

# 3344 Lab: HTML Home Page

## Table of Contents

1.	LAB OVERVIEW .....	1
2.	BEFORE YOU START.....	2
3.	SELECT WEB SITE TOPIC.....	2
4.	LEARN ABOUT WEB DESIGN .....	3
5.	REQUIREMENTS FOR YOUR WEB SITE .....	4
6.	GRADING AND SUBMISSION .....	6
7.	SUGGESTED APPROACH .....	8
APPENDIX: HOW TO DEBUG HTML AND CSS.....		10
APPENDIX: HOW TO MOVE AN INTERNAL STYLE SHEET TO AN EXTERNAL STYLE SHEET .....		12
APPENDIX: HOW TO USE ANGULAR "NG-INCLUDE" DIRECTIVE TO REFERENCE COMMON HTML CODE .....		13

## 1. Lab Overview

In this lab you will:

- **Install the NetBeans bundle.** *Later in the semester, all students will need to use NetBeans (to write JSP Web APIs), so it is best to familiarize yourself with using NetBeans in the earlier/easier labs. Even for this lab, you need to use the editor as well as the built-in web server that is provided by NetBeans.*
- **Learn basic web design**, including basic HTML elements, CSS properties, web color codes, and how to create some simple layouts.
- **Select a web site topic**, an image or two, and a color scheme that is appropriate for your topic.
- **Create a home page** with an internal style sheet (that meets the requirements outlined here).
- **Create a three page web site** that employs **CSS and HTML code reuse**.
- 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 blackboard.

## 2. Before You Start

- If you don't already have the **Netbeans Bundle** installed at home, do that first (instructions are in a separate document). Even if you already have NetBeans installed, you may have to reinstall it, to get the latest version and to be sure you have the NetBeans distribution that is bundled with apache (web server software) and glassfish (JSP server software).
  - If you have a laptop, it would be very good to bring it to your first lab (less work, no need to move your project between development PCs/Macs).
- Using NetBeans, **create a Web Application Project that you will enhance throughout the semester**. Name the project what you like, but include your name in the project name (helps us when grading). Then test your ability to add, view, run, **modify a HTML page, and then run the page to see your changes**.
  - If you want to get some sample code to play with, you can right click one of the sample layout pages, View Source, then copy/paste the code into a HTML file within the NetBeans editor.
  - **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 Viewing Source from Firefox (errors colored red).
- 1. **Learn how to publish a page**. We will probably do this in your first lab – just to be sure everyone knows how to publish and that there are no technical problems (especially for students who added late and need to have web root folders created for them on cis-linux2). Publishing instructions are described in a separate document.
- 2. Learn where to **find course labs and related tutorials**: google “sallyk temple” and click on your course number (CIS 3344) under “teaching”. Each lab is listed there, along with links to any related tutorial material.
- 3. **If you are new to web design**,
  - study the “Web Design” links that are referenced from this lab 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 (increasingly complex) sample layouts that are referenced from the Web Design Tutorial page. You will be quizzed on the concepts used in these pages.

## 3. Select Web Site Topic

In this lab, you will select a web site topic which you will keep for the whole semester. Your topic can be anything, but try to pick something that some company might pay you to write. This will make your site look "realistic" and more professional and this improve your lab and project grades.

To help you select a topic, you might want to know a little about how future labs will extend your website. Later on this semester, you will create a database table (that meets the requirements specified below) and your web site will allow users to add records into that database table. As you can see, the database table requirements really do not limit your choice of web topic. I recommend that you first think about what database table you would like to create, then come up with a web site topic related to that table.

- id (primary key, auto-increment),
- a (unique) descriptor,
- a field that will hold a long URL to an image,
- at least one non-character field (date, integer, or decimal data type), and
- at least one more field.

## 4. Learn About Web Design

If you have no web design experience, follow the links from this lab. Study that material and experiment with it. If you have web experience, follow the links – to be sure you already know the information presented. These are the concepts for which you are responsible:

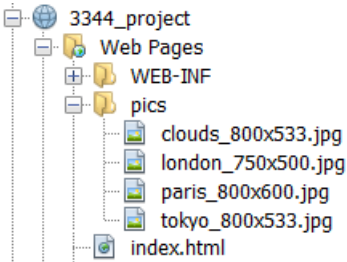
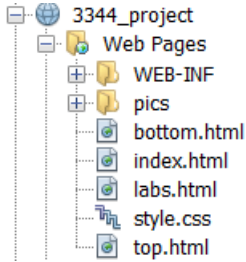
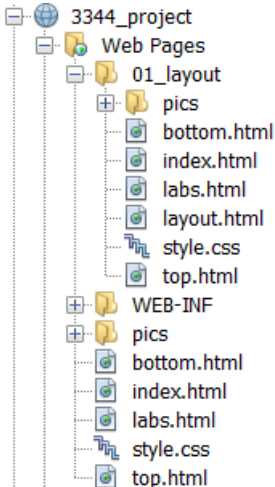
- All of the HTML and CSS concepts that are listed on my "Web Design: Getting Started with HTML and CSS" page - excluding "HTML Forms" and "Advanced CSS" (which we will learn later).
- Learn how to take advantage of the W3Schools "Try It Yourself" pages which are so helpful for people who are trying to learn the basics about web design/development.
- Be sure that you understand CSS float. An element that is floated affects not only the floated element, but elements next to it. 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 stop unwanted float behavior.
- 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. The fixed element might be behind or in front of other elements. To control which element is "on top", you use the CSS "z-index" property.
- 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

In this lab, you will create a web site that includes 3 html pages and some auxiliary pages. More specifically, your web site shall include:

- a home page (index.html),
- a labs page (labs.html),
- a folder (e.g., "pics") that holds image files,
- an external style sheet, and
- two reusable "html snippet" files (like top.html and bottom.html).
- a backup folder (e.g., named "01\_layout") that contains a backup copy of all of the above files.

These screen shots (taken from NetBeans project pane), shows how your project files might be organized as you work through this lab:

1. Laying out your home page using internal style sheet - every student will find their own image file(s).	2. After employing reuse Styles are moved to "styles.css", reusable HTML code is moved to files like "top.html" & "bottom.html", "labs.html" is created as a copy of "index.html".	3. After backing up this lab We back up this lab (so it continues to work in your project) so that every student is free to change the look and feel of their home page (after lab1 is graded).
		

*Your home page lab shall **NOT** use Bootstrap* (at least, not until after lab 1 is graded). Bootstrap is a great tool, but it obfuscates many of the concepts we are trying to learn here.

1. Your **index.html** shall

- include HTML elements with the **following attributes**: id="title", id="nav" (the navigation bar), class="navLink" (2 or more of these), id="content", id="footer". You can have more elements, of course. I ask for specific id attributes to ensure that students can't submit a page from the internet AND to facilitate grading.
- 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 **nav bar** with two relative links: "home" (references [index.html](#)) and "labs" (references [labs.html](#)).
- have a **content** area that
  - describes your website and entices viewers to visit the site and looks professional.
  - has at least one external link (href= "<http://...>") with different styling than your navbar links.
- 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 **styling** shall:

- 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
  - are not underlined (CSS declaration "text-decoration:none").
  - 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).
  - always have colors that stand out visually against their background: before clicking (:link), after clicking (:visited), and while hovering (:hover),
  - 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.
- employ at least **two advanced CSS techniques**, such as, gradients, rounded corners, box shadow, text shadow, transparency, or transitions.
- 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. Your **index.html** shall
  - reference an **external style sheet** (e.g., "styles.css")
  - use the **Angular "ng-include" directive** to reference common "HTML code snippets", e.g., one for your title/nav and one for your footer.
4. Your web site shall include a file named **labs.html** that is an exact copy of index.html except its content will consist of a heading "Lab 1 – Home Page" and three paragraphs:
  - a) Describe your experience with HTML, CSS, javascript/jquery and/or any client side frameworks.
  - b) Explain what advanced CSS techniques you employed in your style sheet.
  - c) Blog about what you learned in this lab (everyone should have learned something new).
  - d) Links to 01\_layout/index.html (see next item).

**Note:** each week, as you complete each lab, you will add another "blog entry" to your labs page – a header with lab number and lab title, followed by at least one paragraph that describes what you learned and links to the work you did in that lab.

5. All 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**.
    - 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).
6. Your web site shall include a **backup folder named "01\_layout"** that has a copy of the final version of all the elements listed above. This folder will contain your lab1 work even though you may decide later to change your layout in ways that no longer meet the lab1 requirements. You may change your layout at any time (after lab1 is graded), as long as
  - index.html has the same look/feel as labs.html and
  - Your blog from labs.html links to **01\_layout/index.html** (that still works and stills meets the requirements of this lab).

## 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 lab. Your layout does not have to be fancy and fantastic. It can be simple, as long as it meets the requirements and looks like the home page of a recently updated commercial web site.

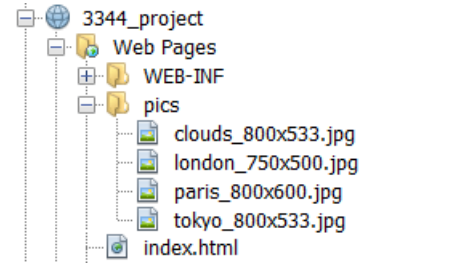
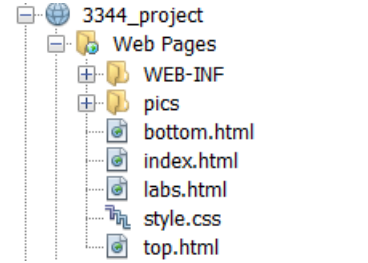
When you have completed all the requirements, tested locally (and syntax checked), and published, don't forget to submit a zip file of your web site into blackboard. (No RARs please.)

### Here is how we evaluate your lab:

1. We check that you have submitted a **zip file** into blackboard (by the due date) then visit your **published** pages.
2. We check that your home page contains one or more **images** (image tag or background image).
  - We check that your home page does not load slowly, that it contains no jpg images which render slowly left to right, that it contains no png images which render slowly (blurry then crisp due to png interlacing).
  - To test this for yourself before submitting your lab, clear your browser's cache, then reload your index.html page.
3. We look at your home page and subjectively evaluate its realism, its **professionalism**, and its **originality**.
  - 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?
  - Is your name in the footer?
  - 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?
  - Is your layout substantially different from sample layouts that were provided (and from your classmates)?
4. We check for **fluidity** by slowly narrowing the browser until it cannot be narrowed any more.
  - 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 than the screen you used to develop your layout.
5. We test **nav bar links** (index.html  $\leftarrow$   $\rightarrow$  labs.html).
  - Does labs.html have the same layout as index.html, without any "jumping around" of elements, as we link back and forth between index.html and labs.html using your nav bar?
6. We look at your **labs page**.
  - Does your labs page contain the blog about this lab? Does that blog answer the questions posed in this lab (your web experience, the advanced CSS techniques you used, what you learned in this lab)?
  - Does your blog link to 01\_layout/index.html and does that page render properly?
7. We "**View Source**" on your **home page** from Firefox and make sure that your HTML code:
  - includes the ids and classes specified in this lab ("title", "nav", "navLink", "content", & "footer").
  - references an external style sheet and uses Angular to reference reusable HTML code (like "top.html").
  - has no (red) syntax errors and is properly indented (e.g., using NetBeans "Source – Format").
  - contains no irrelevant code and is neat and organized.
8. From "**View Source**" of your home page, we click on your **external style sheet** link and check that:
  - you have used the CSS float property somewhere.
  - you have implemented a fixed footer – with some padding or margin so that the bottom content is never covered up by the footer (e.g., when/if your content would be long enough to need vertical scroll bar).
  - you have no syntax errors and is properly indented (using NetBeans "Source – Format").
  - contains no irrelevant code and is neat and organized.
9. We type in the URLs to your **reusable HTML code** (top.html, bottom.html) to ensure that these are HTML "snippets" (partial code) - not complete HTML pages that start with <html> and end with </html>.

## 7. Suggested Approach

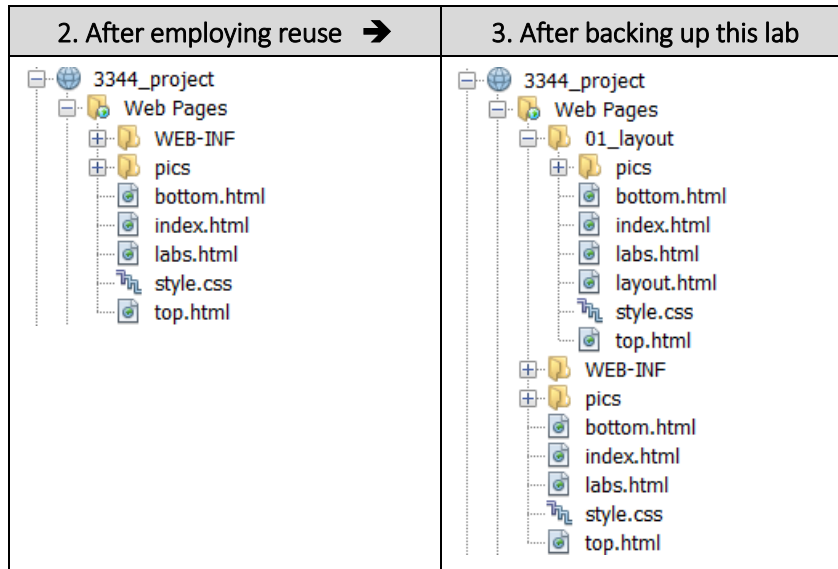
1. Carefully **read the first four sections** of this document. If you have little or no HTML/CSS experience, it is imperative that you follow the links and learn about web design. **Study the sample layouts** (referenced by the Web Design Tutorial page). You can create your layout from scratch or select one of the samples to use as a starting point for your lab. 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 labs.
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. 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. Next, employ the **code reuse** techniques that are explained in the appendices of this document.
  - Move the style rules from your internal style sheet (of index.html) to an external style sheet (styles.css) and reference the style sheet from index.html.
  - Move the "HTML code snippets" to their own files (e.g., title/nav and footer) and reference these using the Angular ng-include directive. Once you do this, you can no longer use NetBeans "View" to test your code; you have to use the "Run" option.

1. Laying out your home page →	2. After employing reuse
	

5. **Copy index.html to labs.html** and then modify the content of the labs.html (as listed in the requirements section – add a blog about lab1).
6. **Test both pages locally** (index.html and labs.html) using NetBeans (right click and "run" option). Make sure that all links work and that the layout does not "jump" from page to page (due to a change you made after copying).



7. **Create backup folder** (folder "01\_layout") which you can do right from the NetBeans project pane. Test your link from `labs.html` to `01_layout/index.html`



8. **Publish** your website to the web server, by following the instructions that are provided in a separate document (link provided from this lab of the 3344 labs page).
9. **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.
10. **Submit** your homework by attaching a zip file of your web application into the lab 1 assignment in Blackboard.

## Appendix: 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 lab. 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 lab 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**:



The diagram shows a CSS rule: `h1 {color:blue; font-size:12px;}`. Above the rule, labels indicate the parts: "Selector" points to `h1`, and "Declaration" points to the curly braces. Inside the braces, "Property" points to `color` and `font-size`, while "Value" points to `blue` and `12px`.

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,
  - 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).
  - 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 lab 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 lab 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 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: 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.

index.html BEFORE (style in the head)
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;title&gt;Title that shows in browser tab...&lt;/title&gt;     &lt;style&gt;       body {         background-color:pink;         color:green;       }     &lt;/style&gt;   &lt;/head&gt;   &lt;body&gt;     Hello World   &lt;/body&gt; &lt;/html&gt;</pre>

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

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

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.

## Appendix: How to Use Angular "ng-include" Directive to Reference Common HTML Code

Using "ng-include" allows us to reference reusable "HTML code snippets". If we have all pages reference a HTML snippet that contains nav bar code, we will only have to change one file to change the nav bar for the whole site.

index.html BEFORE using <i>ng-include</i> (highlighted code will be moved and referenced)	
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;title&gt;Layout using Angular Include Directives&lt;/title&gt;     &lt;link href="styles.css" rel="stylesheet" type="text/css" /&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;div id="container"&gt;       &lt;div id="titleNav"&gt;         &lt;div id="pageTitle"&gt;           &lt;a href="index.html"&gt;City Gallery&lt;/a&gt;         &lt;/div&gt;         &lt;div id="nav"&gt;           &lt;a href="london.html"&gt;London&lt;/a&gt;&lt;a href="paris.html"&gt;Paris&lt;/a&gt;&lt;a href="tokyo.html"&gt;Tokyo&lt;/a&gt;         &lt;/div&gt;         &lt;span class="stopFloat"&gt;&lt;/span&gt;       &lt;/div&gt; &lt;!-- titleNav --&gt;       &lt;div id="content"&gt; ...     &lt;/div&gt; &lt;!-- content --&gt;     &lt;div id="footer"&gt;       Copyright W3Schools.com     &lt;/div&gt;   &lt;/div&gt; &lt;!-- container --&gt; &lt;/body&gt; &lt;/html&gt;</pre>	
index.html AFTER using <i>ng-include</i>	
<pre>&lt;!DOCTYPE html&gt; &lt;html&gt;   &lt;head&gt;     &lt;title&gt;Layout using Angular Include Directives&lt;/title&gt;     &lt;link href="styles.css" rel="stylesheet" type="text/css" /&gt;     &lt;!-- Reference angular js library, so we can use the ng-include directive --&gt;     &lt;script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.4.8/angular.min.js"&gt;&lt;/script&gt;   &lt;/head&gt;   &lt;body&gt;     &lt;div ng-app=""&gt; &lt;!-- inside this div you can use Angular directives --&gt;       &lt;div id="container"&gt;         &lt;div ng-include src="top.html"&gt;&lt;/div&gt; &lt;!-- will insert "top.html" code here --&gt;         &lt;div id="content"&gt;           ...         &lt;/div&gt; &lt;!-- content --&gt;         &lt;div ng-include src="bottom.html"&gt;&lt;/div&gt; &lt;!-- inserts "bottom.html" code --&gt;       &lt;/div&gt; &lt;!-- container --&gt;     &lt;/div&gt; &lt;!-- ng-app --&gt;   &lt;/body&gt; &lt;/html&gt;</pre>	<div><div>top.html</div><pre>&lt;div id="titleNav"&gt;   &lt;div id="pageTitle"&gt;     &lt;a href="index.html"&gt;City     Gallery&lt;/a&gt;   &lt;/div&gt;   &lt;div id="nav"&gt;     &lt;a     href="london.html"&gt;London&lt;/a&gt;     &lt;a href="paris.html"&gt;Paris&lt;/a&gt;     &lt;a href="tokyo.html"&gt;Tokyo&lt;/a&gt;   &lt;/div&gt;   &lt;span class="stopFloat"&gt;&lt;/span&gt; &lt;/div&gt; &lt;!-- titleNav --&gt;</pre><div>bottom.html</div><pre>&lt;div id="footer"&gt;   Copyright W3Schools.com &lt;/div&gt;</pre></div>

After moving the code, RUN your page and see if the title/nav and footer rendered properly.