The Refactor Factor

Web development principles and design patterns: separation of concerns, don't repeat yourself, high cohesion, low coupling, low representational gap, version control, testing, refactoring

Refactoring

Refactoring is an important Web application development skill that involves identifying violations of software development principles and design patterns and applying those principles and patterns to produce an improved solution. *The Pragmatic Programmer* defines refactoring as "rewriting, reworking, and rearchitecting" software.

For this assignment, I have given you a simple Web site with some dummy content that uses JavaScript to select between different site themes. This application works, but it is written badly. Your assignment is to refactor the application to produce a better design. Although there is not a "correct" answer in absolute terms, I will tell you that there are at least 5 design problems with this site as it is currently implemented.

Assignment

Your task for this assignment is to:

1. Identify at least 5 problems with the site implementation
2. Design and (partially) implement a more modular, organized design. For each of the 5 problems you identified, write a description for your solution design including:
    a. A description of the problem. What patterns or design principles are violated?
    b. The solution design. How will you design a solution to the problem? What patterns or principles, if different from above, inform your design decisions?
3. For at least 3 of the problems you identified, implement a solution guided by your design. Provide an overview description of your implementation changes in the assignment documentation.

Process Requirements

Approach the problem methodically and document the problems you uncover and your solutions. Remember to use an iterative approach; you can use this methodology for each iteration:

1. Understand the problem
2. Design a solution
3. Implement and test your solution
4. Consider other solutions

In particular, if you find yourself making changes that alter or invalidate previous changes, **don't delete the documentation of your previous solution!** Part of iterative software development involves continually examining existing code and designs and updating them. In your documentation, leave the descriptions of your previous solutions and reference them and descriptions of altering or invalidating solutions you come up with later.

As with every software development project you undertake, you should:

- use version control with descriptive commit messages
- apply the Web development patterns and principles we have discussed this semester
- adhere to the software craftsmanship code quality requirements spelled out in the syllabus

Turn in: design/process documentation, version control change log, reimplemented application: HTML, CSS, and JavaScript