Home Page Homework

Table of Contents

1. Homework Overview ........................................................................................................................................... 1
2. BEFORE YOU START ............................................................................................................................................... 2
3. CHOOSE A WEB APPLICATION TOPIC .................................................................................................................. 3
4. LEARN ABOUT WEB DESIGN .................................................................................................................................. 3
5. REQUIREMENTS FOR YOUR WEB SITE FOR THIS HOMEWORK .............................................................................. 4
6. GRADING AND SUBMISSION .................................................................................................................................... 6
7. DEDUCTIONS ............................................................................................................................................................ 6
8. SUGGESTED APPROACH ............................................................................................................................................ 7

APPENDIX A: HOW TO DEBUG HTML AND CSS ........................................................................................................... 8
APPENDIX B: HOW TO MOVE AN INTERNAL STYLE SHEET TO AN EXTERNAL STYLE SHEET .............................................. 10

1. Homework Overview

In your first two homeworks, you proposed a data model and a topic for a web site. Now, you will implement a two page web site that implements your proposed web site topic.

In this homework you will:

1. Install and test “NetBeans Bundle version 8.2”.
   - Use the NetBeans Bundle for this first homework even though you could use some other editor for client side coding. Reason: we will need the NetBeans Bundle to write server side Web APIs, so it’s best to get used to it in the earlier/easier homeworks.

2. Install and test MySQL Workbench (to avoid installation / setup problems that could impact your work next week).

3. Decide what you want to do for your web application this semester. Parts of your web app (and database) will be the same for all students, but each of you will have your own unique database table (and related web app functionality).

4. Learn basic web design, including basic HTML elements, CSS properties, web color codes, and how to create some simple layouts.

5. Create a home page with an image or two, and a color scheme that is appropriate for your topic.

6. After testing locally, publish your web site, test your published site, and submit your homework by attaching a zip file of your website folder into Canvas.
2. Before You Start

1. Learn where to find course homeworks and related tutorials. Just google “sallyk temple” and click on your course number under the “teaching” tab. Each homework is listed there, along with links to related tutorial material.

2. Install the NetBeans Bundle version 8.2 at home (following instructions provided in a separate document – see link from the course web page under this week’s homework). Even if you already have NetBeans installed, you need to install “bundle” which includes the Apache web server and Glassfish Application server.
   - After installing the NetBeans Bundle, create a project and test your ability to add, view, run, and modify a HTML page.
   - Recognize that browser caching can be problematic, for example, if you just changed a style in your CSS file and you want to see the effect of that change. You may need to hold down the control key when you click your browser’s refresh button - or even clear the browser’s cache.
   - Learn how to reformat your code (from the NetBeans menu, “Source – Format” or Alt-Shift-F).
   - Learn how to check for syntax errors. NetBeans should identify many syntax errors (red bubbles to the left of the line with the error), but NetBeans may not catch all of them. Although we primarily use Chrome for development, Firefox shows HTML/CSS syntax errors better. So, check your syntax by right clicking and “Viewing Source” from Firefox (errors colored red).
   - If you have a laptop, please bring it (with the NetBeans bundle pre-installed and tested) to your first lab. If you are not able to bring your laptop installed with the NetBeans bundle, you can use one of the lab PCs, but you will have to move your project from the lab PC to your development PC/Mac – not a huge problem, just less convenient.

3. To avoid setup issues for next week, install and test MySQLWorkbench by following the instructions (find link under this week’s homework from the course web page):
   - How to install MySQLWorkbench
   - How to connect to your Temple database (shows you how to find your database credentials, etc)

4. Learn how to publish a page. We typically do this in your first lab activity – just to be sure everyone knows how to publish and that there are no technical problems (especially for students who added late and need web folders created for them). Publishing instructions are described in a separate document (link provided in the course web page under this week’s homework).

5. If you are new to web design,
   - study the “Web Design” links that are referenced from this homework in our class website. This tutorial explains basic HTML, basic CSS, web color encoding, and how to select colors. It also tells you how to reduce image file sizes so that your pages do not load slowly.
   - study the sample layouts that are referenced from the Web Design Tutorial page.

6. Email your instructor ASAP if you have ANY installation and/or setup problems.
3. **Choose a Web Application Topic**

So that we can cover interesting topics related to client side coding (JavaScript), I’m requiring your database design be very simple. Basically (if you do all the homeworks), your web application will allow users to add/update/delete records into a user table as being able to add/update/delete records from some other database table (a table that is unique to each student). So, think about what attributes you want in your table. The guidelines for what you’ll need in this table are: id, unique name, image URL, long character (description), at least two user optional non-character fields (e.g., integer, decimal/money, date). You can add more fields if you like, but don’t go crazy because it will be more work for you with every homework assignment. You will not create your database table in this homework – you will do that later. This week you are just being asked to select a topic and a database table that match each other (unique to you).

4. **Learn About Web Design**

Follow the links from this homework (from the course web page). If you have no web design experience, you’ll need to study this material and experiment with the code examples (the “try it yourself” pages are really awesome). If you have some web experience, check to be sure you already know the information presented. These are the concepts for which you are responsible:

- All of the HTML and CSS concepts that are listed on my "Web Design: Getting Started with HTML and CSS" page - excluding "HTML Forms" and "Advanced CSS" (which we will learn later).

- Be sure that you understand CSS float. An element that is floated “retains its space in the normal flow of the page” but it is moved and this affects the elements that are next to and after the floated one. A typical use for float is an image (so that the text will "float around it", like a newspaper layout), but we can also float divs and other block elements. Know when to use "clear:both" (to prevent elements below from moving up).

- Be sure that you understand and CSS fixed positioning. When you apply fixed positioning to an element, that element is "removed from the flow" so the rest of the elements are laid out according to the "normal flow" – as if the fixed element does not exist. Thus, the fixed element will likely be behind or in front of other elements. To control which element is "on top", you use the CSS "z-index" property. To prevent one element from hiding the other, apply padding or margin.

- The basics of working with images: image file formats (jpg, png) and how to reduce image file sizes (so that your pages do not load slowly).

- Web color codes: RGB (red/green/blue) and RGBA (red/green/blue with opacity), understanding the concept of saturation (fully saturated is bright and colorful, fully unsaturated is a black and white image), how to select colors from an image, how to blend colors.

- How you can use all the above information to create a simple layout such as the one that is described in the next section.
5. Requirements for your Web Site for this Homework

1. Your index.html shall
   - include HTML elements with the following attributes: id="title", id="nav" (the navigation bar), id="content", and id="footer". You can have more elements, of course.
     - I ask for specific id attributes to ensure that students don't just submit a page that they copied from the internet or from a previously designed web site. It also facilitates grading and layout help.
   - have a navigation bar that implements drop down menus such as the ones provided in sample code. This week, the links won’t work, but the drop down functionality shall work. HTML elements in your nav bar shall have CSS classes (like dropdown, dropHeader, dropContent) that style multiple similar elements.
   - have title text (that stands out visibly on the page) – as well as a <title> tag specified in the <head> section (which shows up in your browser tab heading).
   - have a content area that
     - looks professional,
     - has at least one external link (href= "http://...") with different styling than your navbar links.
     - describes functionality of your website and entices viewers to visit the site (as its first section).
     - has a second area that shows the database table name and field names/data types that you plan to use for your web application. Use <ol> (ordered list) or <ul> (unordered list) with <li> (list items) so that it your field names are easy to pick out (one per line).
     - has a third area that is a "blog entry" that
       - describes your web development/design experience, if any (none is required),
       - tells what you learned in this assignment and tells what you found easy and hard about this assignment..
   - Your home page may look odd this week, but in the next homework, you’ll move the blog entry.
   - have a footer area with your name in it.
   - include at least one image (either <img> tag or background image) somewhere on the page.
   - load quickly (by keeping your image file sizes small, like 500K or less). Either select images with small file sizes or reduce image file sizes as described in the "Working With Images" section of the 3344 web design tutorial. To experience page load speed as would be experienced by a first time visitor, load the page after emptying your browser cache and/or visit your page from a new PC/MAC.

2. Your index.html page's style rules shall be placed in an external style sheet and:
   - implement a professional looking color scheme that is appropriate for the topic of your web site.
   - have a fluid layout (not fixed), which means that your HTML elements nicely wrap when you narrow the browser (instead of having a horizontal scroll bar).
     - To achieve this, avoid specifying widths, but when you must, use percentage widths.
     - Apply the min-width CSS rule to prevent certain items (like nav bar links) from unwanted wrapping when the browser gets really narrow. When the min-width is attained, your page will begin having a horizontal scroll bar. For example, you might have a div (let's call it titleNav) that surrounds your title and your nav element, you might apply min-width to this titleNav.
     - To test your layout for fluidity, slowly narrow your browser and make sure no unwanted wrapping or overlaying occurs (as it narrows). If it does, use min-width to prevent that, but your min-width should not be so large that there is always a horizontal scroll bar.
• have **nav bar links** that
  o are not underlined (CSS declaration "text-decoration:none").
  o are styled differently than the links in your content. Achieve this by using compound CSS selectors that are described at the top of my “Web Design Tutorial” page.
  o have colors that stand out visually against their background - always: before clicking (:link), after clicking (:visited), and while hovering (:hover),
  o visibly change (e.g., font color and/or background color) when you hover over them – either the link or the div holding the link.
• make all **text readable** (large enough and with enough contrast to the background)
• provide **padding and/or margins** so that no text is too close to any visible border.
• use the CSS **"float"** property at least once.
• have a **fixed footer** using CSS positioning (add margin or padding to the bottom of your content so that your footer will never cover up the end of your content – once you have enough content).

3. **All code files** (including index.html and your style sheet) shall:
• be **syntactically correct** (check this by "Viewing Source" from Firefox, even if you designed with Chrome) and properly **indented** (use NetBeans "Source – Format"). **Tip:** If you are viewing source from a HTML page, you can click on the style sheet (or any images) that are referenced in that code.
• is neat and organized, with **no irrelevant code**.
  o In other words, everything in your style sheet must affect the look and feel of your index page - if you try something that has no effect, remove it. Since style is hierarchical (each element inherits style from its parent element), put your styles at the highest level possible (so you do not have to copy/paste style to a lot of low level children elements).

4. **Project Organization.** To assist us with grading, please put your last name in your project name. This helps us with grading. There is a “rename project” option if you right click on the project name - be sure to check off that you also want the folder to be renamed.

As you start laying out your home page, you may just have a simple index.html file (with internal style sheet) plus a “pics” folder with some images inside.

After you move your styles to an external style sheet and incorporate the drop down JavaScript code, your NetBeans project shall be organized like this:

5. **Your home page homework shall NOT use Bootstrap** (at least, not until after homework 1 is graded). Bootstrap is a great CSS framework, but it obfuscates many of the concepts we are trying to learn here.
6. Grading and Submission

Grading will be based on timeliness, meeting the requirements, and effort/professionalism/originality. Students with no HTML/CSS experience can get full credit for this homework. Your layout does not have to be fancy and fantastic. It can be simple, as long as it meets the requirements and looks reasonably professional.

When you have completed all the requirements, tested locally (and syntax checked), and published (and tested what you published), submit a zip file of your website into Canvas (No RARs). Your zip file submission is important because we base the late penalty off of the date/time of submission. Also, if there is ever a question about how we graded a homework, we will go by the code that was submitted, not by what is currently published.

7. Deductions

- -9 if no zip file submitted into Canvas (by the due date).
- -9 if no pages published by the due date.
- -2 if no image used (as regular image or background image).
- Up to -2 if your pages load too slowly - you should have selected images with small file sizes or doctored up the images so that the total of all images on page around 500K or less.
  - To test this for yourself before submitting your homework, clear your browser’s cache, then reload your index.html page. Or visit your page from a new computer that you have not been using for testing.
- Up to -3 for lack of professionalism.
  - Would the content compel a user to visit the site?
  - Does it utilize "white space" effectively to emphasize important aspects of the page?
  - Have you styled your nav bar links so that they look professional (and are not underlined)?
  - Does your content include an external link that is styled differently than your nav links?
  - Does it have any text that is too close to visible edges?
  - Is all text legible (including link text before clicking, after clicking, and while hovering)? Is it large enough to be easily read? Does it have enough contrast with its background?
- Up to -2 for not having a fixed footer (and making sure that the last content is not covered up by the footer – add margin or padding to the bottom of the content area).
- Up to -1 for not having your name in your footer (this helps us grading).
- Up to -3 for lack of fluidity. When we narrow the browser,
  - We should never see unwanted wrapping of nav bar elements.
  - We should never see any overlapping text (the min-width CSS property should have been employed to prevent this).
  - We expect the horizontal scroll bar to kick in (to prevent unwanted wrapping and overlay of text), but it should not kick in until the browser is narrower than about 600 - 800 pixels. Otherwise users with low resolution screens would be forced to use the horizontal scroll bar just to see the whole page.
  - Note: this is an important and practical test. User screens will have different sizes and different resolutions that the screen you used to develop your layout.
- Content area:
  - Up to -1 if your content area does not describe website functionality realistically and compel users to visit the site.
  - Up to -2 if you do not explain your database table with fields that meet the requirements (number and type of fields etc) AND/OR your database table would not support the web site functionality you described.
  - Up to -2 for lack of blog entry in the content area that describes your web experience, what you learned, and tells what you found easy and hard about this assignment.
- Up to -3 for lack of originality.
  - Your code may not be overly similar to sample code that is provided, nor overly similar to any of your class mates.
  - Your code may not be copied from the internet nor from a site you may previously have made. This is one reason I asked you to specifically name your ids (e.g., "title", "nav", "content", etc). "footer" - the other reason is so that we can easily identify areas of your layout and assist you with layout issues.
- When we “View Source”:
  - Up to -3 for HTML/CSS syntax errors which would show in red font from Firefox View Source.
  - Up to -3 for lack of code reuse.
    - Your pages should both reference an external style sheet.
    - Your pages should utilize JavaScript to inject the home content and the blog content into the content area when the user clicks on the links in the nav bar. The content snippets, e.g., homeContent.html, blogContent.html contain partial HTML (not start with <html> and end with </html>).
  - Up to -3 for poor coding style. Your code should have proper indentation, some comments, good naming, and no unnecessary code.

8. Suggested Approach

1. Carefully read this document. If you have little or no HTML/CSS experience, it is imperative that you follow the links and learn about web design. Study the sample layouts (referenced by the Web Design Tutorial page). You can create your layout from scratch or select one of the samples to use as a starting point for your homework. If you use one of the samples, don’t just copy/paste and make a few changes. You must understand what you are doing or your grades will suffer and you will have difficulty in subsequent homeworks.
2. Select a look and feel for your home page that is appropriate to the topic you selected. If you are having trouble coming up with a color scheme for your home page, find an image that you want to use in your home page, an image that has colors that you’d like for the color scheme for your layout. Use a color picker browser plug-in (like ColorZilla) to extract the color codes from the image. As described in the Web Design Tutorial, use a color blender (like the Meyer web site) to make variations of these colors (lighter/darker, more/less saturated).
3. Begin the web design cycle (edit, save, view, repeat), creating your home page (index.html). Check out the "How to Debug HTML and CSS" appendix within this document. If you think that you are not seeing updated style (especially after you have moved your styles to an external style sheet), you may have to hold down the control key while you refresh the browser (this is supposed to clear out cached files) OR right click and View or Run the page (opens up a new tab) instead of just refreshing the page OR clear out caches files from your browser history. Once index.html meets all of the requirements listed in the previous section, do one final syntax check (right click and View Source from Firefox).
4. Test your web site locally (the home link and the blog link) using NetBeans (right click and "run" option). Make sure that all links work and that the layout does not "jump" from page to page.
5. Publish your website to the web server, by following the instructions that are provided in a separate document (link provided from this homework of the 3344 homeworks page).
6. Test your published pages. You may have forgotten to upload a file, or a file may not have the right access privileges on the server, or you may have an issue with upper/lower case file references – windows is not case sensitive regarding file names, but unix is.
7. Submit your homework by attaching a zip file of your web application into the homework 1 assignment in Canvas.
Appendix A: How to Debug HTML and CSS

If you are "lost", it is probably because you are new to web design and you have not studied the links about Web Design that were provided with this homework. Take time to study those first.

To debug HTML, keep an eye out for the red bubbles that NetBeans places to the left of every line that has a syntax error. Nesting is just as important in HTML as is in java.

- Take advantage of NetBeans "Source – Format" (menu option) to automatically indent your code and help you understand the nesting within your HTML code. (Proper indentation is also a homework requirement.)
- Most HTML elements have a starting and ending tag, e.g., <div id="content"> ... </div>. If you click on the starting or ending tag of an element, NetBeans highlights the corresponding ending or starting tag. I suggest that you put a comment next to ending tags that are far away from their matching starting tag. Remember that HTML comments look like this  <!-- your comment -- >  not /*...*/ and not //

CSS TERMINOLOGY. This is a CSS RULE:

To debug CSS,

- One (possibly "old fashioned") way to debug debug HTML/CSS is to put temporary borders on various HTML elements (like divs). This will let you determine where the padding or margins may be coming from. border: thin solid red;
- OR you can click on F12 in Chrome to bring up the debugger and click on the "elements" tab. For any element that you select, Chrome will show you all the style rules that apply to that element.

Remember these things about CSS:

- If you have syntax error in a declaration, CSS ignores all declarations until the end of the rule. Luckily NetBeans identifies CSS syntax errors now so this is less of a concern. CSS comments can only use this form /* ... */ Any other style comment, like //, will be a syntax error (and nullify any subsequent declarations).
- If you are having trouble getting a declaration to take effect,
  o Move the declaration to the end of the rule. You might have specified the property twice (last one wins). Also, if several rules are applied to an element, a declaration (say background-color) would be based on the most specific rule (for example, inline style in the html tag overrides styles from a style sheet).
  o Make sure your browser is referencing the latest files (instead of possibly cached files) by holding down the control key while refreshing or clear your browser’s cache to see your latest changes.
- With CSS, less is better. Put your declarations at the highest level. For example, put your most used font-family in the body and only specify a different font-family for the areas that need something different.
- If your style sheet becomes cluttered, clean it up, organize it, and remove any unused rules (this is also a homework requirement).

If you are having trouble with colors, some would say visit a site that provides color schemes, but then you’d have to find images that go with the scheme you chose.

- I recommend that you select an image upon which you want to base your look and feel, then select your color scheme colors from that image. As mentioned previously, you can download/install a color chooser browser plugin like ColorZilla that can extract color codes from an image. I recommend selecting two colors, but you might select three (counting lighter/darker or more saturated/less saturated variations of those colors).
Visit a color blending website like Meyer Web that will create color gradations that are between two colors that you enter. When making a color lighter, you probably want to blend it with gray (not white) or you will get pastel baby colors. This is called making a color less saturated. A less saturated color will have less difference between the R, the G, and the B elements of the color code. When you make a color darker, you probably want to increase saturation (a stronger color, less black/white, more difference between the R, the G, and the B color codes) – otherwise the color will look pretty black. Remember that a homework requirement states that all text must be visible on its background, even links (before clicking, after clicking, and while hovering).

If you are using float, remember that this property is odd in that it affects the elements around it. For example, if you float a picture right, not only does the picture does go to the right, but the text before and after the picture is also affected. Remember to add the rule (below) where you want the floating behavior to stop (e.g., you want the text of the next paragraph to not float around the picture).

clear:both;

If you are using fixed CSS positioning (e.g., the nav bar is fixed to the top of the screen), remember that these elements are "removed from the flow" which means all other elements act as if that fixed element never existed. This can cause elements to be placed on top of each other. Remember to use z-index to control which element is on top (the element with higher value of z-index will be on top). A static HTML element (i.e., an element that is not affected by any CSS "position" declaration) has a z-index of 0, even if you try to apply a different z-index to it. In other words, you cannot apply z-index to a statically positioned element. You may have to add padding or margin to the top of your content to prevent the beginning of the content text from being hidden underneath a fixed title/nav. You may have to add padding/margin to the bottom of your content to prevent the ending text of the content from being hidden below a fixed footer.

Your layout should be "fluid", using percentages to specify widths (not pixel count). If this creates unwanted wrapping (when browser is at various widths), apply min-width to one of the elements like titleNav or nav, etc. As the browser narrows to that width (that you specified with min-width), it will provide a horizontal scroll bar and you will avoid the unwanted wrapping. It is fine to use pixel count for vertical measurements (margin, padding, font-size).
Appendix B: How to Move an Internal Style Sheet to an External Style Sheet

Eventually, we will have more pages than just the home page and so we want all pages to reference a single style sheet (for ease of web site maintenance). When all files reference a single style sheet, then you can change the style of ALL pages by making one change to the style sheet. Before making the changes below, back up your index file.

<table>
<thead>
<tr>
<th><strong>index.html</strong> BEFORE (style in the head)</th>
<th><strong>styles.css</strong> (NEW external style sheet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!DOCTYPE html&gt;</td>
<td>body {</td>
</tr>
<tr>
<td>&lt;html&gt;</td>
<td>background-color:pink;</td>
</tr>
<tr>
<td>&lt;head&gt;</td>
<td>color:green;</td>
</tr>
<tr>
<td>&lt;title&gt;Title that shows in browser tab...&lt;/title&gt;</td>
<td>}</td>
</tr>
<tr>
<td>&lt;style&gt;</td>
<td>/* internal styles override site styles - just for this page. */</td>
</tr>
<tr>
<td>body {</td>
<td>&lt;/style&gt;</td>
</tr>
<tr>
<td>background-color:pink;</td>
<td>&lt;/head&gt;</td>
</tr>
<tr>
<td>color:green;</td>
<td>&lt;body&gt;</td>
</tr>
<tr>
<td>}</td>
<td>Hello World</td>
</tr>
<tr>
<td></td>
<td>&lt;/body&gt;</td>
</tr>
<tr>
<td></td>
<td>&lt;/html&gt;</td>
</tr>
</tbody>
</table>

That HTML page with its styles moved to an external style sheet.

<table>
<thead>
<tr>
<th><strong>index.html</strong> AFTER (references external style sheet)</th>
<th><strong>styles.css</strong> (NEW external style sheet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;!DOCTYPE html&gt;</td>
<td>body {</td>
</tr>
<tr>
<td>&lt;html&gt;</td>
<td>background-color:pink;</td>
</tr>
<tr>
<td>&lt;head&gt;</td>
<td>color:green;</td>
</tr>
<tr>
<td>&lt;title&gt;Title that shows in browser tab...&lt;/title&gt;</td>
<td>}</td>
</tr>
<tr>
<td>&lt;link href=&quot;styles.css&quot; rel=&quot;stylesheet&quot; type=&quot;text/css&quot; /&gt;</td>
<td>/* internal styles override site styles - just for this page. */</td>
</tr>
<tr>
<td>&lt;style&gt;</td>
<td>&lt;/style&gt;</td>
</tr>
<tr>
<td>/* internal styles override site styles - just for this page. */</td>
<td>&lt;/head&gt;</td>
</tr>
<tr>
<td>&lt;/title&gt;</td>
<td>&lt;body&gt;</td>
</tr>
<tr>
<td>&lt;/head&gt;</td>
<td>Hello World</td>
</tr>
<tr>
<td>&lt;/body&gt;</td>
<td>&lt;/body&gt;</td>
</tr>
<tr>
<td>&lt;/html&gt;</td>
<td>&lt;/html&gt;</td>
</tr>
</tbody>
</table>

After making these changes, save all your files and make sure index.html still renders properly (you can still do NetBeans "View"). Once you have moved styles to an external style sheet, you may find that you have to clear your browser cache to view any recent making style sheet changes. In Chrome, this can be done by selecting "More Tools – Clear Browsing Data" from the menu. If you suspect that browser caching is preventing you from debugging, make an obvious style sheet change, refresh the browser and see if you see it.