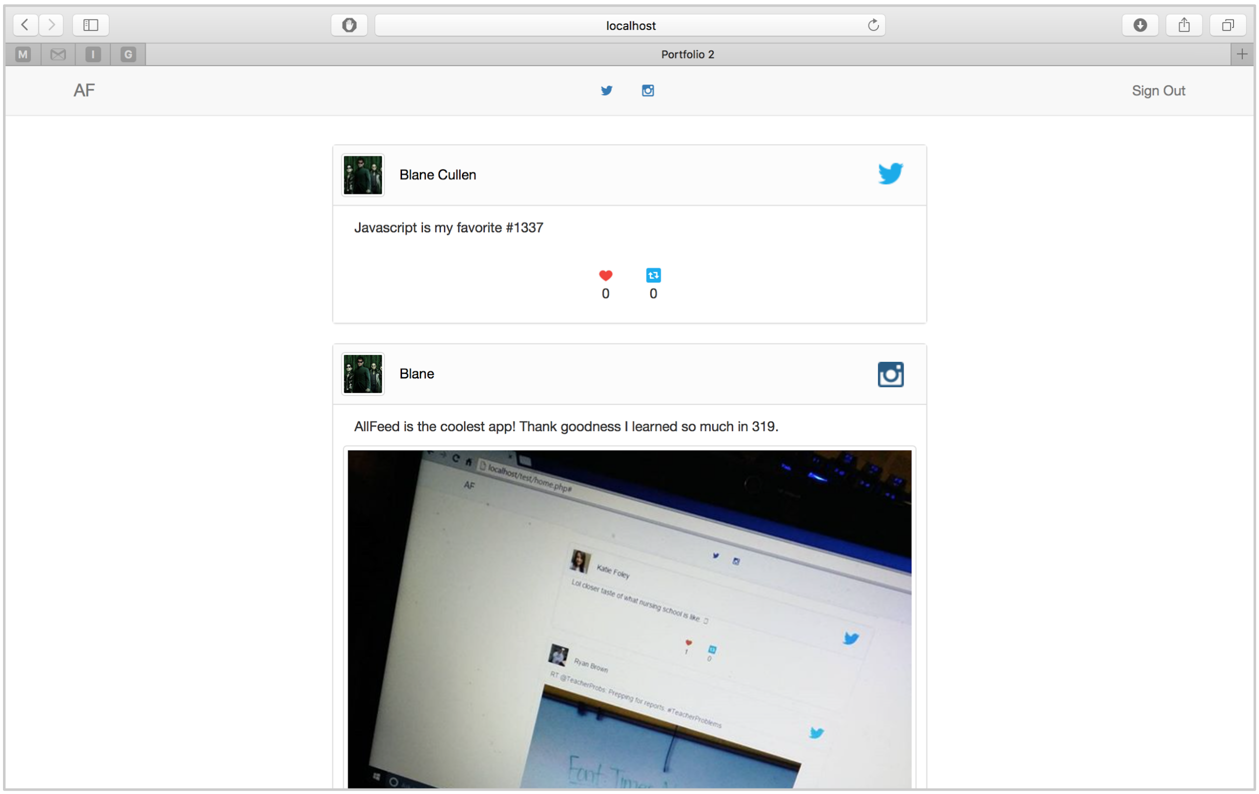
Portfolio II

Computer Science 319
Fall 2015

Jered Hoff and Claude Cullen

Introduction

The purpose of this portfolio was to demonstrate the web development skills we gained in the last few weeks of class and to bolster these skills to develop a useful website. After lots of brainstorming and bouncing ideas off of each other, we decided to build a website that allowed the user to access all of their social media news feeds in a single feed. We came to this decision because we knew that this website would be a fun and interesting way to demonstrate our newfound skills. With web development being rather new to the both of us, there was lots of learning and research to do. Because our idea was rather complex, multiple issues presented themselves throughout the development process and we had to come up with numerous creative solutions to overcome these issues and achieve our goals.



Blane Cullen

Javascript is my favorite #1337

0 0

Blane

AllFeed is the coolest app! Thank goodness I learned so much in 319.

Interaction with Material

Questions

- How do we get the data from the social media websites?
- How will the social media websites confirm with the user that we can access their data?
- What form will the data come in from the social media websites?
- How do we sort all of the posts in the feed by the time that they were posted to their native website?
- How do we organize the data coming from the other sites in a way that is appealing and recognizable to the user?

Insights

We used a variety of topics that were covered in class, and applied them to things that were useful and interesting. The majority of our project was done in Javascript and we used a variety of different Javascript skills we learned in lab, for example, classes, objects, and callback functions. We also had to use JSON and javascripts methods to convert JSON to javascript objects. Seeing JSON used this way made it very clear how useful and convenient passing data in JSON format is.

Aside from using JSON objects, we also dealt with anonymous objects in our project. Different kinds of data could be passed to us based on what social media the data was coming from; for example, if the data was coming from twitter it would have favorites and retweets, but when coming from instagram it would have comments, and likes. At first we thought about creating two completely different classes to accommodate for this small difference in data that we wanted to obtain, but then we realized we could use an object literal and place give our post class an interactions object, which would have different properties depending on if it was a Twitter or Instagram post.

We also took a variety of other topics that we covered in lecture and lab and used them to create a real, working application, for example we used php to communicate with a database and manage users, created our feed by appending elements to the document in Javascript like in the library lab, used jQuery to select elements, and many others.

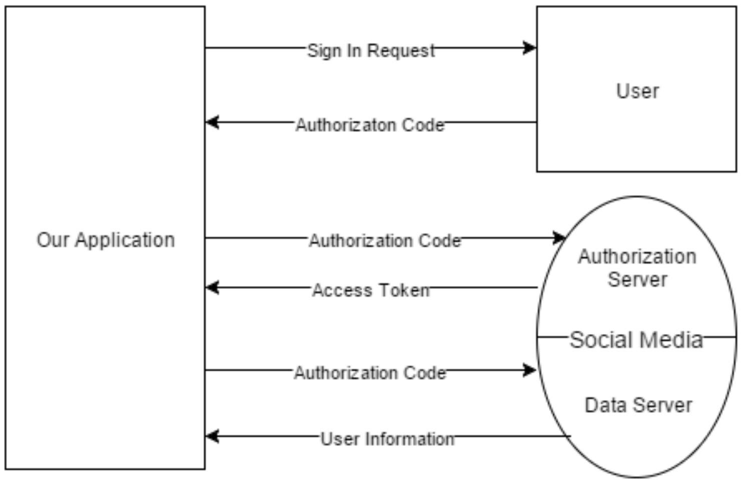
Complex Issues

OAuth

In our application we used standard OAuth protocol to allow users to sign into their different social media accounts. To do this we used a very convenient javascript library called OAuth.io. This library streamlined the OAuth process for multiple different social media sites, but it was still important that we understood the basics of how OAuth was functioning within our application.

In order to get the user's information from a social media source (i.e. Twitter and Instagram) using an API, we first had to verify with the user that our application is allowed to use the content on their page. This is where the OAuth authentication protocol comes in. OAuth is used by most social medias to verify with their users that it is acceptable for third party applications such as ours to use data that is protected on the user's social media profile within the third party application without actually sharing the user's password with the third party application.

The idea of OAuth can be easily described but implementation can be rather complex. The diagram below shows the general idea of OAuth.

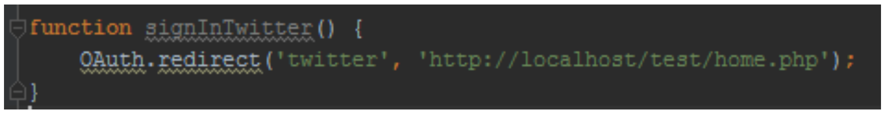


The first step of the OAuth process is to request authorization for our application to use the user's data from the user. This is done by redirecting the user to a site where they can sign in using their username and password without directly giving their username and password to our application. Once they have signed in with their username and password they will then be redirected back to our application with an authorization code called an “Authorization Grant”.

The next step of the OAuth process is to use the authorization code received in the previous step to get an access token from the social medias authorization server. This is done by making a specific request to the server which varies depending on the social media. The authorization server will verify that the authorization code that we sent it is valid and connected to a certain user. Once it is verified the authorization server will send us an access token.

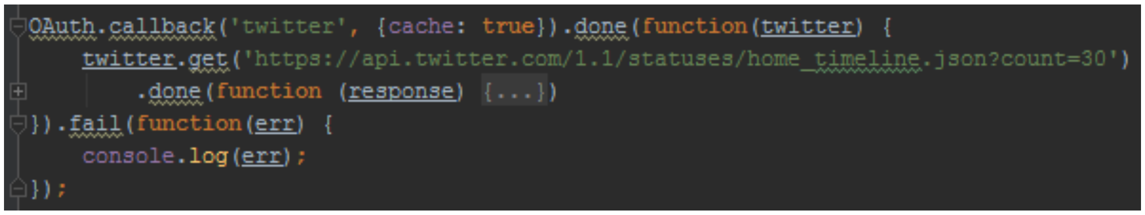
Once we have an access token we can make a request for a piece of information, in our application, we wanted to request the user's feed. We send a request to the social media's server, along with the access token; and if the token is valid, we will get a response with the data we requested.

When we attempted to implement this process ourselves we got very stuck and were unable to continue. The two social media's that we implemented had unclear documentation on how to use their OAuth process. That is when we found the very convenient library OAuth.io, it allowed us complete the OAuth process very simply with a variety of different social medias. We could do this with the following code:



```
function signInTwitter() {
    OAuth.redirect('twitter', 'http://localhost/test/home.php');
}
```

The above code covers step one of the OAuth process. It sends the user to a twitter sign in page and then returns to our page with an authorization code.



```
OAuth.callback('twitter', {cache: true}).done(function(twitter) {
    twitter.get('https://api.twitter.com/1.1/statuses/home_timeline.json?count=30')
        .done(function (response) {...})
}).fail(function(err) {
    console.log(err);
});
```

Once this code has an authorization code it will make a request to twitter. It will go through the authorization code to access token step for us, then make a request for data to the API endpoint shown in the code.

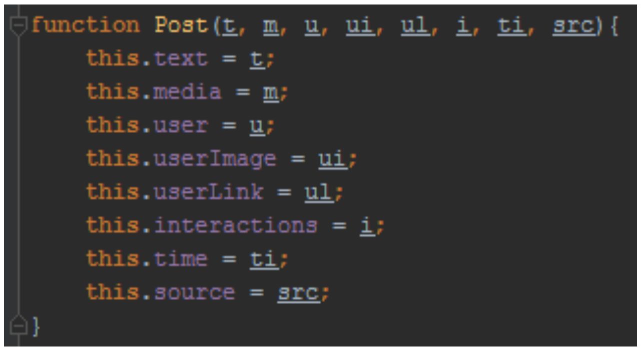
Rest API Calls

Another technology our application used was Rest API Calls. We used these to get the user's feed from different social media sources. To make an API call we simply sent a GET request (along with the access token discussed in the previous section) to an endpoint that was specified in a given social media developer documentation. Then the data was returned to us as a large JSON object. The JSON was then converted to javascript objects which we had to look through in order to acquire the data we need to create our feed.

Each request we sent returned a massive amount of data, each post object returned by the request contained a very large amount of information that we did not need. We were required to search through the returned object and locate the information we needed which consisted of; the text of the post, the image or video of the post, information about the user who made the post (like name and image), the interactions with the posts, and the interactions with the post.

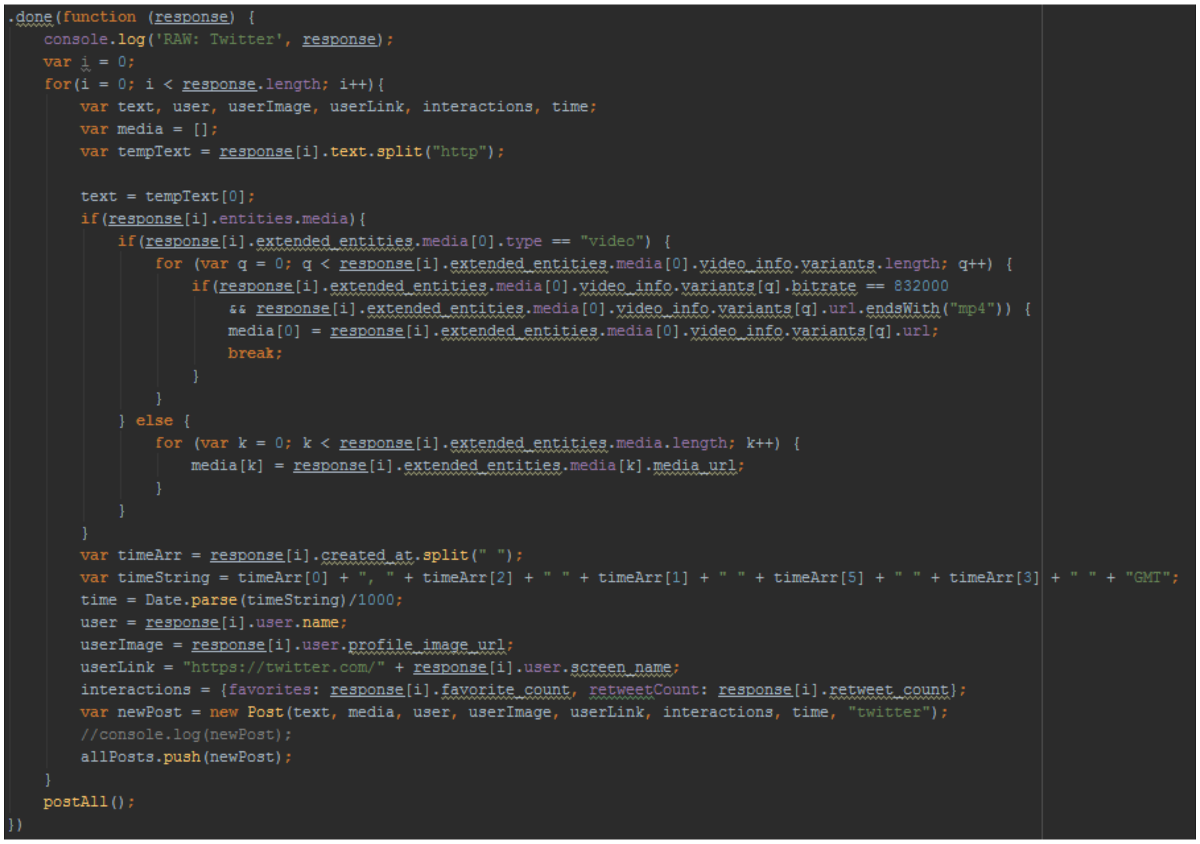
One of the biggest issues we ran into while doing this was that the data returned from different social medias was in a very different format, therefore required two very different methods to get the required data out of the returned data. The good thing was that we required the same data from each of the different social medias and this allowed us to only create one Post class which could contain posts from all different social medias.

Here is a what our general Post class looked like. It contained all the data we needed to create our news feed out of posts from different social media sources.



```
function Post(t, m, u, ui, ul, i, ti, src){
    this.text = t;
    this.media = m;
    this.user = u;
    this.userImage = ui;
    this.userLink = ul;
    this.interactions = i;
    this.time = ti;
    this.source = src;
}
```

The code below shows the process of getting a response object from twitter and using it to fill up our Post object. There are a variety of situations and locations we had to consider when getting the data from the response object.



```
.done(function (response) {
    console.log('RAW: Twitter', response);
    var i = 0;
    for(i = 0; i < response.length; i++){
        var text, user, userImage, userLink, interactions, time;
        var media = [];
        var tempText = response[i].text.split("http");

        text = tempText[0];
        if(response[i].entities.media){
            if(response[i].extended_entities.media[0].type == "video") {
                for (var q = 0; q < response[i].extended_entities.media[0].video_info.variants.length; q++) {
                    if(response[i].extended_entities.media[0].video_info.variants[q].bitrate == 832000
                        && response[i].extended_entities.media[0].video_info.variants[q].url.endsWith("mp4")) {
                        media[0] = response[i].extended_entities.media[0].video_info.variants[q].url;
                        break;
                    }
                }
            } else {
                for (var k = 0; k < response[i].extended_entities.media.length; k++) {
                    media[k] = response[i].extended_entities.media[k].media_url;
                }
            }
        }
        var timeArr = response[i].created_at.split(" ");
        var timeString = timeArr[0] + ", " + timeArr[2] + " " + timeArr[1] + " " + timeArr[5] + " " + timeArr[3] + " " + "GMT";
        time = Date.parse(timeString)/1000;
        user = response[i].user.name;
        userImage = response[i].user.profile_image_url;
        userLink = "https://twitter.com/" + response[i].user.screen_name;
        interactions = {favorites: response[i].favorite_count, retweetCount: response[i].retweet_count};
        var newPost = new Post(text, media, user, userImage, userLink, interactions, time, "twitter");
        //console.log(newPost);
        allPosts.push(newPost);
    }
    postAll();
})
```

Bloom’s Taxonomy

Analysis

People enjoy their social medias; that’s why they’re always engulfed in them. People also enjoy convenience and simplicity, so in this project we figured out a way to combine the two. Since most people use multiple social media platforms, a combination service would greatly simplify the social media experience. By combining the user’s social media news feeds into a single feed, we cut out the user’s inconvenience of having to switch between websites or applications to get completely caught up in their feeds. Our service allows the user to access their entire news feed in a shorter amount of time, making the social media experience much more simple and enjoyable.

Evaluation

As we created our web application we came across a variety of design choices that we had to evaluate many options for doing various different tasks. The first of which was how we should implement the OAuth process. When we tried doing that ourselves we had a very hard time and the documentation for how to implement OAuth was quite unclear. Then we decided to use a library, and we had to track down the one that met our needs. This lead us to OAuth.io which worked very well for our project.

Another thing we had to evaluate was what sort of data we wanted to store in our post object and how we would store it. This had to change a bit as we progressed through development; for example, we didn’t have posted time in our original class, but realized that was a required piece of data for creating an ordered news feed. Another decision we had to make about the data was whether one class could contain all of the different kinds of posts from different social medias. This was unclear in the beginning of the development process, but as we began getting data, we realized that although the posts from different sources had different names and types of data, they could easily and accurately be placed into one single type of object as long as we correctly got the data out of the object that was returned from our request.

Another decision that we had to evaluate was what social medias should we use. We initially thought that we should use Instagram, Twitter, and Facebook. After we looked at the API’s for Facebook we found that there was not a request for getting a users news feed, which was what we were doing for all of the other applications. Instead of the Facebook news feed, we would have been able to get the posts on the user's wall, but we decided that this information would seem out of place since we were getting the news feeds from the other two social medias.

Creation

- Our website is a completely new website that was built from scratch and implemented multiple open-source libraries to create an optimal user experience.

- Created multiple data objects in javascript to handle the data coming in from the social media websites.
- Designed the user interface to display the feed in a way that is visually appealing and easily recognizable to the user.