Week 5/6 – Website Design and Hosting

Lab Assignment – 100 pts

Due 11:59 pm November 4, 2019

Objectives:

1. To learn about how the components of a webpage, namely HTML, CSS, and JavaScript, combine to make a functioning site. (Week 5)
2. To learn about serving dynamic webpages and interacting with databases. (Week 6)

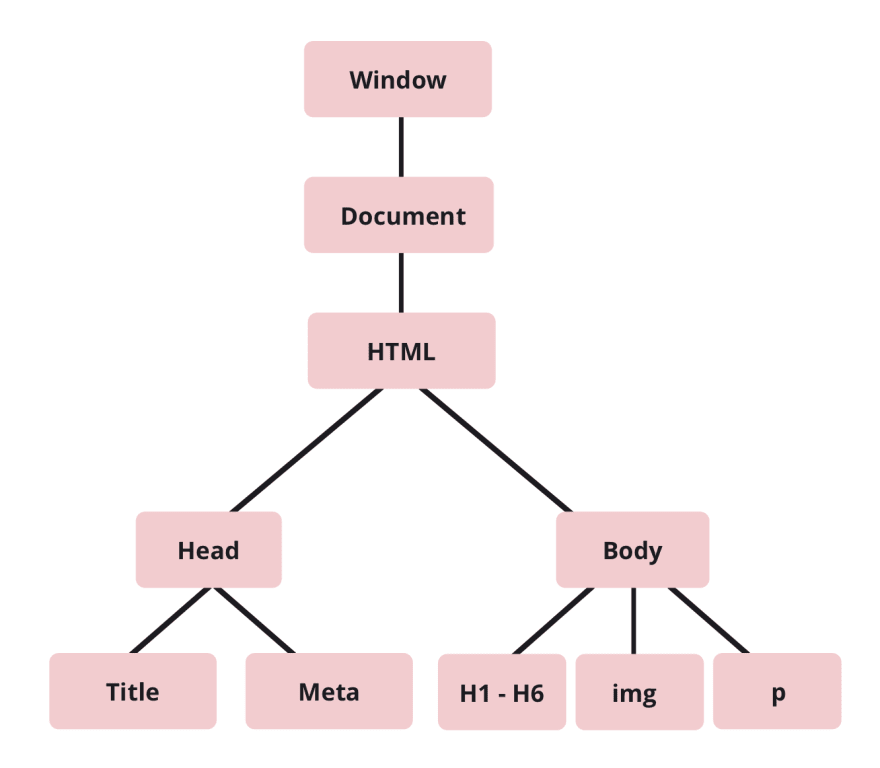
Procedure:

You will be working in teams of 2 for this lab. One partner will be the front end (FE) developer, and the other will be the back end (BE) developer. In week 5, the FE developer will interact with the website while the BE developer will work on the answer sheet. In week 6 the responsibilities will flip, with the BE developer working on the website, and the FE develop filling out the second half of the answer sheet.

Since both people will work on both parts, it would make sense to put the answer sheet in your group’s git repository. This ensures that everyone will always have the most up-to-date version.

Before we begin, it should be noted that this lab has some required programs to run it properly. We recommend that you use remote desktop to Drexel’s Unix server (Tux). The advantage here is that Tux already has everything you need installed and ready to go. Many of you have already linked it to your GitLab accounts, which also makes life easier. If you want to try it locally, you’ll need to install [**Node**](https://nodejs.org/en/download/), and [**SQLite**](https://www.sqlite.org/download.html)**.** You will also need local ssh authorization for your GitLab. You will of course also need Git installed. At the end of the day it doesn’t matter as we will only be looking at the answer sheets and your git repositories on GitLab.

HTML, CSS, and JavaScript (Week 5)

1. Remote Desktop
   1. If you chose to go with the remote desktop option, you will access tux through a remote desktop application (instead of Putty or ssh). On Windows the application comes pre-installed, but on Mac you have to download it. Search for **Microsoft Remote Desktop** on the Mac OS app store and it should come up as the first option.
   2. Once it’s open, you will be prompted for a computer name. Enter **tux2.cci.drexel.edu**.
   3. You will then be prompted to log on, this is just your normal tux logon username/password. (Note, it can be glitchy and prompt you to enter your password multiple times. This does not mean you messed it up. After a few tries it will let you in).
   4. You should now see a desktop not too unlike your own.
   5. If you right click on the screen, a menu will open. One of the options is to open the terminal. If you select that option, you will get a terminal just like PuTTY or ssh. The advantage of the remote desktop is that you also have a visualization of the file system and can open a browser to look at / interact with your website.
   6. Lastly, if you have persistent issues with Tux that TAs were not able to solve, email **ihelp@drexel.edu**.
2. Create Your Group
   1. The FE will create a group in GitLab that includes the BE, the lab professor and the TA’s as developers.
3. Clone the Repository
   1. Using the terminal, the FE will clone the repository at <https://gitlab.cci.drexel.edu/ny66/CI101Lab5> into your tux directory.
   2. Next, you have to disconnect the cloned repository from the original. Use the command **git remote rm origin**. Also delete the .git folder.
   3. Now in GitLab make a new repository, but don’t put anything in it.
   4. Back in the terminal, in the directory of the folder, type **git init**. This tells Git that the folder will be used for a git project.
   5. Next you will stage the files (**git add .**), and commit them (**git commit -m “**first commit message”.
   6. Copy the ssh or http url from your new repository. In the terminal type **git remote add origin <<insert the url here>>.** If you would like to check that it worked, use **git remote -v**. You should see the new url under origin.
   7. The last step is to push your changes using **git push origin master**
   8. You will also want to go into the package.json file and change the git url to your new repository.
4. The Project Structure
   1. This project contains a lot of files, seemingly too many. While they are all important, most won’t be touched in the next two labs. The important ones for our purposes are:
      1. Index.html – This is where all of the HTML goes. The HTML is the skeleton of the website, keeping everything organized and in place. To open the file for editing, right click and choose open with notepad++ (or another editor). If you just click on it, it will try to open the rendered version of the page in the browser, which is not what we want.
      2. Index.js – This is one of the two JavaScript files in the project. This file is used for storing JavaScript that will help with website functionality.
      3. Stylesheet.css – This is where the CSS goes, the CSS controls all of the stylings of the website. Things such as color, font, and relative positioning all fall under this category.
      4. The other two files to note are server.js, and CLCBooksSQL.sql, but they only come up in week 6.
5. HTML
   1. As mentioned earlier, html is the structure of the website. It’s all of the content, but without positioning or styling. If you had just html on your webpage, it would be very ugly. HTML is written in what we call **tags**. A tag is a keyword enclosed by < and >. Most tags need to be closed after the content that goes into them is finished. This is done using </ the tag name again >. Different tags have different behaviors. Some of the most common ones are:
      1. <div> – the basic box tag. It purposefully has very little behavior of its own, as it’s manipulated to do what the user wants. Can contain text, an image, or even other tag elements within it.
      2. <p> – the paragraph tag. This one is used for text. You can think of it almost like a textbox.
      3. <h1> – this tag is in a group with 5 others (h1, h2,…, h6), but they all do basically the same thing. They are headers, or titles. If you need big text for a title, you can use one of them. They are in decreasing order of size (h1 is the biggest), but all are bigger or at least bolder than normal text.
      4. <img src=”#”/> – this tag is different from the other ones on the list in two major ways, firstly it’s self-closing (meaning that it doesn’t need an </img> tag to close it), and secondly it has an attribute. In this case the attribute is src, which stands for source. In between the quotes, you put the address of an image, whether it’s a URL or file path. The tag knows to go to that source and pull the image. Similar tags are <link> (which uses href instead), and <script>, which loads a JavaScript into the HTML page.
      5. <html>, <head>, and <body> – These three tags are also special in that they show up in every html file. The <html> tag wraps the entire page, and the <head> and <body> tags separate the content. The <head> tag contains information about the page such as the character set in use and loads dependencies. The <body> tag is used for the actual content. Everything you see on a webpage is in the <body> tag. Along with these three tags, you must also always include <!DOCTYPE html> as the first thing on the page. Together they keep the structure of HTML uniform across developers, browsers, and generations.
   2. The combination of opening tag, content, and closing tag is often referred to as an element. These elements comprise what is called **The DOM** or **D**ocument **O**bject **M**odel. The browser keeps track of the hierarchy of elements in the DOM. Since elements can be inside of each other, it’s important to keep track of who is who’s parent. Languages like JavaScript use the DOM to interact with html elements. Unfortunately, with the DOM you can only see inward. From a given element, you can find it’s children, but not its parent.
   3. 
   4. There are also comments in html. Comments are denoted by <!-- to open, and --> to close. Everything between that opening and closing gets ‘commented out’ and will no longer be shown if the webpage is loaded on a browser.
6. CSS
   1. CSS is where the styling lives. If you want an element be in a different position, or have a different color, you have to use CSS.
   2. In CSS you target elements based on their tag, id, and or class. While the tag is implicitly defined when you add an element, ids and classes are added by the programmer as attributes. If you want to add an id or class, you simply include it <div id=”X” > or <p class=”Y”>
      1. Id – must be unique within the DOM, so only one item can have a specific id. This makes them useful if you want to target a specific item of many.
      2. Class – they are basically the opposite of ids. If you target a class in CSS what you add will affect all of elements that have that class.
   3. The general format of a CSS rule is the name of the element(s) you’re trying to target, opening curly brackets, the properties you wish to change for that element, and closing brackets.
   4. 
7. JavaScript
   1. JavaScript is the action center of web development. Just HTML and CSS gives a purely static website. It might look nice, but it would have little to no functionality. CSS can do some animation and movement, but nothing like JavaScript. HTML and CSS are not actually programming languages, and you can’t use them for logic. JavaScript comes in to solve these problems in a big way. By having access to the DOM (through document.getElementById(“id-X”)), JavaScript brings websites ‘to life’.
   2. Since this is just an introductory lab, we are not going to dive too deep into JavaScript, instead showing one example of how to attach it to an already existing element. Some tags, like button, have built in interfaces with JavaScript. Button for example has the onclick attribute which lets it perform an action when a user clicks it. This attribute can take a JavaScript function as an argument, calling the function when it’s clicked.
   3. A lot of people start off with what is called inline JavaScript. You can technically write your code inside of the HTML files as long as you use <script> tags. While this might be easier for a beginner to understand, it’s a really bad habit. It’s very easy for files to get large and unruly, and everything generally works better when it’s split up into parts.

What to Turn in:

Turn in your completed Lab Answer Sheet. Only one answer sheet needs to be submitted to Blackboard for the whole team. Make sure all team members names are on the sheet

How you will be graded:

Your answer sheet must be submitted to Blackboard on time for full credit to be possible

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | Developing – 0 to | Proficient – ¾ | Excellent - max | Points |
|  |  | ½ |  |  |  |
| Content | |  |  |  |  |
| 1. |  |  |  |  | /20 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 2. |  |  |  |  | /10 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 3. |  |  |  |  | /10 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| Expression | |  |  |  |  |
| 4. |  |  |  |  | /5 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| 5. |  |  |  |  | /5 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| TOTAL |  |  |  |  | / 50 |