Homework Assignment 6: Bestreads

Due Date: Wednesday, May 23nd, 11pm

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.

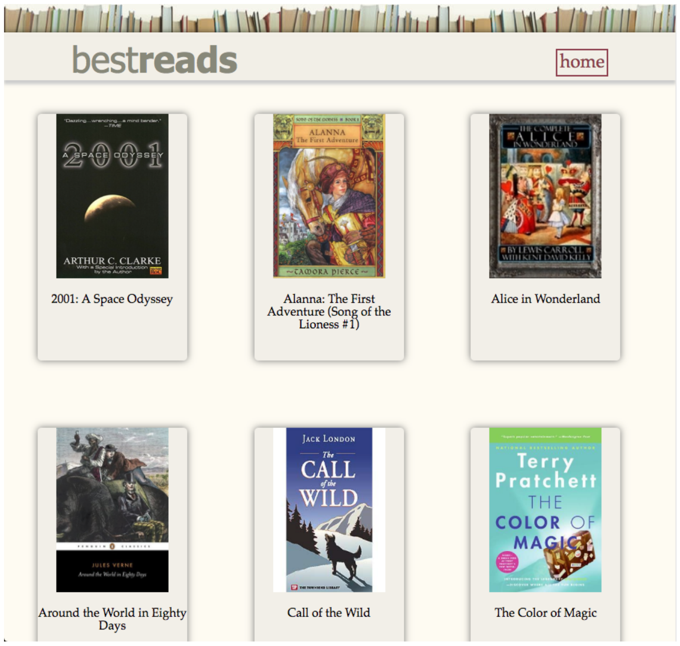


Figure 1: Bestreads front page (rendered in Chrome on Mac OSX)

Files Provided

You should download the following two files we have provided for you for you to use with your JavaScript:

- `bestreads.html` the HTML file for this assignment.
- `bestreads.css` the CSS file for this assignment.
- `resources.zip` the books directory with all of the book images, details and review files.

You may not modify these files as part of the assignment.

Files to Submit

Turn in these files:

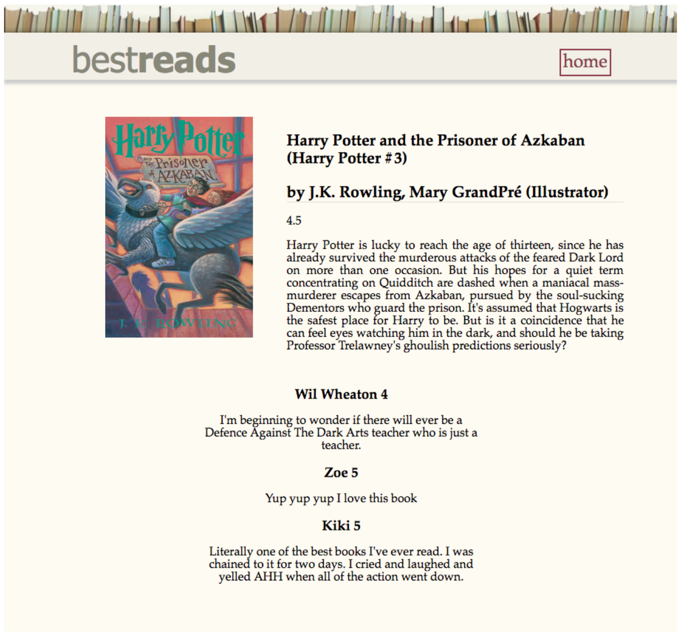
- `bestreads.php` the PHP service that will supply the book data
- `bestreads.js` the Javascript that will request the information from `bestreads.php` and inject it into `bestreads.html`

Learning Objectives

- Continue to practice all of the learning objectives from Homeworks 1-4, including:
  - Carefully reading a specification.
  - Reducing redundancy in your code while producing expected output.
  - Listening and responding to user events using JS event handlers on DOM objects.
  - Modifying your web page using JS and DOM objects.
  - Producing quality readable and maintainable code with unobtrusive modular JavaScript.
  - Clearly documenting your code using JSDoc conventions as specified in the CSE 154 Code Quality Guide.
  - Fetching text and JSON data from a web service using JavaScript `fetch`
- Building an API that responds to GET requests using the PHP language
- Using the PHP language to read information from the server's file system.

External Requirements

When the page loads it should display the images and titles of each book for which you have data in a grid as depicted in Figure 1. The book details page (contained in the `singlebook div`) will be hidden. When 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, as shown in Figure 2.



Harry Potter and the Prisoner of Azkaban (Harry Potter #3)

by J.K. Rowling, Mary GrandPré (Illustrator)

4.5

Harry Potter is lucky to reach the age of thirteen, since he has already survived the murderous attacks of the feared Dark Lord on more than one occasion. But his hopes for a quiet term concentrating on Quidditch are dashed when a maniacal mass-murderer escapes from Azkaban, pursued by the soul-sucking Dementors who guard the prison. It's assumed that Hogwarts is the safest place for Harry to be. But is it a coincidence that he can feel eyes watching him in the dark, and should he be taking Professor Trelawney's ghoulish predictions seriously?

Wil Wheaton 4

I'm beginning to wonder if there will ever be a Defence Against The Dark Arts teacher who is just a teacher.

Zoe 5

Yup yup yup I love this book

Kiki 5

Literally one of the best books I've ever read. I was chained to it for two days. I cried and laughed and yelled AHH when all of the action went down.

Figure 2: Bestreads book detail page (rendered in Chrome on Mac OSX)

The information displayed on these pages will be retrieved through `GET` requests to the API you create. You should send a request to the server for data for this book and display its cover image, title, author, rating, description and reviews.

Web Service Details

The `resources.zip` file that you download from the course web site will contain the 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. This will allow you to use relative paths to the files in each directory, such as `books/harrypotter/info.txt` and `books/wizardofoz/cover.jpg` in your code, which will be necessary in order for your site to function properly when turned in.

Your service will respond to the following two parameters: `mode` and `title`. Valid values of `mode` are `description`, `info`, `reviews` or `books` depending on which information you want to retrieve. The value of `title` should be a string representing a single book that you would like information about. An example request might be:

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

Web Service Implementation Details

- Your PHP code can retrieve these parameters into local 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.

- You should check that the parameters passed in are correct and have valid values. If they are not correct you should output the descriptive 400 error messages. For instance, if the mode is missing you could return the plain text:

  ```
  Error: Please provide a mode of description, info, reviews, or books.
  ```

  and if the mode of description, info, or reviews is given with no title, you might output:

  ```
  Error: Please remember to add the title parameter when using a mode of description, info or rev
  ```

- Each book is stored in a subdirectory in the `books` directory named. The subdirectory book names are all lower case with no spaces. For example, the book harrypotter stores its files in a folder named `books/harrypotter/`. You are to retrieve the book details (based on the `mode` variable's value: description, info, or reviews) in the subdirectory corresponding to the `title` parameter.

- If the book does not exist (i.e. there is no corresponding folder for the name of the book passed in with the `title` parameter), you may choose to output nothing (which honestly is the easiest to do, but least user friendly) or you can choose to output a descriptive 400 error message of your choosing.

- The folder `books/harrypotter/` will contain the following text files `description.txt`, `info.txt`, `review1.txt`, `review2.txt`, ...(the number of reviews may vary).

- All extraneous trailing whitespace from information read in from the above files should be removed before being returned by your web service.

Web Service Parameter Description

Your `bestreads.php` service will provide different data based upon the GET query parameters `mode` and `title` that are passed in. These are described below:

Query 1: Get a book's description

Request Format: `bestreads.php?mode=description&title={title}`
Request Type: `GET`
Returned Data Format: `plain text`
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.
Example Request: `bestreads.php?mode=description&title=harrypotter`
Example Output: (abbreviated)

```
Harry Potter is lucky to reach the age of thirteen, since he has already survived the murderous
attacks of the feared Dark Lord on more than one occasion.  But his hopes for a quiet term
concentrating on Quidditch are dashed when a maniacal mass-murderer escapes from Azkaban, pursued
by the soul-sucking ...
```

Query 2: Get a book's information

Request Format: `bestreads.php?mode=info&title={title}`
Request Type: `GET`
Returned Data Format: `JSON`
Description: 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 in JSON format.
Example Request: `bestreads.php?mode=info&title=harrypotter`
Example Output: (abbreviated)

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

Query 3: Get a book's reviews

Request Format: `bestreads.php?mode=reviews&title={title}`
Request Type: `GET`
Returned Data Format: `JSON`
Description: 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 Request: `bestreads.php?mode=reviews&title=harrypotter`
Example Output: (abbreviated)

```
[
    {
        "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."
    }
]
```

Query 4: Get the list of books

Request Format: `bestreads.php?mode=books`
Request Type: `GET`
Returned Data Format: `JSON`
Description: 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 Request: `bestreads.php?mode=books`
Example Output: (abbreviated)

```
{
    "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 `fetch` 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 article` already on the page. The `singlebook article` 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 home 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, or the container holding both, the `allbooks article` should be emptied out and the `singlebook article` 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 is a combination of the name of the reviewer and the score. The title should go into an `h3`, with the score in 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` section.
- Remember that you need to use `credentials: 'include'` in your `fetch` call when working on Cloud 9. (You should not need to remove this when you submit your assignment.)
- Your fetch should include a `.catch` method call to handle the unlikely event that the `bestreads.php` web service returns an error. If an error does occur the `error-message article` should be shown with an error message displayed to the user. Whenever error is not present, the `error-message article` should be hidden.

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.
- 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.

Internal Requirements

- For full credit, your page must use valid JS and successfully pass our JSLint (https://webster.cs.washington.edu/jslint/) with no errors. JSLint warnings are nominally ok, but you should read them and understand why they are warnings and be thoughtful about why you might or might not fix them. Note that JSLint has been changed for this assignment in an effort to help you achieve better code quality.
- Your `JS` should also maintain good code quality by following the guide posted on the class web site. We also expect you to implement relevant feedback from previous assignments.
- Your `.js` file must be in the module pattern and run in strict mode by putting `"use strict";` within your file.
- Avoid unnecessary fetch requests to web services - you should only make requests where needed to update DOM elements based on the expected behavior outlined in this spec.
- Your JS and PHP files should have adequate documentation. The top of the file should have a descriptive header describing the assignment. You are to use the JSDoc commenting style for each function and module-global variable in your `JS` file (see the code quality guide on our website for more details on how to write JSDoc). Additionally, complex sections of code should be documented. You may use inline commenting within `JS` functions. Your PHP variables and functions should be similarly documented in a style like JSDOC documenting the method's description, parameters and return values.
- Format your code similarly to the examples from class. Properly use whitespace and indentation. Use good variable and method names. Lines of code should be fewer than 100 characters long.
- Variables should be localized as much as possible. Minimize the use of module-global variables. Do not ever store DOM element objects, such as those returned by the `document.getElementById` or or `document.querySelectorAll` functions, as module-global variables.
- If a particular literal value is used frequently, declare it as a module-global `const IN_UPPER_CASE` and use the constant in your code.
- You should make an extra effort to minimize redundant JavaScript code. Capture common operations as functions to keep code size and complexity from growing. You can reduce your code size by using the `this` keyword in your event handlers.
- Separate content (`HTML`), presentation (`CSS`), and behavior (`JS`). Your `JS` code should use styles and classes from the `CSS` when provided rather than manually setting each style property in the `JS`. For example, rather than setting the `.style.display` of a DOM object to make it hidden/visible, instead, add/remove the `.hidden` class in the provided CSS to the object's `classList`. You may find the `classList toggle` function helpful for toggling certain classes.
- Fetch data using Ajax. Process JSON data using `JSON.parse`.
- You should not use any external JavaScript frameworks or libraries such as jQuery to solve this assignment.
- Your PHP code should not cause errors or warnings. Add `error_reporting(E_ALL);` to the top of your `.php` file so errors are thrown and not kept silent.
- Do not use the `global` keyword.
- Utilize PHP functions for good readability. Capture common operations as functions to keep code size and complexity from growing.

Grading Rubric

This assignment will be out of 20 points. The key areas we will be looking at assess directly relate to the learning objectives, and your matching the specification for the external behavior as well as the internal correctness of your code. **NOTE:** While we can not guarantee the same distribution of points, past rubrics have been split with 50% of the points allocated to external correctness and the 50% for internal. Thus a **potential** rubric **might be** summarized as:

External Correctness (10 pts)

- Web service produces the correct results for the parameters given
  - description mode (plain text)
  - info, reviews and all books mode (JSON)
- Service client (JavaScript)
  - displays all book images and titles on load
  - displays info for a single book on a click
  - home button displays all book images and titles

Internal Correctness (10 pts)

- PHP
  - follows class code quality guidelines
  - avoids redundancy
  - uses function to encapsulate functionality
  - functions are well documented including parameters and return values
- JavaScript
  - Passes CSE 154 JS Lint and follows Code Quality guide
  - avoids redundancy
  - uses function to encapsulate functionality
  - no globals (module pattern); minimizes module-globals; uses variables well
  - uses AJAX and JSON properly
- All functions are clearly documented using JSDoc as described in the Code Quality guide
- Otherwise good quality code - a catch all for things like indentation, good identifier names, long lines, large anonymous functions, etc.

Academic Integrity

As with any CS homework assignment, you may not place your solution to a publicly-accessible web site, neither during nor after the school quarter is over. Doing so is considered a violation of our course academic integrity policy. As a reminder: The University of Washington has an entire page on Academic Misconduct on their Community Standards and Student Conduct Page. Please acquaint yourself with the University of Washington's resources on academic honesty, and in particular how academic misconduct will be reported (which has been changed for 2017).