Assignment Two: JavaScript (for Designers)

Due: February 20, 2013 @ 11:59pm

In this assignment, you'll catch a fever: Bieber fever. You will be implementing the BieberFeed, a tool for displaying a streaming list of Tweets posted about Justin Bieber.

Important Note: this project uses Ajax that violates the same-origin policy, and as such, it requires a web browser that supports XMLHttpRequest 2 and CORS. This means you **must** be using a semi-recent web browser: Firefox 4+, Chrome 7+, Safari 5+, or IE 10+.

The Problem

Every second, there are nearly 50 new Tweets published about Justin Bieber. It can be quite harrowing to keep track of all of them on Twitter itself, so you are being tasked with finding a way to separate the proverbial wheat from the chaff.

Requirements

We have provided a web server that will provide you with the 25 most recent Tweets about Justin Bieber every time you ask for them. You will build a website that takes those Tweets and presents them to the user in some kind of a streaming list.

You will be responsible for reacting to incoming Tweets, and using the DOM to append them to a new part of the page.

Getting Started

Create a new directory to hold your project's files, and then run:

```
cs132_install javascript
```

You can also download the assignment from GitHub: https://github.com/brown-cs132-s13/javascript/archive/master.zip.

This will give you a blank `README.md` file (to contain known bugs and any features you want to highlight), a very barebones shell of an HTML file, and a picture of Justin Bieber's face that you may use and modify as you see fit in the design of your project. There is also a "no photo" placeholder image you might want to use for Tweets that don't have pictures associated with them.

You should implement all of your HTML in `bieber.html`, and all of your JavaScript in `bieber.js`.

The BieberFeed Server

You will be interacting with a server we've provided that will give you Bieber-related Tweets. To do this, you'll use a TA-provided library which abstracts away some of the complexity of making Ajax requests and publishing DOM events.

To get started, you have to link the TA library into your HTML page. Open up `bieber.html` in your favorite text editor, and add this to the head:

```
<script type="text/javascript" src="designer-helper.js"></script>
```

... and right below that:

```
<script type="text/javascript">Feeder.init('bieber.mattpatenaude.com')</script>
```

... to initialize the events. The domain in `Feeder.init` may change (we'll send updates).

Since the idea of the BieberFeed is that it's a live stream of Bieber-oriented Tweets, having the page refresh every second with new content would be a bit annoying. For that reason, the TA library will fetch new Tweets automatically using Ajax. You'll then need to setup a handler for responding to these incoming Tweets.

Responding to DOM Events

DOM Events are a useful way for your JavaScript to be informed of things that happen in the context of your website. A number of default events are provided, like `load`, `click`, and `mouseover`, and it's also possible to define your own.

For example, let's say you want to execute a piece of JavaScript once your page finishes loading. The `window`

object publishes the `load` event, which you can listen for by assigning a function to the `window.onload` property:

```
window.onload = function(event){
    // your code here
};
```

... or more appropriately, using the `obj.addEventListener(...)` method:

```
window.addEventListener('load', function(event){
    // your code here
}, false);
```

Note that the last parameter, `false`, defines whether the event happens in bubble or capture mode - under almost every circumstance, you want this set to `false`. Don't worry about it. Also note that stencil code for this is already provided in `bieber.js`.

You can also pass the name of the function (**without** parentheses at the end) in either case:

```
window.addEventListener('load', myLoadHandler, false);

// more code...

function myLoadHandler(event) {
    // your code here
}
```

The TA library publishes an event called `swagreceived` on the `document` object whenever new Tweets are ready for use. You can listen for events using something like this:

```
document.addEventListener('swagreceived', function(event){
    var username = event.handle;    // the username of the Tweeter
    var text = event.text;          // the text of the Tweet

    // your code here
}, false);
```

The event object passed into the event has a handful of properties you may find useful. `event.handle` is the username of the person who posted the Tweet; `event.text` is the body of the Tweet; and `event.tweet` is an object full of extra information on the Tweet. The available properties of `event.tweet` are those returned by the Twitter API for Tweets. Note that Tweet entities are supported, so you should consider using them to create a

more engaging user experience.

Appending the Tweets

You'll want to come up with some way to insert new Tweets into the page. Usually, you'll do this (for example) with `document.createElement(...)`:

```
// we have an unordered list of Tweets
var ul = document.getElementById('tweets');

// create a new li element for the Tweet, and append it
var li = document.createElement('li');
li.innerHTML = '<strong>' + event.handle + '</strong> ' + event.text;
ul.appendChild(li);
```

You might want to use `element.insertBefore(...)` instead of `element.appendChild(...)` depending on how you want to indicate Tweet chronology (newest on top vs. newest on bottom). The Mozilla Developer Network documentation for HTML elements includes a large number of DOM methods you could use to make your life easier.

Other Niceties

Think about how your choice of mechanisms will affect the user experience of the BieberFeed. Do new Tweets appear at the top, or the bottom? Does the page keep growing in length, or do you start removing old Tweets after a certain point? Can the user start and stop the feed? Is the list of Tweets independently scrollable (hint: lookup `overflow-y: auto`), or does the entire page scroll with it?

Add any features you can to try to make the experience of looking at so many Bieber Tweets bearable.

Handing In

Before handing in your project, make sure you've checked to make sure you've filled in your `README.md` file (you can use any format you like for your README, but Markdown is highly recommended).

To hand in your project, from your project directory, run:

```
cs132_handin javascript-designer
```

That's it!