

SE31520
Developing Internet-Based Applications

Enhancing the CS-Alumni Application

Submission Due 4pm Monday, Decemeber 8, 2014

Thomas Mark Rosier (THR2)
110113188

Contents

Document Summary	3
Application Design	3
CS-Alumni Chrome Browser Extension	3
Rational	3
Module Diagrams	4
Program Operation	5
CS-Alumni Rails Application	10
New Additions	10
Modifications	10
Communication Between Applications	10
RESTful Web Interfaces	10
Data Flow Diagrams	10
Application Testing	11
QUnit Tests	11
CS-Alumni Chrome Browser Extension	14
HTML	14
CSS	14
JavaScript	14
Chrome Extensions API	14
Bootstrap	14
jQuery	14
Moment.js	15
Alertify.js	15
Qunit	15
CS-Alumni Rails Application	15
Ruby	15
Ruby on rails	15
Qunit	15
Communication Between Applications	15
RESTful Interfaces	15
CORS	15
Basic Authentication	15
Evaluation	16
Something Extra	17

Document Summary

Application Design

CS-Alumni Chrome Browser Extension

Rational

My rational behind deciding to developing a browser extension for this project, was that they have been a technology that I have been following for a while and I have not seen them used that regularly for small little applications that interface with a bigger application.

Another reason for my choice to develop a browser extension was that I wanted to learn about implementing some of the more advanced features within the Chrome SDK for example the use of 'Background Pages' which are used to keep a task running in the background while the extension is closed.

One of the newer features within the Chrome SDK had also caught my attention in the weeks before starting this assignment was the use of Chrome desktop notifications where you can create a notification within your application that will be displayed to the end user via the Chrome UI's notification draw.

I wanted to see if it was possible to encapsulate some of the key management features of the CS-Alumni application into an attractive and well presented browser extension that also notified the user of any new broadcasts that had been made.

Why did I choose to develop a Chrome extension over a Firefox, Safari or Internet Explorer, the main reason was due to Chris Loftus's preference of Chrome over the other browsers. In addition to that I felt that due to developing for Chrome also meant that the code will work without modification in Opera this a bonus in the real world as it means we can support a greater population of users without investing more money into development.

I also feel that Chrome has a much nicer development environment for developing browser extensions over its competitors, along with having some newer and more unique features that allow the Chrome based extension to stand out.

If this was a real world product it would not take that much resources to port the extension to work on Firefox and Safari aswell but this out of the scope of this assignment.

Module Diagrams

Program Operation

When the user opens the browser extension they will be presented with the home screen, this shows a small blurb about the application along with a disclaimer.

Due to this being an assignment this is essentially a development version of the browser extension thus I have included a link to the Unit tests on the home page to make it easily accessible during the development and testing of the application, if this extension was intended to be released as production code this link would be removed or turned off within the configuration.

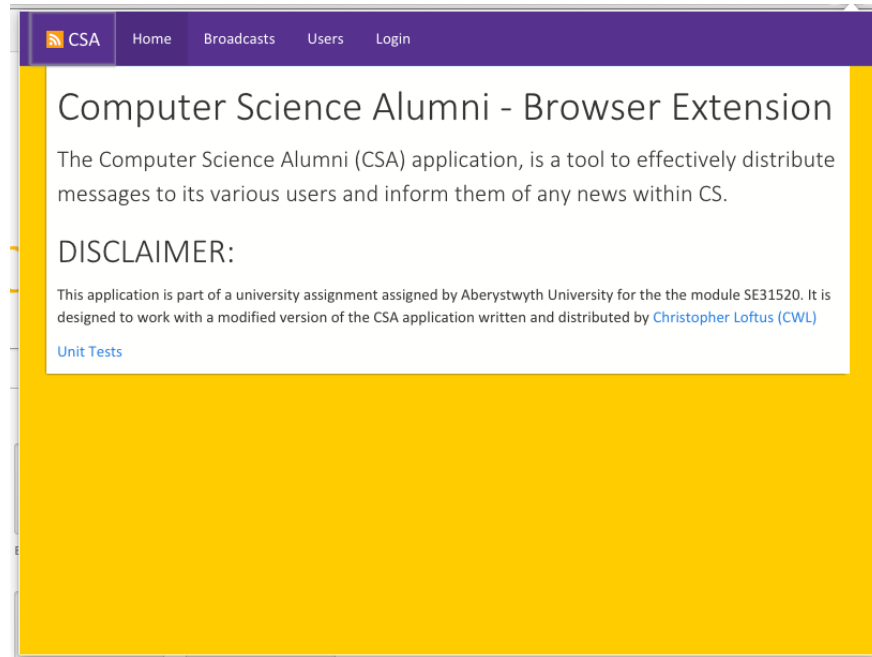


Figure 1: The screenshot above is the landing screen for the browser extension.

From the home page the user can navigate to the 'Broadcasts' tab, in the figure captioned there is no broadcasts available to the user so they are show a message that says that there is no broadcasts.

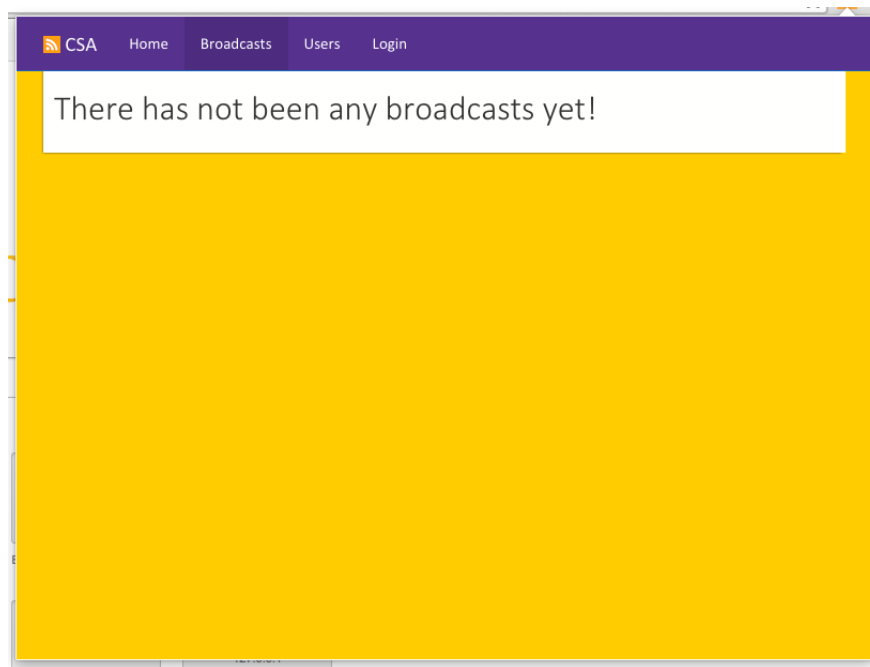


Figure 2: Above we see the broadcasts screen with no broadcasts available.

Again below we see the 'Broadcasts' tab this time we see how the page appears when there is broadcasts.

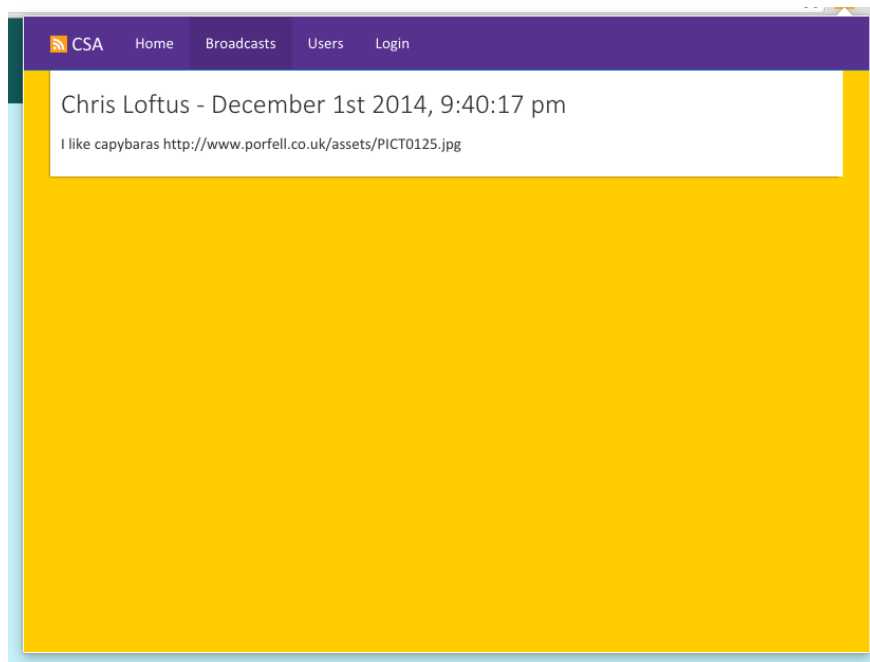


Figure 3: This is what the populated broadcast screen looks like.

This is the 'users' tab of the browser extension here the user can view a specific users details and edit there information if it is deemed appropriate.

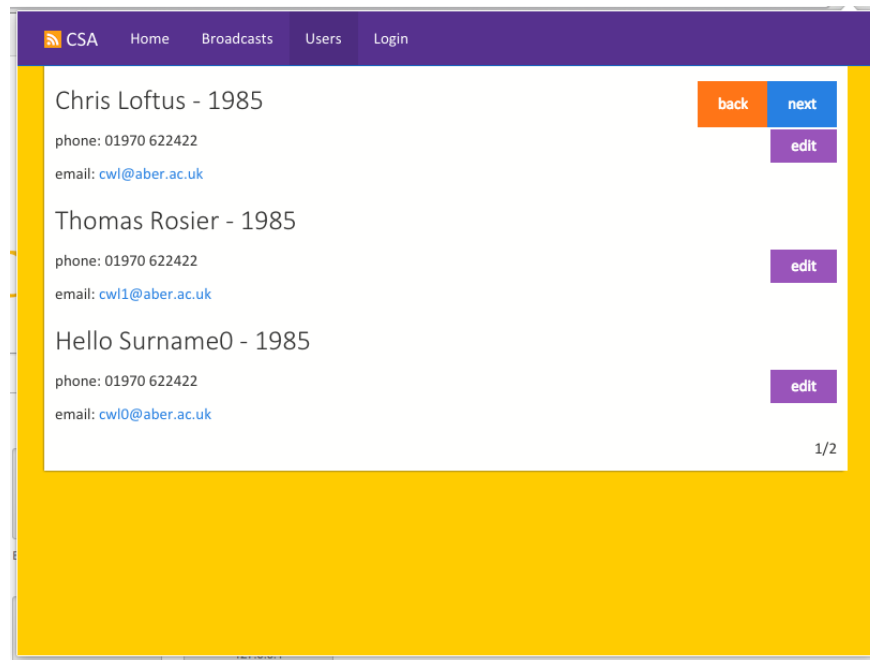


Figure 4: Here we have a screenshot of the user's screen.

When the edit button has pressed on the a specific user it shows this modal window that allows the user to edit the details that are kept on the current user they are editing.

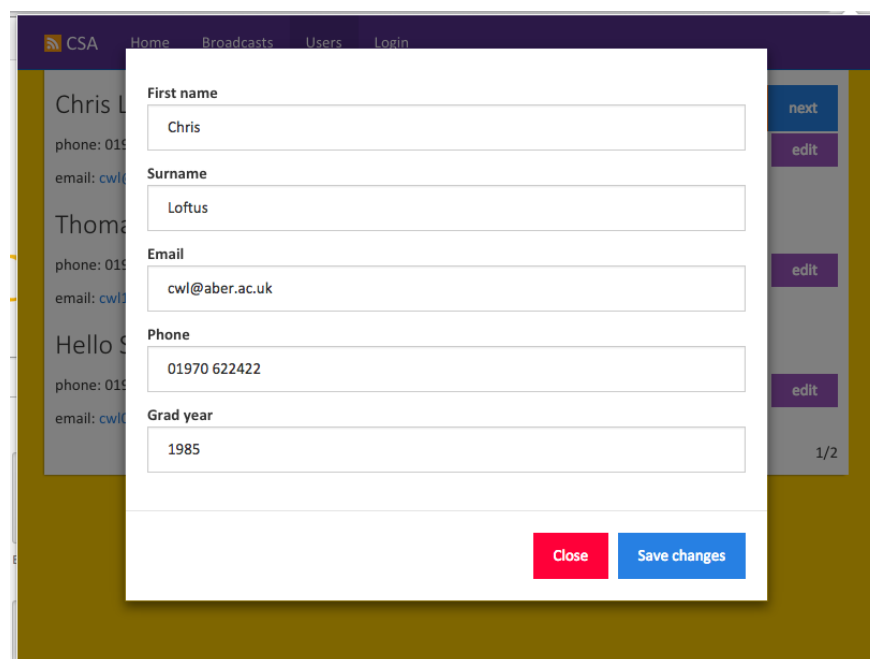


Figure 5: Above we have a screenshot showing the modal window we used to edit the user information.

The 'Login' tab is where the user enters their login credentials that will be used to authenticate the user against the CS-Alumni application.

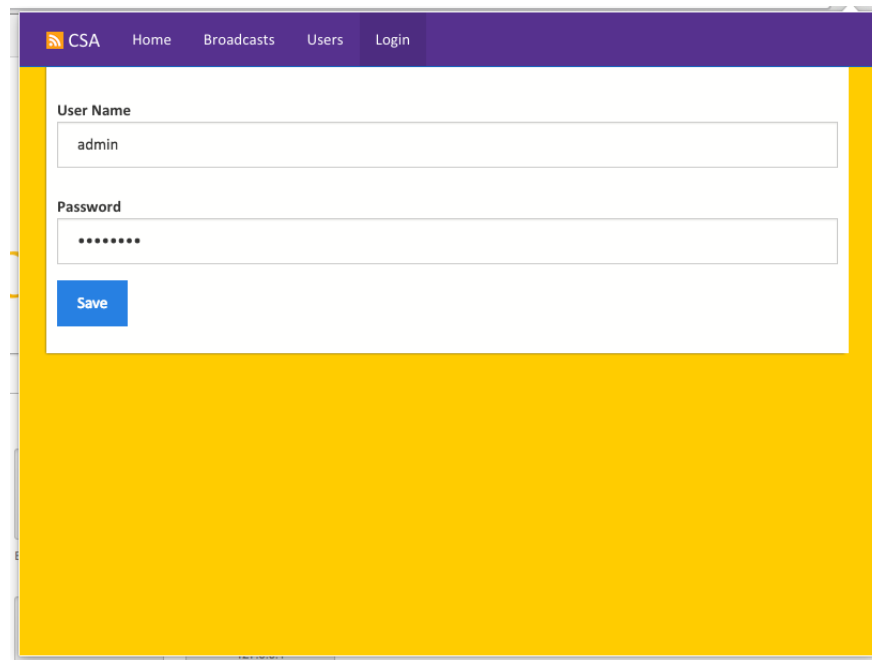


Figure 6: This is a screenshot of the login screen.

In the screenshot below I have created a new broadcast within the CS-Alumni application to showcase the notification features within the browser extension.

Note: The destination list differs from the standard CSA application. They have been modified to remove the networks that do not work along with adding in the functionality to target the browser extension specifically.



Figure 7: Creating a broadcast within the CSA application.

Confirmation of the broadcast being created with the parameters that I gave on the previous screenshot is pictured in the screenshot that has been included below.

You can see in the list of feeds that it includes the feed named 'Extension' which signifies that we want to alert users that are using the browser extension.

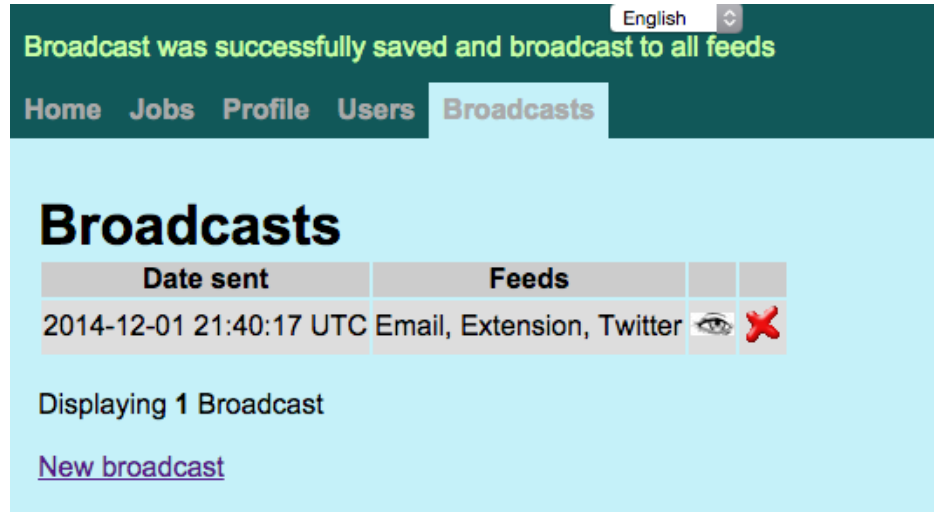


Figure 8: Here is the confirmation that a new broadcast has been created.

Here is shown a Chrome Notification for the broadcast that we just created, this notification will only be shown once when it has been created then is stored and will not be shown again to the user which will prevent them from being annoyed by repeated notifications telling them the same information over and over.

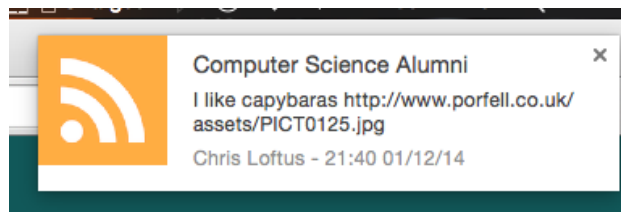


Figure 9: Here is a Chrome notification showing the latest broadcast.

Due to the nature of how the browser Opera is designed it users the same core frameworks as Chrome thus allowing it to use the same browser extensions without any modifications the screenshot below shows the same broadcast that we created earlier also being displayed within Opera's notification system.



Figure 10: Same again but for Opera this time just to show multi platform support.

Just to confirm that this is a real broadcast and has not broken any of the other functionality within the CS-Alumni application this is a screenshot of the broadcast on the Twitter website.



Figure 11: Proof that the broadcasts still get sent to Twitter.

CS-Alumni Rails Application

New Additions

New Database View

New RESTful interface

Modifications

Changes to Seeds

Changes to Broadcasts

Changes to Security

Changes to REST implementation

Changes to Users

Communication Between Applications

RESTful Web Interfaces

Data Flow Diagrams

Application Testing

JUnit Tests

Background.js – Unit Tests.



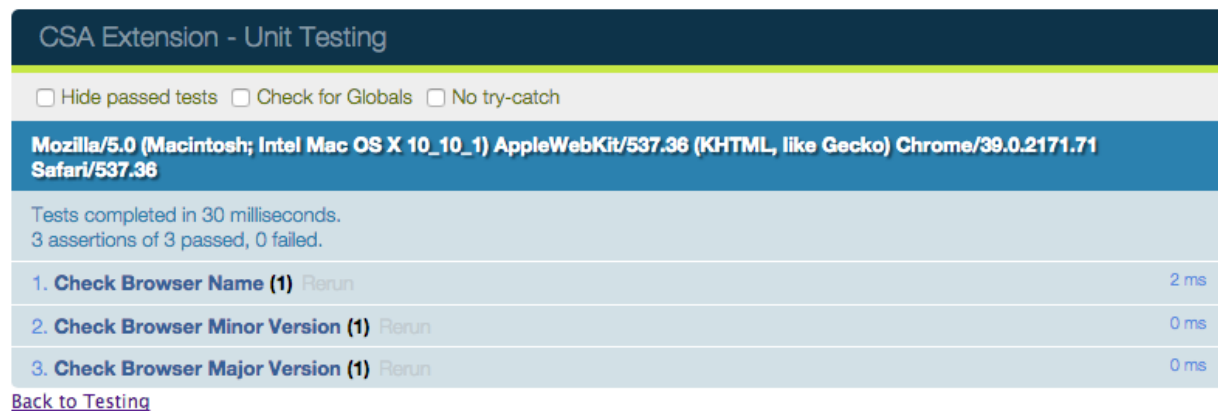
The screenshot displays the 'CSA Extension - Unit Testing' interface. At the top, there are three checkboxes: 'Hide passed tests', 'Check for Globals', and 'No try-catch'. Below this, a blue bar indicates the browser environment: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36'. The main area shows the test results for 'Background.js'. It states 'Tests completed in 41 milliseconds. 3 assertions of 3 passed, 0 failed.' Below this, there are three test items, each with a 'Rerun' button and a duration:

Test Item	Duration
1. Background pulls from REST correctly and stores it. (1)	19 ms
2. Background Initiates Correctly. (1)	2 ms
3. REST generates auth token correctly. (1)	1 ms

At the bottom left, there is a link labeled 'Back to Testing'.

Figure 12: This is the unit tests for the background.js module for the browser extension.

Browser Info – Unit Tests.



The screenshot displays the 'CSA Extension - Unit Testing' interface. At the top, there are three checkboxes: 'Hide passed tests', 'Check for Globals', and 'No try-catch'. Below this, a blue bar indicates the browser environment: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36'. The main area shows the test results for 'Browser Info.js'. It states 'Tests completed in 30 milliseconds. 3 assertions of 3 passed, 0 failed.' Below this, there are three test items, each with a 'Rerun' button and a duration:

Test Item	Duration
1. Check Browser Name (1)	2 ms
2. Check Browser Minor Version (1)	0 ms
3. Check Browser Major Version (1)	0 ms

At the bottom left, there is a link labeled 'Back to Testing'.

Figure 13: This images shows the browserinfo.js module completing its tests successfully.

Local Storage – Unit Tests.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 27 milliseconds.
3 assertions of 3 passed, 0 failed.

1. Load Nothing (1)	Rerun	3 ms
2. Local Storage Save. (1)	Rerun	2 ms
3. Local Storage Reload. (1)	Rerun	0 ms

[Back to Testing](#)

Figure 14: This shows the LocalStorage.js module passing all its unit tests.

Rest Tool Kit– Unit Tests.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 70 milliseconds.
7 assertions of 7 passed, 0 failed.

1. Auth Token is built correctly. (1)	Rerun	2 ms
2. Rest Get Object (1)	Rerun	20 ms
3. Authentication succeeded. (1)	Rerun	12 ms
4. Build Auth Token (1)	Rerun	1 ms
5. Authentication clear succeeded. (2)	Rerun	19 ms
6. Update user record via POST (1)	Rerun	2 ms

[Back to Testing](#)

Figure 15: Above we see the RestToolKit.js module successfully completing all its unit tests.

Utils.js – Unit Tests.

The screenshot displays the 'CSA Extension - Unit Testing' interface. At the top, there are three checkboxes: 'Hide passed tests', 'Check for Globals', and 'No try-catch'. Below this, a blue bar indicates the browser environment: 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36'. The main area shows the test results: 'Tests completed in 19 milliseconds. 2 assertions of 2 passed, 0 failed.' Below this, a list of tests is shown: '1. Diffed Array (1) Rerun' with a duration of '2 ms', and '2. Reverse Diffed Array (1) Rerun' with a duration of '1 ms'. At the bottom, there is a link 'Back to Testing'.

CSA Extension - Unit Testing

☐ Hide passed tests ☐ Check for Globals ☐ No try-catch

Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.71 Safari/537.36

Tests completed in 19 milliseconds.
2 assertions of 2 passed, 0 failed.

1. Diffed Array (1) Rerun	2 ms
2. Reverse Diffed Array (1) Rerun	1 ms

[Back to Testing](#)

Figure 16: Included above we see the Utils.js module efficiently complete all of its unit tests.

Technologies

CS-Alumni Chrome Browser Extension

HTML

HTML is a markup language that dictates the layout of the screens that are shown within the browser extension it is the skeleton in which the content is placed and displayed to the end user, it has no style to start with it just is the structure of the screen.

CSS

CSS works tangent with HTML and gives the skeleton of a document some style, it is responsible for colour schemes along with the positioning of various elements on the page and how they are displayed.

JavaScript

Is the programming language that the browser extension has been written in, Google's definition for JavaScript is 'an object-oriented computer programming language commonly used to create interactive effects within web browsers.'

JavaScript is an integral part of creating an interactive experience for web based content. It interacts with the various libraries we require to make the application work as intended. It works together with the other web based web technologies within the browser extension to give us total control of the content that is presented on the screen.

Chrome Extensions API

This is the framework that we must adhere to to make the 3 main web based technologies (HTML, CSS and JavaScript) work as a browser extension, the Chrome Extensions API bundles all of the technologies we need to create a viable and fully working browser extension, it takes in the web technologies plus a few extra configuration files to generate a browser extension that will do what the developer intended.

Bootstrap

Bootstrap I see as an essential tool to creating a modern looking website, it unifies the appearance of the application along with giving the ability to make the pages adaptive and work on any screen size. This means that if we were to run the browser extension as a web page it would work on any screen size or any resolution.

I choose to not use the standard Bootstrap theme opting to use a modified one that was supplied by <http://bootswatch.com/> (Theme name is Yeti) I choose to modify this further and make it try and more closely fit the Aberystwyth colour scheme.

jQuery

Along with Bootstrap I also see jQuery as an essential library to be able to create modern and feature full web applications (If I remember correctly all of the other JavaScript libraries I am using rely on jQuery).

Within the application I use jQuery to handle many bits of the functionality within the screens, the key points being aJax requests this enables very simple REST type responses to the server without much hassle, I also use it to edit DOM elements in real time and make them display the information I want them to display in real time.

The jQuery library is so vast that I use a lot of the trivial functions for doing all sorts of other work as the method calls and work that is required is greatly reduced.

Moment.js

This library simplifies interaction with date objects, its is well known that is quite difficult working with date objects within JavaScript, this library takes all the pain out of modifying the dates and putting them into a format that developer actually wants.

I use it for the broadcast times to make them display in the correct way, the objects have been converted into a JSON string and when they are converted back they stay as a string I use moment to convert theses strings into a viable date object that then can be displayed on the screen.

Alertify.js

Alertify is a JavaScript library that makes modern user alerts, this gives well designed and easy to use dialogs to present information to the user.

I have used it within the application to notify when a user has not been logged in and redirect them to the login page, along with showing a message confirming that there settings have been saved successfully.

Qunit

Due to doing a test driving development strategy to developing this browser extension I needed a unit test framework for JavaScript, it is developed by the same developers as jQuery and seems to have strong industry recognition.

I have used this for all my unit testing for the JavaScript libraries that I have written for this project, It has a simple and efficient way of testing the functions along with having a simple and powerful syntax that allows tests to be written without much code.

CS-Alumni Rails Application

Ruby

Ruby on rails

Qunit

Communication Between Applications

RESTful Interfaces

CORS

Basic Authentication

Evaluation

Something Extra



Figure 17: Above is a image of the Pebble smartwatch running the CSA application.



Figure 18: This is a screenshot taken from the Pebble smartwatch.

Attributions

References

- [1] Mark McDonnell. *SQLite and ActiveRecord*. Oct. 2013. URL: <http://www.integralist.co.uk/posts/sqlite-and-activerecord/>.
- [2] Daniel Morrison. *Building Awesome Rails APIs: Part 1*. June 2013. URL: <http://collectiveidea.com/blog/archives/2013/06/13/building-awesome-rails-apis-part-1/>.