displayed, moreover the video title and description are also displayed. Finally, the **"src"** attribute for the iframe tag is updated to the URL of the chosen YouTube video.

Design Extension

My extensions include:

- Restricting the max date for the date input boxes to the current date
- Getting the views for each video
- Using the YouTube Data API to load videos and detect video events
- Adding a playlist functionality
- Automatically playing videos from the playlist one after another
- Adding a right click directive for playlist functionality
- CSS Styling and responsive design

I have added a function, which means the maxdate of the date picker input boxes is set to the max date. **I found this function at <http://stackoverflow.com/questions/32378590/set-max-date-to-today-for-date-input>** This just

enforces that valid dates can only ever be entered and therefore means less checks have to be performed on the date input boxes.

I have added a function “**getJSON**”, which takes in “**videoID**” as a parameter. This function uses the jQuery API and is used to get JSON data using an AJAX HTTP GET request. The function is used with the YouTube Data API which using a URL constructed with the video id handed in and my API key, returns a JSON object containing all the information about the given video. Using this method, I have been able to retrieve the number of views for a given video. These statics is then displayed on the webpage bellow the video description.

(“Youtube Player API Reference For Iframe Embeds | Youtube Iframe Player API | Google Developers”) Source

The main extension I’ve implemented is the use of the YouTube Data API all the code using this API could not be in the angular controller. The use of this API changes the way YouTube videos are loaded and embedded.

In order for the API to work correctly the program uses DOM modification to download the API code, which ensures the code is retrieved asynchronously, this in turn enables asynchronous downloads.

I’ve decided to keep the iframe tag in the HTML file as this allows me to update the “**src**” attribute and therefore change the embedded video. It should be noted that because of this I have had to add on two extra parameters for the video URL, these are:

- An “**origin**” parameter, specifying the URL scheme and full domain of your host page, this is simply an extra security measure
- And finally a “**enablejsapi**” parameter which is set to 1, this enables the player to be controlled via IFrame or JavaScript Player API calls.

The first function using the API is the “**onYouTubeIframeAPIReady**” function. This function defines a global variable, “**player**”, which refers to the video player which is being embedded. The function then constructs the video player object. This function takes in a video item as a parameter, this is used to get the attributes for the embedded video such as the title, description and ID. The reason I have done this so that the function can manipulate the DOM appropriately and add the video title and description, outside of the AngularJS controller. The next function is the **onPlayerReady** function will execute when the **onReady** event fires and will automatically play the mended video when executed.

Finally the last function is the using the API is the “**onPlayerStateChange**”. The API will call the **onPlayerStateChange** function when the player’s state changes, which may indicate that the player is playing, paused, finished, and so forth.



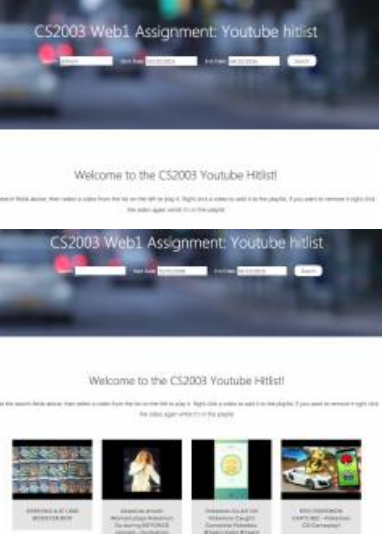
The next extension I’ve added is a playlist functionality which allows the users to right click on videos returned by the search, to add them to a playlist. I’ve made a global array which stores all the videos added to the playlist so that all the functions can access the array.





The controller has had three new methods added “**clearPlaylist**” which just resets the array, “**removeFromPlaylist**” which takes in a “**video**” object as a parameter, checks if the given video is in the playlist array and if so removes it and finally the “**addToPlaylist**” method which also a “**video**” object as a parameter and checks if the given video is in the playlist array, if so the video is not added again, otherwise the new video is added. The HTML document contains a list which stores the videos in the playlist and displays the video thumbnail and the video title. Users can click on videos in the playlist to play them alternatively the “**onPlayerStateChange**” function has allowed me to automatically load and play videos when the current video has finished. Once the video has been loaded into the iframe it is removed from the playlist. Videos can also be removed from the playlist by right clicking their icon in the playlist list. In order to add right click functionality I had to make a new angular directive called “**ng-right-clicked**”. I found the code to do this at <http://stackoverflow.com/questions/15731634/how-do-i-handle-right-click-events-in-angular-js>.



Finally, I have used a Dreamweaver bootstrap template to style my website and make it responsive. I also used the Google Grid Gallery blueprint in order to style the way search results are displayed. The code for this styling can be found here: <http://tympanus.net/codrops/2014/03/21/google-grid-gallery/>.

Testing

I have decided to do the testing on the extension so that it encompassed all the features of both the basic requirement and the extension.

Test Description	Expected Output	Actual Output	Comment
Search for videos without any dates	A list of videos relating to search are returned and the default content is displayed		Works as expected
Search for videos with start and end date	The first search where the date range is a day should yield no results the second where the date range is 16 years should yield results		Works as expected
Search for videos with just start date	The first search where the start date is today's date should yield no results. The second search where the start date is 16 years ago should return a list of videos should be displayed where their upload date is later than the input start date		Works as expected

Search for videos with just end date	The first search where the end date is 16 years ago date should yield no results. The second search should where the end date today should return a list of videos should be displayed where their upload date is later than the input start date		Works as expected
Leave all fields blank and search	An alert should come up		Works as expected
Set Start date after end date	An alert should come up		Works as expected
Add a video to the playlist	A video was added to the playlist and the playlist section was displayed		Works as expected
Right click video when playlist empty	The video should automatically play and not be added to the playlist	The video was automatically played and wasn't added to the playlist	
Remove a video from the playlist	The video should be removed from the playlist and no longer be displayed in the list	The video was removed from the playlist and was no longer displayed in the list	Works as expected

Clear the playlist	The playlist should be cleared and no videos should be displayed in the list	The playlist was cleared and no videos were displayed in the list	Works as expected
Wait for videos to play automatically	When videos are in the playlist and video is being played, once the currently playing video finishes the next video in the playlist should be automatically loaded. The automatically loaded video should then be removed from the playlist	When the currently playing video finished the next video in the playlist was automatically loaded. The automatically loaded video was then be removed from the playlist	Works as expected
Check videos are loaded when the page is loaded	A query is made with "*:*" which will return the most popular videos, these should be displayed.		Works as expected
Check videos stop playing when a new search is made and the playlist is empty	The currently playing video should stop	The current video stopped playing	Works as expected
Check videos keep playing when a new search is made and the playlist isn't empty	The video kept playing the playlist remained as it was		Works as expected

Check the video information is correct	The view counter, title and description should all be correct	The view counter, title and description were all correct	Works as expected
Check the video information updates when the next video is automatically played	The video information should change upon new videos being loaded, whether their chosen from the search results, the playlist or automatically loaded	The video information correctly changed upon new videos being loaded, when chosen from the search results, the playlist and when automatically loaded	Works as expected
Add same video twice to the playlist	Only one instance of the video should be added to the playlist	Only one instance of a video could be added to the playlist	Works as expected
Remove all the items from the playlist	Each item should be successfully removed	Each item was successfully removed	Works as expected

Evaluation

After testing my program, I pleased to say it works as expected and is able to gracefully display YouTube videos. I'm also pleased to report that all the extensions work as expected.

Conclusion

In conclusion I'm satisfied with end website, it is able to carry out the required tasks with ease and is able to handle erroneous data without crashing. However, if I had time I would like to first of all improve the code layout as it quite messy and needs refactoring. I would also like to properly style the interface and make it smoother.

"Youtube Player API Reference For Iframe Embeds | Youtube Iframe Player API | Google Developers". Google Developers. N.p., 2016. Web. 4 Oct. 2016.