CSE 154: Web Programming — Fall 2017

Homework Assignment 6: Bestreads

Due Date: Wednesday, November 22nd

Special thanks to Allison Obourn for the original version of this assignment.

Overview

This assignment is about making a web service and using Ajax to retrieve data from it. You will write a PHP service that will generate a variety of data about different books depending on the parameters it is sent. You will also write Javascript code to make requests to this service and inject the responses into a page. Turn in these files:

- `bestreads.php` the PHP service that will supply the book data
- `bestreads.js` the Javascript that will request the information from `bestreads.php` and inject it into `bestreads.html`

`bestreads.html` and `bestreads.css` are provided for you under the resources tab for HW 6 on the course website. You may not modify them.

When the page loads it should display the images and titles of each book that you have data for and the `singlebook div` should be hidden. If the user clicks a book cover or title all of the books and titles should be removed from the page and the `singlebook div` should be shown. You should send a request to the server for data for this book and display its cover image, title, author, rating, description and reviews.

From the course web site you will download a `.ZIP` archive of input files for many books, described on the next page. Unzip this file into the same directory as your HTML, CSS, JS, and PHP files, so that the files will be located in relative paths such as `books/harrypotter/info.txt` and `books/wizardofoz/cover.jpg`. Your code should assume these are the relative paths to use.

Web Service Details

Your `bestreads.php` service will provide different data based upon a couple query parameters named `mode` and `title` that are passed from your Javascript to the page in its URL. The value of `mode` should be `description`, `info`, `reviews` or `books` depending on which information you want. The value of `title` should be a string representing the single book to display. The browser will request your page with a URL such as the following:

```
//some.host.com/path/to/bestreads.php?mode=description&title=harrypotter
```

Your PHP code can store these parameters into variables using code such as the following:

```
$book = $_GET["title"];
```

All of your PHP code should use these parameters' values, and you should never hard-code particular book names. Your code may assume that the browser has properly supplied these parameters and has given them valid values. You may assume that the book exists and has a corresponding folder of valid input data and images. You do not have to handle the case of a missing or empty `mode` or `title` parameter, a value that contains bad characters, a value of a book that is not found, etc. Based upon the `mode` variable's value, your service must output that data. All book data is in a books directory. Each book is stored in a directory named the same as your query parameter in that books directory.

For example, the book harrypotter stores its files in a folder named `books/harrypotter/`. You may assume that the files exist and are valid at all times.

The behavior of each `mode` is described below:

- `mode=description`: The `title` parameter must also be passed with this mode. Your service should locate the file called `description.txt` for the book, and output the entire contents as plain text. This is the only `mode` that should output its response as plain text; all of the others output JSON.
- `mode=info`: The `title` parameter must also be passed with this mode. Your service should output the contents of `info.txt`, a file with three lines of information about the book: its title, author, and number of stars, as JSON.

  Example output:

```
{
    "title":"Harry Potter and the Prisoner of Azkaban",
    "author":"by J.K. Rowling, Mary GrandPre(Illustrator)",
    "stars":"4.5"
}
```

- `mode=reviews`: The `title` parameter must also be passed with this mode. Output an array (in JSON form) containing all of the reviews for the book, the review score, and the name of the reviewer. The reviews are stored in files called `review1.txt`, `review2.txt`, etc. Each file contains one review for each book which is exactly three lines: The reviewer's name, the number of stars they gave the book and their review. If a book has 10 or more reviews, the names will be e.g. `review01.txt`, ... So don't hard-code file names like "`review1.txt`"; instead, look for all files that begin with "`review`" and end with "`.txt`".

  Example output:

```
[
    {
        "name" : "Wil Wheaton",
        "score" : 4,
        "text" : "I'm beginning to wonder if there will ever be a Defense
                  Against The Dark Arts teacher who is just a teacher"
    },
    {
        "name" : "Zoe",
        "score" : 5,
        "text" : "Yup yup yup I love this book"
    },
    {
        "name" : "Kiki"
        "score" : 5,
        "text" : "Literally one of the best books I've ever read. I
                  was chained to it for two days."
    }
]
```

- `mode=books`: Outputs JSON containing the titles and folder names for each of the books that you have data for. Find all the books inside the books folder, and build JSON containing information about each one.
  Your overall JSON object should have one property 'books' that points to an array of books. Each book should be represented by a JavaScript object with two properties: title, and folder. You'll need to extract the title of the book from the book's `info.txt`. The folder property should be set to the name of the folder that contains the resources about that particular book.

  Note, that if you add more books into the books folder, your PHP code should serve these additional books without modification to your PHP file.

  Example output:

```
{
    "books" : [
        {
            "title": "Harry Potter and the Prisoner of Azkaban",
            "folder": "harrypotter"
        },
        {
            "title": "The Hobbit",
            "folder": "hobbit"
        },
        ... (one entry like this for each folder inside books/)
    ]
}
```

Javascript Details

Your `bestreads.js` will use Ajax to request data from your PHP service and insert it into bestreads.html. Here is the functionality your page should have:

- When the page loads it should request all of the books (`mode=books`) from the web service. It should display each of these books by adding the image of the books cover and the books title (in a paragraph) to a div and adding that div to the `allbooks div` already on the page. The `singlebook div` should be hidden.
- If the home button on the upper right is clicked it should do the same thing that the page does when it loads. Even if the button is pressed multiple times very quickly, only one listing for each book should appear on the page.
- When a user clicks on a book cover or title of a book the `allbooks div` should be emptied out and the `singlebook div` should be shown. You should then request the info, description and reviews for that book from the server. The title, author and stars gotten from the info request, and the description, should be inserted into the elements with ids matching their names. You will need to create elements to append the reviews into the page: The title of the review should go into an `h3`, and the score should go into a `span` inside the `h3`. The text of the review should be inserted into a `p` element. Both the `h3` and the `p` can be appended directly into the `#reviews` div.

Development Strategy and Hints

PHP code is difficult to debug if you have errors. Write the program incrementally, adding small pieces of code to a working page, and not advancing until you have tested the new code. The following functions may be helpful:

- count - returns the number of elements in an array
- explode - breaks apart a string into an array of smaller strings based on a delimiter
- file - reads the lines of a file and returns them as an array
- glob - given a file path or wildcard such as "`foo/bar/*.jpg`", returns an array of matching file names
- list - unpacks an array into a set of variables; useful for dealing with fixed-size arrays of data
- trim - removes whitespace at the start/end of a string (gets rid of stray spaces in output)

We recommend developing the PHP parts of the assignment first, as it can be hard to track down problems in JavaScript and PHP at the same time. Before using JavaScript to test your PHP page, consider using your browser to visit your PHP URL, passing the appropriate query parameters, until you are satisfied that your PHP program is producing the correct output. Then, build your Ajax calls to use the PHP web services to make sure you can get the correct data into your JavaScript program. From there, use JavaScript to modify the page appropriately.

Implementation and Grading

For full credit, your JS and PHP code should follow the rules listed in the Style Guide on the course web site. Your page's output must successfully pass the W3C HTML validator. (Not the PHP service but your `bestreads.html` after you have inserted data you have gotten back from the service in it. To validate your page, view the page, then choose View Source in your browser and copy/paste it into the validator.) Your JavaScript code should pass our JSLint tool with no errors. Your `.js` file must run in JavaScript strict mode by putting "`use strict;`" at the top.

Your PHP code should not cause errors or warnings. We will also grade the style of your PHP and JS code similar to a CSE 14x assignment. Do not use the `global` keyword. Use indentation/spacing, and avoid long lines over 100 characters. Avoid redundant code, and use parameters and return values properly. Make extra effort to minimize redundant code. Capture common operations as functions to keep code size and complexity from growing.

Make sure to test your code on all available books from the ZIP file. You may want to think about other edge cases, such as what your page should do if the book has only a single review, etc. You should not have code that depends on particular books or uses `if/else` statements to see which book to display.

Use the "module pattern" shown in class to wrap your JS code in an anonymous function. No global variables or symbols, nor "module-global" vars, are allowed in JS on this assignment; values should be declared at the most local scope possible. The only exception is using a constant variable. You may declare them as a module-global "constant" variables IN_UPPER_CASE. Fetch data using Ajax. Process JSON data using `JSON.parse`. Another grading focus is PHP and JS commenting. We expect comments here similar to CSE 14x. Put a descriptive header (name, course, TA, description) at the top of your code and comment each function and non-trivial section of PHP and JS code explaining the purpose of that code.

Format your JS and PHP code similarly to the examples from class. Properly use whitespace and indentation.

Do not place your solution on a public web site. Submit your own work and follow the course misconduct policy.