Homework for Database Set Up and JS Routing

1. Create a Database and Populate it with Data.


CONTENTS

1. OVERVIEW .................................................................................................................................................. 2
2. DATABASE PART ......................................................................................................................................... 2
3. WEB SITE PART ............................................................................................................................................ 2
4. SAMPLE CODE FOR WEBSITE PART (USER INTERFACE REUSE AND ROUTING) ......................................... 3
5. JAVASCRIPT DEBUGGING .......................................................................................................................... 4
6. PROJECT ORGANIZATION .......................................................................................................................... 4
7. SUBMISSION REQUIREMENTS ................................................................................................................... 4
8. GRADING AND SAMPLE DEDUCTIONS (FOR BOTH PARTS: ROUTING AND DB SETUP) ................................ 5
1. Overview

In this homework, you have a **database part** and a **website part**:

- For the database part, you’ll create database tables, populate them with data, and run some SQL SELECT statements against your data. You’ll put screen captures from MySQL WorkBench into a word document to show what you did.

- For the website part, you will achieve reuse of the HTML/CSS user interface by using a home grown JavaScript routing “framework” code that enables the User Interface to be reused. When the user clicks on “Blog” from the nav bar, Blog content gets injected into the content area. When they click on “Home”, the home content is injected into the content area. This style of web design is called “Single Page Application”, because your application is primarily stored in a Single Page with only the content area getting changed based on user actions.

2. Database Part

For the database part of this homework, please see a separate document entitled “DB Setup”.

3. Web Site Part

- **Your home and blog links shall change the content when clicked**, implemented using a home grown JavaScript “routing framework”. This means you’ll have a JS file named home.js and another named blog.js and each will be invoked (when the associated nav link is clicked), injecting the proper content into the content area.

- When you **click on home**, you should just see your web site functionality overview enticing users to visit the site. This is the first 1-2 paragraphs from your last homework. You’ll move the rest of that content into your blog section (see below).

- When you **click on blog** your content area shall contain two blog entries – one for Homework 1 and one for Homework 2.
  - The **blog entry for Homework 1** shall contain what you move out of your content area from homework 1 (where it described your web design/development experience, your proposed database table, and your experience with the homework - what you found hard/easy/valuable).
  - The **blog entry for Homework 2** shall:
    - Tell what you found hard/easy/valuable about the JS routing part of the homework.
    - Describe your database experience.
    - Provide a working link to the word document that shows your database work for this week.
    - Tell what you found hard/easy/valuable about the Database part of this homework.
  - Each entry in your blog content shall use a header (e.g., h1 or h2 o r h3) that indicates the homework number and name like “Homework 1 Home Page”, “Homework 2 Database and JS Routing”, etc.
  - Note: as you complete future homework assignments, you will add a blog entry for each homework.

- Your site shall continue to meet last week’s requirements (drop down menus, images, fixed titleNav and footer, fluid layout, etc.). Check last week’s homework if you need to. Improve your layout if you can.
4. Sample Code For Website Part (User Interface Reuse and Routing)

This week’s homework (in the course web page) provides is a zip file with various folders of sample code. The folders show the bits and pieces that enable you to create your own simple “Routing Framework” using JavaScript.

- **01_Two Pages Copy Paste** shows you how a beginner would create a web site — with no reuse of the User Interface code. They would just make a copy of the index page, call it blog.html, then modify the content area of the blog page and the result looks pretty good. The only problem is maintenance. If you had 100 pages in your site and then you needed to change the title or the footer or a nav link, a programmer would have to make that same change in 100 files. Not good!

- **02_Internal Links** just shows you how internal links work. When the href attribute of an `<a>` anchor tag looks like this `<a href="#myId">`, then clicking the link makes the page scroll down to whatever element has that id (on the page).

- **03_Reuse without Routing** shows how to make JS components (that put some HTML into an element on the page). Since this version does not use any links -- the browser’s forward/back buttons cannot be used to go back to previous link.

- **04_Routing Multi-Branch If** uses links and introduces a routing function that is invoked whenever the browser’s address bar changes. Instead of using a sophisticated routing table, the router function has a multi-branch if statement that invokes the correct component function (e.g., home or blog) based on what link was clicked (e.g., #/home or #/blog).

- **05_javascript_review** reviews the JS examples from the previous module, regarding JS flow of control and JS data types (that in JS, a function is a first class data type just like integer or String or object).

- **06_Associative Array**. An associative array is like a “hash table” where you can store key/value pairs. With associative arrays, the “index” (inside the square brackets) is the key, usually a String, not an integer. The value can be anything, like a string or an object or a function.

- **10_Routing Table** uses a hash table (associative array) instead of a multi-branch if statement to determine which function to call based on the link in the address bar. Here is how the hash table can be populated -- the link is the key and the value stored along with the key is the function to be called whenever that link is invoked.

  ```javascript
  routes['#/home'] = home;
  ```

- **11_Parameter Objects** shows how the benefits of using a parameter object instead of a parameter list. Parameter objects are more self documenting (you see property names to indicate what you are setting, instead of relying on which parameter it is, e.g., 1st or 2nd, etc). It is also convenient for the HTML coder to skip setting any parameters s/he does not care about, allowing the provider code to just set those to default values.

- **13_Routing Framework** is encapsulated. It has consumer code (in the HTML page) and provider code (in the JS file). With this code, the consumer code (the JS in the HTML page) sets up the routing table and then passes this to the provider code (in the JS file) which handles everything from there.

- **14_Routing Framework with DropDown Menus** just combines the routing framework code with the drop down framework code (just to make sure there’s no problem with both frameworks working together).
5. **JavaScript Debugging**

NetBeans provides context sensitive error reporting, so pay look for these errors and address them. Since JavaScript is interpreted, not compiled, you only get runtime errors. You only see these runtime errors if you actually run the line of code that has the error AND you have the Chrome console open (Press F12 and click on “Console”).

6. **Project Organization**

Your project shall be organized as shown below.

- In the js folder, we differentiate between general purpose reusable JS code (e.g., dropdownFw.js and routeFw.js) and specific “components” created by JS code, e.g., the blog.js component creates blog content and injects that into the content area. The home.js component creates home content and injects that into the content area.

```
  Web Pages
  WEB-INF
  icons
  js
    components
      blog.js
      home.js
      dropdownFw.js
      routeFw.js
  pics
  style
  index.html
```

Take advantage of the class Template document that is provided along with the sample code.

7. **Submission Requirements**

- As usual, test your web site locally (including clicking from home to blog to home, then clicking from blog to the database document) – all links should work properly.
- Then publish, then test all links again.
- Then submit a zip file of your whole web project folder into Canvas.
8. Grading and Sample Deductions (for both parts: Routing and DB Setup)

- **-9 for Not Publishing**: We perform functional testing each week based on what you have published. You need to test locally, publish, then test what you published.

- **-9 for Lack of Canvas Submission**: If there is ever any question about a grade, we go by the code that was uploaded into Canvas at the time the assignment was due.

- **Up to -4 for Improperly Designed Database Tables**: We will check that your data model meets all of the requirements listed in the database document (requirements such as data type, PK, FK, null-able, unique). This is very important, since you will lose points in many future homeworks and the project if your data model is not as specified.

- **Up to -2 for Non-Realistic Data**: You were asked to enter realistic data so that your web application looks good when it begins to display data on its pages.

- **Up to -3 for Problems with SELECT Statements** as prescribed in the database document.

- **Up to -3 for Unprofessional / Inappropriate Home Page Content**: The “marketing material” from your home page should be of a quality that acceptable by a “real company” that might be paying you to create their web site. The marketing material (what you say your web site will do) must be able to be supported by your choice of layout for your “other” database table.

- **Up to -8 for Lack of Originality of Project and/or Data Model**: As in most homework assignments, points will be deducted if your submission is too similar to the sample(s) provided or to another student in the class.

- **-4 if Routing not implemented**: For example, if a student manages to get their navigation bar to work pretty well by copying and pasting code from an index.html to a blog.html and just linking back and forth from one page to another, without using JavaScript to create a Single Page Application to achieve User Interface code reuse...

- **-1.5 if there is no Database Blog Entry.**

- **-1.5 if there is no Blog Entry about UI Code Reuse (Routing) using JavaScript.**

- **Up to -2 for Look and Feel**: Make sure to meet the requirements for the first homework (Home Page). If you had issues, you should have fixed them by now. For example, you should have drop down menus and nice layouts that do not render slowly (e.g., due to too large image files), should respond well when you narrow and expand the width of the browser, and should meet requirements for your first homework (e.g., all text legible, no text close to visible edges, not overly similar to the sample code).

- **Up to 2 for Syntax Errors**: When we View Source (in Firefox), we should not see any red syntax error messages.