CSE134B TEAM STACKS

COINFLIP – HW5

Team 19

Application Use

Live Demo

Please use the following URL to see our app. Using the URL will allow you to see our JS error tracking and analytics work. If you choose to open our files statically, be wary that you WILL GET CROSS ORIGIN POLICY ERRORS because of our tracking/analytics.

http://teamstacks.parseapp.com/

Sample Work Flow

1) Open index.html. Click Register to register an account: and

2) You should automatically be moved to wire2.html at this point. Otherwise, you may log in with and

3) Choose AU and click the plus button, since you don't have any coins added to your inventories, to add a new coin.

4) Choose an image from your computer for the picture.

5) Input VALID entries. Our form validation will prevent you from adding coins that have invalid inputs. Let's save the coin as a 'Gold'.

6) Save the coin.

7) Check AU again and you should see your coin.

8) Click the row with your new coin in order to view the details. You should see what you inputted as well as the image you chose.

9) Notice that there is a red Delete button. Click this button to delete your coin.

10) Go back to AU and you should not see your coin anymore.

11) Click the Gear/Cog icon that you see on every page. This is your profile.

12) Check the box that says "Use Light Theme". Update your profile.

13) Click on your profile again (Gear/Cog icon) and click "Log Out" to log out.

Pages

```
index.html - Login/signup page. Should open modal and allow you to log in
or sign up.
wire2.html - Home page with Total Coin Value and graphs for all coins.
wire3.html - My [Gold, Silver, Platinum] page with details on owned gold
coins.
wire4.html - Item page that looks at a specific gold coin wire5.html - New
Item page for adding a new coin.
```

Navigation

1) If you are not logged in, you will not be able to see the other pages, as they will redirect you to index.html.

2) Once you are logged in, unless you log out, you will automatically be redirected to wire2.html if you look at index.html.

Inside wire2.html, and any other page for that matter, if you click the cog symbol in the top right, you will open your profile, from which you can log out, make changes to your profile, or link your Facebook.

Clicking on the AG (silver) or PT (platinum) boxes on the side nav will navigate you to your inventory pages, which should dynamically serve you your current inventory (empty if you have not added any items).

Design

Responsiveness

All interactions with the application should be supported in Mobile view.

Light/Dark Theme

By default, our app uses the Dark theme. You can change the theme when you edit your profile settings by clicking the Gear/Cog icon on every page.

Validation

HTML

All the HTML validation checked out so there was no problem there.

JavaScript

There should be no console errors, but there are occasionally warnings and also, debug prints when viewing with the developer console.

Implementation

We used a mixture of REST API and JavaScript and jQuery for our backend. Please continue reading to for more detail.

Front-end (From Dream Team spec)

- HTML
    - We used HTML5 so this technology was pretty straightforward. Perhaps the one noteworthy thing is our use of SVG, particularly for icons from IcoMoon.io (source: https://icomoon.io/) to save us the time of actually designing our own icons
- SASS/CSS
    - We wrote all of our CSS using SASS that compiled into CSS. We used SASS in a way that the syntax was identical to that of simply using CSS. The only additions we utilized SASS for were as follows:

      1) SASS Variables - easier refactoring of color themes

2) SASS Nesting - easier for the developer to nest attributes, mainly used as a security fall-back against conflicting styling as we were merging files and work

3) SASS Mixins - For any CSS attributes that needed all the extra prefix declarations for moz, webkit, etc., we used simple @include mixins from SASS to avoid having to type all the prefixes ourselves.

- Otherwise, all of our SASS was written exactly the same as CSS. We simply employed the SASS to save some time and organize common themes easier.

- The SASS file is located in sass/style.scss.
- The CSS file is located in style/style.css.

- JavaScript

```
1) Importing top navigation, side navigation, and footer on all
pages. This allowed us to be able to just change the common elements
in one place and have it affect all 5 pages rather than have to go
through every single page for changes. This was done through simple
document write statements (with extra input in to the function for
the side navigation, this let us easily specify which side
navigation button should be 'pressed'). These import functions are
all in main.js and are:
- loadTopNav()
- loadTopNavPersist()
- loadSideNav(selected)
- loadFooter()
2) Creating the graphs. This was done using a library called
Chart.js. C3JS Source: http://www.c3js.org/
3) Handling navigation toggle buttons. This is when the application
is shrunk down to mobile size where the toggle buttons of
information and chart appear. Instead of having those buttons
navigate to different pages, we thought it made more sense that they
would toggle what information is already on the page. This was done
with
   jQuery on click events.
   Source: https://jquery.com/
```

Backend

- Parse

    - Our backend uses Parse, so if you would like to see a live demo of our app (working with Facebook log in as well, please use http://teamstacks.parseapp.com).

      1) OAuth and Login. We used Parse as our account management system.

      2) Database. Parse also provided us a database, to which we stored all data such as inventory and accounts and coin information. The following is our database schema.

      ```
      - Coin: Holds specific coin information, including image. - User: Holds user information - Inventory: Hold inventory information connecting the Coin and User databases together.
      ```

      3) Storing Images. We also used Parse to store images that were uploaded of each of the coins.

      4) ~~Since making a network call to Quandl anytime a page is loaded is too slow (~2 second load time), we instead wrote background jobs that pull updated information from Quandl everyday. This sped up the script loading by a great deal.~~ Initially the network calls to Quandl were really slow, but they're ok now.

- Facebook SDK
    - We used Facebook as well to also provide users another way of logging into our application.
- JavaScript
    - We used jQuery and JavaScript to accomplish all dynamically delivered content.
    - For more information about JavaScript and JavaScript libraries we used, please refer to the next section.

JavaScript in Detail

JavaScript Libraries Used

- parse-1.4.2.js - JavaScript library from Parse
- bootstrap.min.js (version 3.3.4) - We used this for the pop-up modals
- jquery-1.11.2.js - We used this for jQuery
- c3js & d3js - for graphing needs
- d3.js - for dynamically serving data
- tracker.js - for keeping track of our errors

JavaScript Files We Made

- query.js - Holds some methods for loading information from Parse as well as other things related to querying, such as parsing URL for ID when viewing an item (wire4.html)
- profile.js - Handles all the methods for logging in and verifying log in with Parse SDK and Facebook
- init.js - The javascript that initializes our Parse and Facebook SDK libraries.
- quandl.js - Handles scraping of data from Quandl for our bid and total information.
- Bid/Ask scraper - This code was provided by the TA Alex. It is hosted on a separate Heroku instance at teamstacks.herokuapps.com.

Error Tracking

please go to https://trackjs.com/. Enter the following email and password to see our error tracking:

email: jduan@ucsd.edu

password: 12345671

Here are some of our error tracking results from trackjs:

Dashboard (Realtime Analytics)

Search

Errors per Hit

0.71

Last 24 hours

Error Rate Check Usage

7.3 /hour

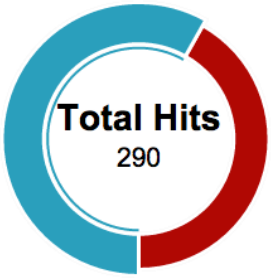
Last 24 hours

Error Delta

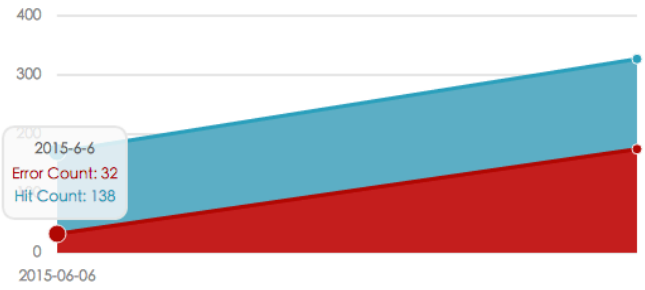
446.9% ↑

Last 24 hours vs Previous

Errors and Hits



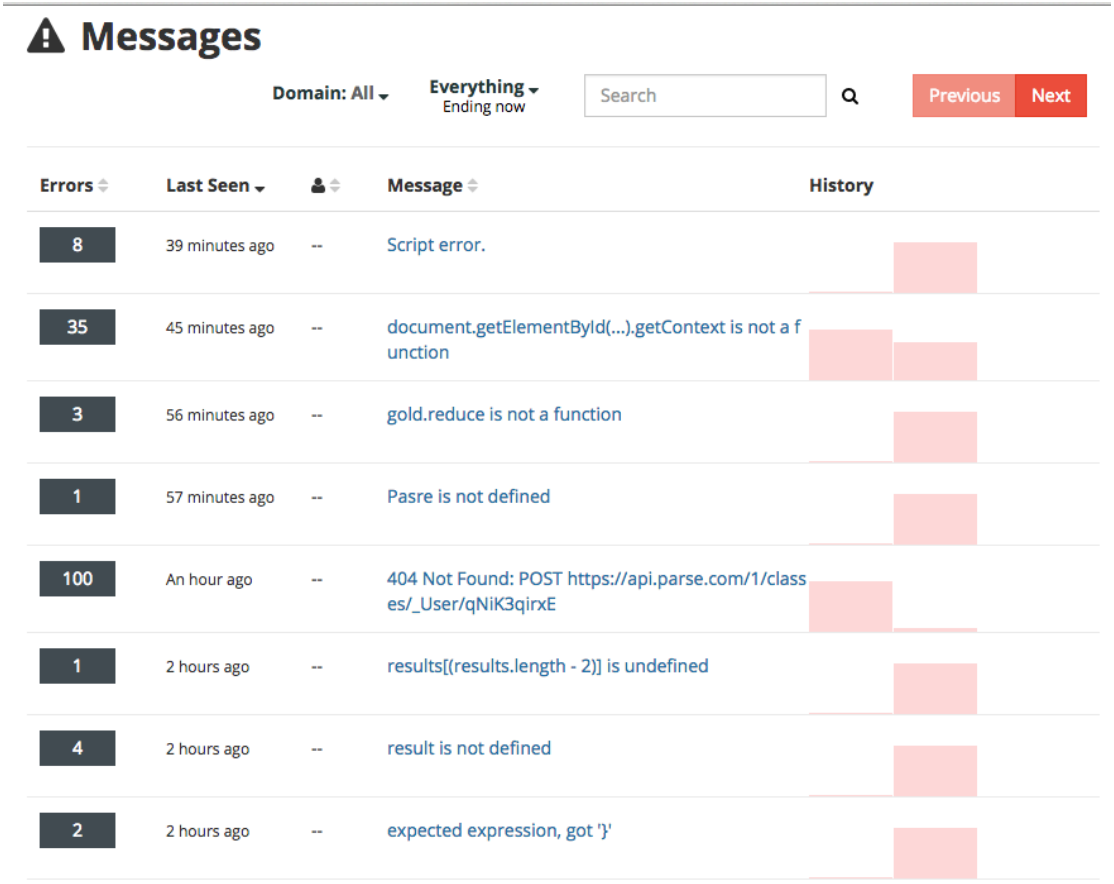
Errors and Hits Timeline



Recent (Showing Recent Errors)

Domain: All ▾ Everything ▾ Ending now Search Previous Next

Timestamp	Message	Url	Browser	OS
2015/6/7 2:31	Script error.	file:///C:/Users/whleung/Development/TeamStacks/hw4/public/wire3.html?q=Gold	Firefox	Windows 8.1
2015/6/7 2:31	Script error.	file:///C:/Users/whleung/Development/TeamStacks/hw4/public/wire3.html?q=Gold	Firefox	Windows 8.1
2015/6/7 2:31	Script error.	file:///C:/Users/whleung/Development/TeamStacks/hw4/public/wire3.html?q=Gold	Firefox	Windows 8.1
2015/6/7 2:30	Script error.	file:///C:/Users/whleung/Development/TeamStacks/hw4/public/wire3.html?q=Gold	Firefox	Windows 8.1
2015/6/7 2:25	document.getElementById(...).getContext is not a function	http://teamstacks.parseapp.com/wire3.html?q=Gold	Chrome	Mac OS X
2015/6/7 2:22	document.getElementById(...).getContext is not a function	http://teamstacks.parseapp.com/wire3.html?q=Gold	Chrome	Mac OS X
2015/6/7 2:21	document.getElementById(...).getContext is not a function	http://teamstacks.parseapp.com/wire3.html?q=Gold	Chrome	Mac OS X
2015/6/7 2:20	document.getElementById(...).getContext is not a function	http://teamstacks.parseapp.com/wire3.html?q=Gold	Chrome	Mac OS X
2015/6/7 2:15	document.getElementById(...).getContext is not a function	http://teamstacks.parseapp.com/wire3.html?q=Gold	Chrome	Mac OS X
2015/6/7	document.getElementById(...).getContext	http://teamstacks.parseapp.com/wire3.html?	Chrome	Mac OS



Concerns

~~The Gold/Silver/Plat totals were very difficult to graph because of ChartJS's limitations. ChartJS only accepts one set of x-axis values, which means that all datasets must share the same set of dates for the charted information to make any sense. We ended up writing a basic function that sums up all purchases made before a certain date. In short, the gold/silver/plat totals on the chart will look inaccurate, especially if an item's purchase date is not within the range of dates on the chart.~~ Resolved by scrapping ChartJS and using c3js.

~~There are also gaps in the dates for the 1oz gold/silver/platinum data from Quandl (no idea why). Ideally we would have written a scraper to scrape daily info from some other site, but we weren't sure how to do this in Parse (would be much easier in Node).~~ Resolved by using the scraper provided by Alex.

There is a discrepancy between the Parse server dates and the browser dates that causes a minor difference in the dates when fetched and when stored. In Chrome, this is off by 5 hours, whereas in Firefox, the time is parsed correctly.

We only had time to support JPG images, however, you can upload any kind of file with the Upload form.

Since we are using the formula Total = Qty * Weight * Gold % + Premium (https://groups.google.com/forum/#!topic/ucsd-cse-134b-spring-2015/Ven54JMGjtc) to calculate the total value of the metal, overall change will always be zero as it does not involve market price.

The light theme is not very legible.

Thank you!

Best, CSE134B Team Stacks