Class: IS219 – Advanced Website Development

Assignment: Project 2

Due: 4/1/2016 (Friday); 11:55pm

## Bonus

If you an extra 6 possible points, you can take part in a short research experiment. Not doing this experiment will not affect your grade in any way. Here's the description:

Research has shown that keeping track of activities can help with time management and understanding patterns in one's behavior. This app will help you keep track of how much time you spend studying for the class project. Whenever you are working on the project, press "start" and "stop" when you are done. Over time, you will be able to see your study patterns, which may or may not help you.

We are not interested in how much time you spend studying. We are interested in whether or not having a tool that helps with study tracking helps you with time management.

In order to get credit, you will have to1) fill out a survey before starting the assignment (pre-survey; link velow), 2) use a simple web app to track your study habits, and 3) take a survey after you submit the project (post-survey; will be sent to you after you submit your project).

Link to the pre-survey:
http://www.surveygizmo.com/s3/2631004/Tracking-study-habits

Link to the Study Tracker App:
(please bookmark it so you can easily go to the site)
http://pixel42.com/projects/studytracker/

Class: IS219 – Advanced Website Development

Assignment: Project 2

Due: 4/1/2016 (Friday); 11:55pm

## Part 0: Get the code

1. Download the Project 2 ZIP file from Moodle (Week 7).
2. Move the contents of the ZIP file to your computer's local web directory.
    a. If you don't have a web directory, you can install an "AMP" stack for Apache (web server, MySQL and PHP. I recommend AMPPS (www.ampps.com).
    b. Once you get your AMP stack running, you can visit your site using http://127.0.0.1/
3. Using your Github Desktop application, create a new project using your Project 2 web directory from Step 2. I recommend using the following project name: is219s16lastname-p2
4. Connect your Github repository to a new Heroku app. I recommend using the following name: is219s16lastname-p2.herokuapp.com
5. Follow the directions below. Please commit to your Github repository often to keep track of your code and to show your progress.
    a. When committing changes to Github, use the "best practices for commit messages" guide that was outlined in Week 3 (available on moodle's Week 3 lecture slides).

**note*: you do not have to edit *index.html* or *index.php* at all to complete this project (but you may if you want). The primary files you will be editing for this project are *gallery.js* and *style.css*

## Part 1: Create an Javascript Object

1. Create a Javascript Object, called **GalleryImage,** by using a function[1] which contains the following properties:
    a. String : location (the location the photo was taken)
    b. String : description (a description of the photo)
    c. String : date (the date the photo was taken)
    d. String : img (the src URL for the photo) –OR–
    HTMLImageObject : img (an Image object which contains an image bitmap)

## Part 2: Slide Show

1. Use an XMLHttpRequest[2] called **mRequest** to fetch the JSON located inside the *images.json* file. This will be your default JSON file.

2. Create a JSON object that contains the retrieved JSON[3] string (in this case, a list of photo URLs and related metadata).

[1] http://www.phpied.com/3-ways-to-define-a-javascript-class/
[2] https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest
[3] http://www.w3schools.com/json

3. Iterate through the JSON object and create **GalleryImage** objects using javascript for each image in the JSON object. Put each of these new GalleryImage objects into an array called mImages[].

    a) You can either make this a STRING or an HTMLImageObject (see Part1.1.d, above)

        1. If you choose to use a STRING:

            1. Get the src URL from the JSON file and put it into your GalleryImage object.

        2. If you choose to use a HTMLImageObject:

            - Use this article for how to create Image Objects: https://web.archive.org/web/20130719194357/http://javascript.mfields.org/2011/creating-an-image-in-javascript/

            - When you set the img.src property, javascript will go and fetch the resource that was set. So you don't have to load the img yourself.

            - Set an onload event on each img you create. When that event fires, add the image into the array. Also, be savvy and remove the onload event so no further events get fired. To prevent closures from happening, use the `makeGalleryImageOnloadCallback` method for your load event handler. Some helper functions have been included for you in gallery.js to get you started.

4. Create a slide show to iterate over all the **GalleryImage** Objects in your mImages[] array. The *gallery.js* file includes code that creates and continuously runs a timer. When the timer goes off after 5000ms (5 seconds), it calls swapPhoto().

    a) Implement swapPhoto() so when it gets called, it finds the <img> inside the **div#slideShow** and swaps its current src with a new src on a **GalleryImage** object in your array. Use the **GalleryImage's** metadata information (Description, Location, and Date) and update the **div.details** section with the extracted info.

    b) Use a counter called **mCurrentIndex** to loop through all **GalleryImage** objects in the array. When you reach the end, start over again.

    c) The timer code is intelligent enough to not run when your webpage isn't currently focused. Keep that in mind when using the console. You could have the console open and not see events happening due to the webpage being unfocused (i.e. the current browser tab is not visible).

## Part 3: Gallery

1. Change the *img* element with a class value = "*moreIndicator rot90*" to be horizontally centered and bottom aligned with its sibling *img* element.

(http://stackoverflow.com/questions/14233341/how-can-i-rotate-an-html-div-90-degrees)

2. Use CSS to implement the *.rot90* and *.rot270* selectors that will transform selected elements to rotate 90 or 270degs. Use the CSS transition property to create a nice animation. (http://robertnyman.com/css3/css-transitions/css-transitions-rotation.html)

3. Add a click handler to the **img.moreIndicator** that does the following:

   a) Add a class attribute value = "*rot270*" if the element currently has a class value with "rot90"; else remove the "*rot270*" class and add "*rot90*". This will cause the arrow to animate upside down.
   (https://api.jquery.com/hasclass/)

   b) Slides down/up the **div.details** depending on the arrow direction.
   (http://api.jquery.com/fadetoggle/)

4. Use jQuery to offset the **#nextPhoto** image so that it is flush with the right side of the **#gallery** div (see *info/finished_gallery.png* for visual).

5. Add hover handlers (in CSS) to the **#nextPhoto** and **#prevPhoto** in the **div#nav** so that when the mouse pointer goes over them, their opacity is set to .8 (80%).

6. Add click handlers (in jQuery) to the **#nextPhoto** and **#prevPhoto** in the **div#nav** so that when they are clicked, they will go to the next photo or the previous photo, respectively. They should be shown in the array order. On the last photo, it should loop back to the first photo when **#nextPhoto** is pressed. Likewise, on the first photo, it should loop back to the last photo when **#prevphoto** is pressed.

## Part 4: Alternate JSON input using GET

1. Using Javascript/jQuery, add a GET handler that accepts a variable called "json" that allows a user to point to an alternate JSON file. If the alternate JSON file is valid, use that file as the gallery's input. If there is no json file provided, or the provided JSON file is invalid, use the default *images.json* file. For example, the following should work when you put it into your browser: index.html?json=images-short.json

## Part 5: Create your own JSON

1. Create your own JSON file based on *images.json* (please name it *extra.json*) that has URLs and metadata for your own photo/image collection.
2. There should be a minimum of 5 images used in the JSON.
3. Make sure that *index.html?json=extra.json* (see requirements from Part 4) show your images and metadata in your gallery.

## Part 6: Submit

1. Take one screenshot of your working website with the details section opened.

2. Submit the following on Moodle by April 1st, 2016; 11:55pm:

    1. Your screenshot

    2. A link to your project's Github repository
    (example: http://www.github.com/mjslee/is219s16lee-p2)

    3. A link to your project's Heroku server/website
    (example: http://www.is219s16lee-p2.herokuapp.com)

## Grading Rubric

| Deliverable | Points Possible | Your Points |
|---|---|---|
| Did you properly use Github to track your changes (with appropriate commit messages)? | 5 | |
| Did you properly deploy your project to a Heroku server? | 5 | |
| Is XMLHttpRequest correctly implemented to read a default JSON file (i.e. images.json)? | 10 | |
| Does the script correctly create and store GalleryImage objects into an array called mImages? | 10 | |
| Does script automatically advance the slides every 5 seconds? (this is already implemented for you) | 2 | |
| Does calling swapPhoto() show the next image (and its details) from mImages on the screen? | 5 | |
| Does swapPhoto() automatically go back to the first photo after it reaches the last one in the mImages array? | 5 | |
| Does the current photo's metadata (location, description, date) show up when the moreIndicator button is pressed? | 5 | |
| Does the moreIndicator button animate (flip) when it's pressed? | 5 | |
| Is the moreIndicator button center-aligned and overlaid on the bottom of the displayed photo? | 3 | |
| Are the "previous" button and "next" buttons aligned properly (on either end)? | 3 | |
| Are the "previous" button and "next" buttons pointing in the correct direction? | 3 | |
| Do the "previous" button and "next" buttons change opacity when the mouse goes over them? | 3 | |
| Does the "previous" button go the the previous image? | 3 | |
| When on the first photo in the mImages array, does the "previous" button go back to the last image in the mImages array? (i.e. loop around) | 5 | |
| Does the "next" button go the the next image? | 3 | |
| When on the last photo in the mImages array, does the "next" button go back to the first image in the mImages array? (i.e. loop around) | 5 | |
| Is the GET correctly implemented so that I can load an alternate .JSON file? | 10 | |
| Did you make (and can I load) your *extra.json* file? | 5 | |
| Does your *extra.json* file display correctly with at least 5 images with different metadata? | 5 | |
| BONUS: Did you effectively use the StudyTracker tool to keep track of when you were working on this project? (must complete: pre-survey, studytracker, and post-survey) | +6 | |

TOTAL/MAXIMUM POINTS POSSIBLE: 100

(If you score a 96 and do the bonus, you will get a final score of 100).