Section Notes
CS75
Week 1

PART I. Using XAMPP in Conjunction with SFTP

Once you've installed XAMPP (http://www.keitr.com/xampp), you can develop, debug, and run web scripts on your own computer. Once your XHTML is looking and your code is working how you'd like, you can then transfer the files to your domain on CS75.net by way of SFTP.

SFTP: Secure File Transfer Protocol

Using the Command Line:

-Open a terminal
-Navigate to the folder in which your local files are stored
-Type "sftp yourusername@yourdomain.tld"
-You will be asked to authenticate
-You will arrive at an 'sftp' prompt
-Type put filename.ext path/newfilename.ext

Example:

-I opened a terminal
-I typed "cd desktop"
-I typed "sftp human@robot.org"
-I typed my password
-At the sftp> prompt, I typed "put notes.XHTML public_XHTML/lamenotes.XHTML"
-I finished.

Using a Client: The easiest way to transfer files with SFTP is by way of an SFTP Client, a program with a graphical user interface (GUI) that does the dirty work for you.

For the Mac OS, Fugu is one popular SFTP client.

http://rsug.itd.umich.edu/software/fugu/files/Fugu-1.2.0-English.dmg

For Windows, we recommend SecureFX or PSFTP:

https://secure.vandyke.com/cgi-bin/download_form.cgi?PRODUCT=SecureFX http://the.earth.li/~sgtatham/putty/latest/x86/psftp.exe

Regardless of the client you use, the process is basically the same:

-Establish a connection and log in to your domain
-Once connected, navigate to the file you want to move from your computer to your domain
-Select the directory on the web where you want the file you're moving to ultimately live
-Transfer the file: some clients support simply dragging and dropping files from your computer to the network; others have side by side browsers that let you pick and choose file destinations. To be sure, most of these clients are meant to streamline the file uploading process, and are self explanatory--but if you encounter any insurmountable difficulties, email staff@cs75.net and we'll be happy to help you get your files online.

Additionally, Though Not Ideal, One Can Transfer Files Using Direct Admin:

-From http://panel.cs75.net, login
-Click the 'Files' icon
-Navigate to the directory in which you want the files to live
-Click the "Upload files to current directory" button on the bottom of the table
-Choose one or more files via the "Choose File" buttons
-Click "Upload"

PART II. Basic PHP

The for loop:

```
<?
for ($i=1; $i < 6; $i++ )
{
print $i . " ";
}
?>
```

This loop will iterate until the value set in the first part of the ( )s becomes something that would make the statement in the second part of the ( )s false. The third part of the ( )s designates the change that's made after each execution of the action inside the loop.

So in this case, everytime the value of $i is printed, the value of $i increases by one. So, this continues until the value of $i is no longer less than 6.

The result is the following:

```
1 2 3 4 5
```

The while loop:

```
<?
$i = 1;
while ( $i <=10 )
{
print $i . " ";
$i++;
}
?>
```

Simply put, the while loop will continue to execute indefinitely, as long as the statement in the ( )s remains true.

In this case, each time the body of the loop resolves, $i is increased by 1. Eventually, $i will no longer be <=10. At that time, the loop will stop repeating.

The result here is:

```
1 2 3 4 5 6 7 8 9 10
```

Conditional Statements:

```
<?
$quarters = 4;
if ( $quarters == 4 )
{
echo “You have a one dollar”;
}
?>
```

Using Arrays:

Arrays are a type of data structure. Instead of storing information as many different variables, arrays allow you to store many different things under one name.

```
<?
$tfs[0]=”Dan”;
$tfs[1]=”Daniel”;
$tfs[2]=”Chris”;
$tfs[3]=”John”;
$tfs[4]=”Keito”;

echo $tfs[1];
?>
```

In this case, each index of the array stores a different TF.

When we get to the echo statement, the result would print:

Daniel

PART III. Programming for the Web

Let’s say you just filled out a form on the web, asking you for a boy’s name and a girl’s name. Part of the form looked like this:

```
     <form action="script.php" method="get">
```

When you submit the form, the URL changes to something like this:

http://www.domain.tld/script.php?boy=jack&girl=jill

Here’s how script.php can make use of the information your user just submitted:

```
<?
echo $_GET[‘boy’]." and  ".$_GET[‘girl’];
?>
```

This would print:

```
jack and jill
```

User input can be accessed in the action file (script.php) by the following syntax:

```
$_GET[‘info’];
```

Where `GET` can be replaced by other methods (POST, REQUEST, etc.) and `info` is the name of the field where the user input the information.

Now that was great, but everyone could see and access the information that was entered by way of the URL. Not particularly secure. Enter $_POST; It allows you to do the same things as $_GET, but behind a mysterious veil of secrecy.

An XHTML form invoking $_POST would have a section of code something like this:

```
     <form action="login2.php" method="post">
```

When the submit button is pressed on this form, the information the user entered is stored in the superglobal $_POST, but bypasses the URL entirely. This is good for account numbers or passwords:

Let’s say the form included both of these:

```
     <input name=”user” type=”text” />
     <input name="password" type="password" />
```

And finally, let’s say for learning’s sake, that I’m really uncool, and am thus the only one who visits my website. So, I’ve hardcoded my username and password into the login script.

```
<?
// first I set variables from the user input
$user = $_POST[‘user’];
$password = $_POST[‘password’];

// Then I compare these with the hardcoded username and
// password
if (( $user == “lonelyguy” ) && ( $password == “loveme” ))
{
echo “welcome to your page”;
}
?>
```

So even though when entering the password, you only saw ******, and then you never saw my username OR my password in the URL, PHP was still able to verify whether or not I was indeed “lonelyguy.”

PART IV. Tricks of the Trade

Even if it’s not your preferred browser, Firefox can still be very useful in the development process. Of note are two specific Firefox Extensions:

LiveHTTPHeaders:

David demoed this in class last Monday, showing us what really happens during an HTTP transaction. The information displayed here though, can give you some insight into not only how a site works but how it doesn’t work—it shows you what the server is doing, and which cookies are being passed to your browser from the site to which you’re connecting.

http://livehttpheaders.mozdev.org/installation.XHTML

Firebug:

Firebug is another extension for Mozilla’s Firefox browser that is a valuable tool in the debugging process: it enables you to debug, edit, and modify and websites XHTML, CSS, Javascript, and more on the fly. This makes error checking and testing faster and much less painful.

http://www.getfirebug.com/

And while we’ve been over it before, it bears repeating: don’t forget to make use of the XHTML validator over at http://validator.w3.org/.

PART V. Permissions

File permissions are an attribute set by an administrator that dictate who can read, write, and execute files. You can change the file permissions on your domain.

As a rule, though, files of the following types should have permissions set to the corresponding number:

.XHTML: 644
.php: 600
.gif/jpg: 644
directories: 711 (more "secure") or 755
.css: 644
.js: 644

You can set the permissions of a file in the DirectAdmin control panel:

-Navigate to http://panel.cs75.net
-Log in with your site Username and Password
-From the Home screen, click on the files icon
-Navigate these folders to where your web files are stored (public_XHTML)
-Once inside, select the file whose permissions you wish to change
-Change the number at the bottom of the table
-Click the “set Permission” button

Or you can change permissions from the command line:

-SSH to your website, and navigate to the directory containing the file
-type “chmod number file,” for example:

```
chmod 644 words.txt
```