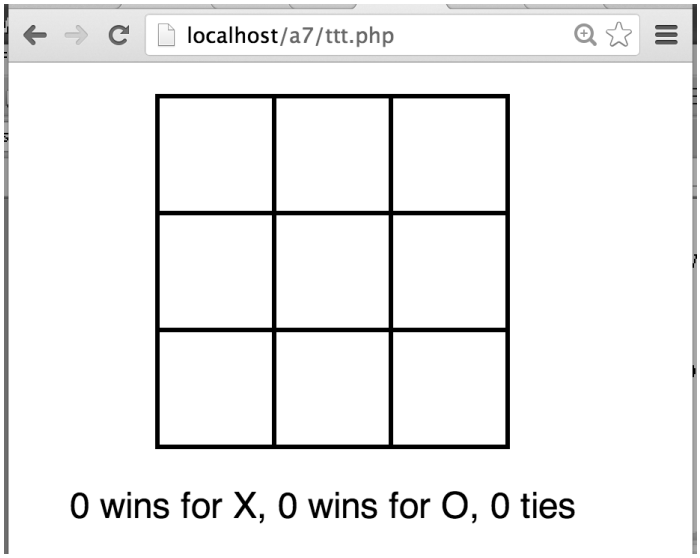
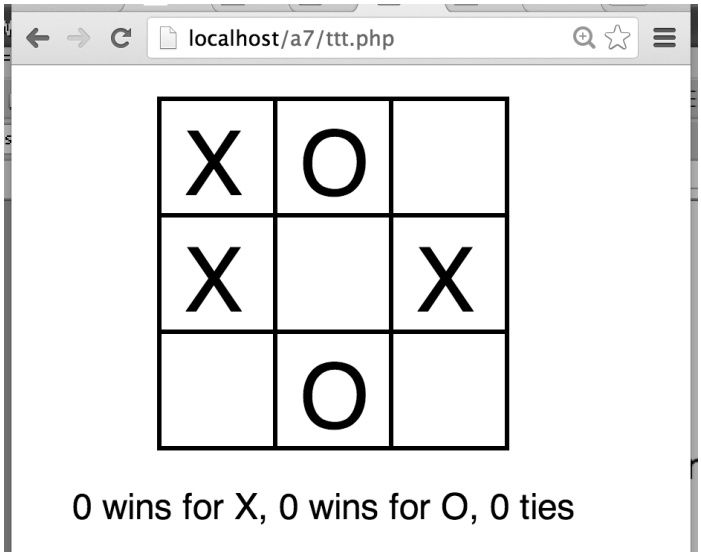
CSC 337, Fall 2013
Assignment 7
Due: Wednesday, November 13 at 22:00:00

Problem 1. (24 points) **`ttt.php`**

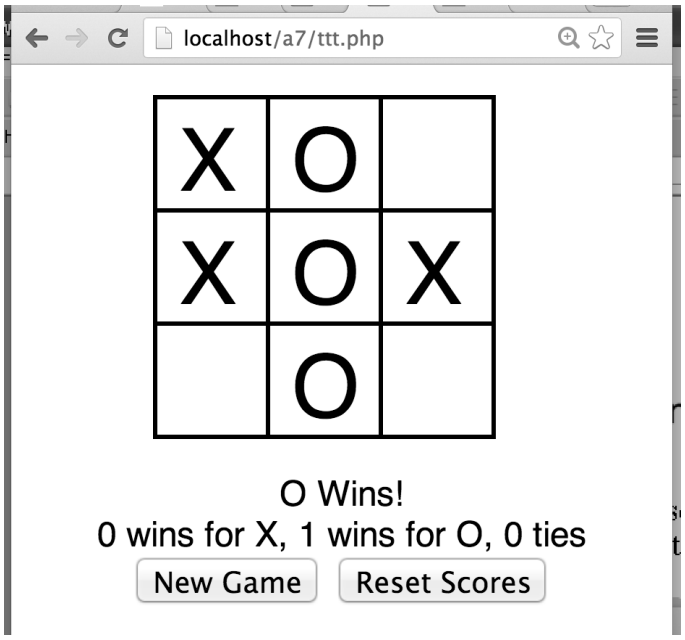
In this problem you are to create a Tic-Tac-Toe game using an HTML form. The game tracks wins and ties using a cookie. Here's what it looks like when run for the very first time.



The first click in a square places an "X", the second click places an "O", and this alternation continues. Mid-game we might see this:



If "O" then plays the center to win, we'd see this:



As you'd expect, "New Game" starts a new game. "Reset Scores" deletes the cookie and starts a new game.

I do some hacky styling to facilitate the click-based interaction. To keep this problem relatively simple you can find that styling code in `$a7/ttt-starter.php`, but if you want more of a challenge, try to work it out yourself. Also in that file is a `check_win(...)` function but again I encourage you to write that code yourself if you have time. If I had you do all that stuff yourself this problem perhaps would be worth 36 points instead of 24 points.

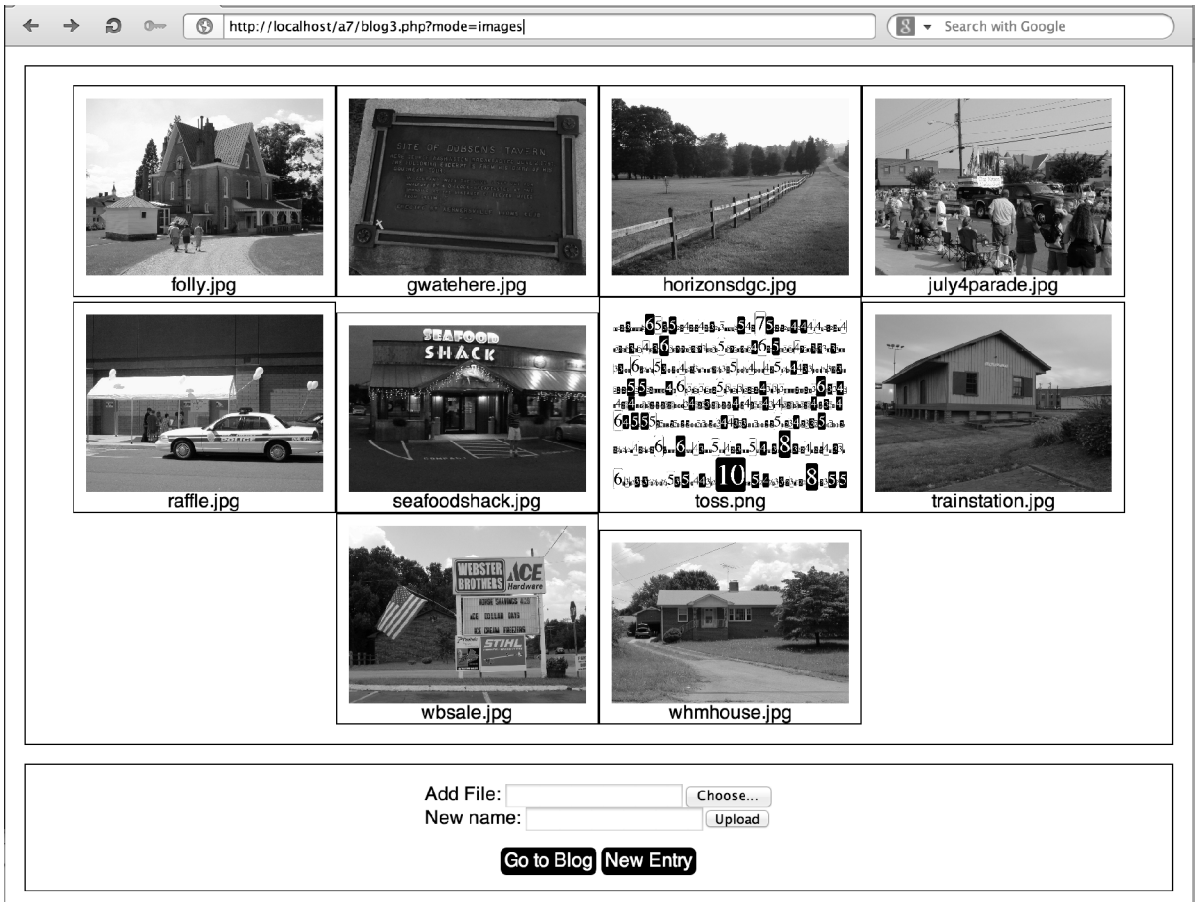
Problem 2. (15 points) `blog3.php`

In this problem you are to make two small changes to the blog:

1. Add a very simple image library and the capability to upload images. (11 points)
2. Make it so the user can easily blog about 337. (4 points)

Image library

To view the image library, hit `blog3.php?mode=images`:



Use the PHP `opendir(...)` and `readir(...)` functions to read the names of the files in the `images` directory and generate an `img` element for each with a name under it. Assume every file in `images` is an image. The above uses `div`s with `display:inline-block`, contained in a `div` with `text-align:center`. You'll notice some irregularity in heights, and you'd see more if any pictures were in portrait mode, but that's fine—I don't want you to sink much time into styling for this problem.

The form below the images allows one to upload a new image. The controls to the right of "Add File:" are generated with an `<input type=file ...>` element. The "New name:" text box below lets the user specify a new name for the uploaded file. For example, the user might want to upload `DSC07757.JPG` and call it `halloween.jpg`. If "New name:" is left blank, the original name of the file, like `DSC07757.JPG`, is used. The "Upload" button submits the form.

Slides 153-158 talk about file uploads but they don't show one thing you'll need to do for this problem: move a file from PHP's temporary upload directory to your `images` directory. Here's code for that:

```
$tmp = $_FILES["image"]["tmp_name"];
if (is_uploaded_file($tmp))
     move_uploaded_file($tmp, "images/$name");
```

The code above assumes that the `type=file input` element has `name=image`, and that `$name` has been set appropriately, depending on whether the the user specified a "New name:".
The `is_uploaded_file(...)` call provides a little bit of security—it checks to be sure that the file really was uploaded with a POST. An additional security measure you'd want to take in a real-world setting is to be sure that `$name` doesn't contain slashes, which would let an attacker place a file somewhere else. (Imagine this name: `../../../../../../../../etc/passwd`—all those dot-dots move up hopefully to the root of the file system and then down to a UNIX machine's password file. That's a little fanciful but perhaps it gives you a hint of the problems you can face.)

Use these attributes for the upload form:

```
<form enctype=multipart/form-data method=post
      action='blog3.php?mode=images'>
```

Submitting the form will cause a POST to be done but `$_GET["mode"]` will be `"images"`! If you see that, and `$_FILES` is not empty, you know you've got an upload to process! Do that first (just move the file into the `images` directory), and then display the images, which will then include the just-uploaded file.

The white-on-black "Go to Blog" and "New Entry" elements are simply styled anchors that reference `blog3.php` and `blog3.php?mode=new`, respectively.

Change your `.li`/`.ri` processing code to generate `img` elements with `src='images/...'`, to reference the image files therein. `$a7/images` has our stock set.

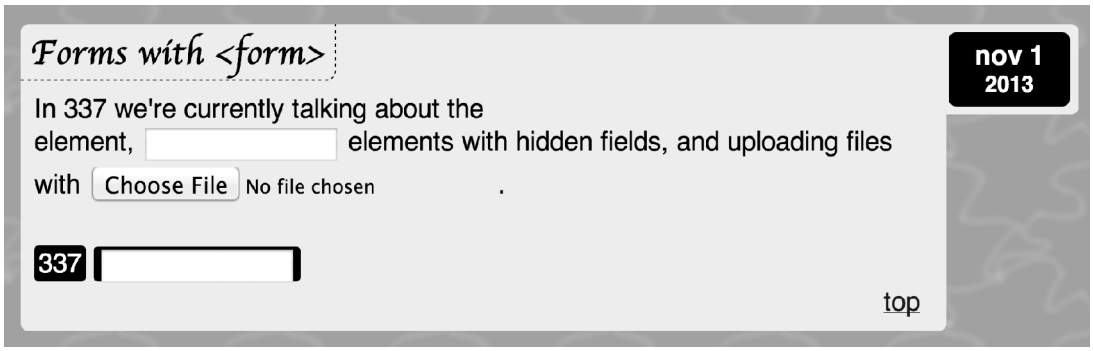
Blogging about 337

I made a mess in class on November 4th when I blended in `htmlspecialchars(...)` with my `upload2.php` example. I apologize! This problem gives you some first-hand experience with `htmlspecialchars(...)`.

Imagine a blog entry with these lines in `blog.txt`:

```
Forms with <form>
tags: 337, <input>
2013-11-01
In 337 we're currently talking about the <form> element,
<input> elements with hidden fields, and uploading
files with <input type=file>.
.end
```

Sadly, with my `blog2.php` it renders like this:



Also, in the My Posts box I see just "Forms with" for the title:

The problem is that the entry contains text that looks like markup. The browser dutifully renders that text, producing input controls where the text contained "`<input>`".

We could require the user to employ character references, like "`&lt;`" instead of < but that'd be pretty silly. Instead we can use PHP's `htmlspecialchars(...)` function to turn HTML special characters into character references. Here's an example:

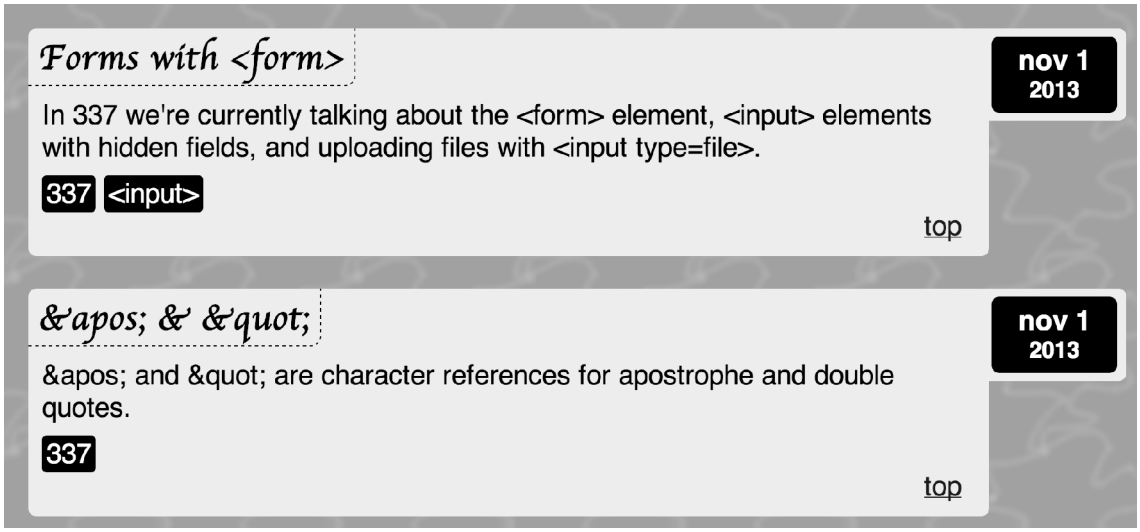
```
php > $s = "a line with < and &";
php > echo htmlspecialchars($s);
a line with &lt; and &amp;
php > echo htmlspecialchars_decode(htmlspecialchars($s));
a line with < and &
```

Note that `htmlspecialchars_decode` is an inverse function for `htmlspecialchars`.

Your task for this portion of the problem is to enable your `blog3.php` to handle entries that contain markup, like the above. For example, these two entries:

```
Forms with <form>
tags: 337, <input>
2013-11-01
In 337 we're currently talking about the <form> element,
<input> elements with hidden fields, and uploading
files with <input type=file>.
.end
&apos; & &quot;
tags: 337
2013-11-01
&apos; and &quot; are character references for apostrophe and
double quotes.
.end
```

must render like this:



We'd see this in My Posts:

You might see some Whack-a-Mole with this—a change that fixes the body of the entries might cause "Forms with <form>" to turn into "Forms with <form>", for example. Be sure to avoid that. And be sure to preserve the hopping behavior of characters in the title.

I made changes in both `blog3.php` and `load_entries.php`, but you may decide to approach it differently.

Note that you are not allowed to change the text in `blog.txt`—your soution must work with `$a7/blog.txt` as-is.

Details

My assignment 6 blog solution is in `$a7/whm-{blog2,load_entries}.php`. Feel free to start with it or use pieces of it.

Problem 3. (10 Project Points) `project2.html`

This assignment is a relatively simple one because I want you to spend some of your 337 time this next week putting some time into your project.

For the 10 project points of this problem I'd like you to have a definite web application project in mind and also sketch out in a little detail the features of your app. You might use English, you might draw some storyboards, or draw some interface flow diagrams, or do some combination of those and other things. The

purpose of whatever you sketch out is to give me enough detail so that I can make a guess as to whether what you've got in mind is enough to constitute a project worth the full amount of points.

I'm sure that class-wide there will be lots of changes of direction, both major and minor. You're not chiseling promises into stone; you're just putting forth an initial vision.

If you don't yet have any ideas on what to do, then I want to meet with you for maybe 30 minutes and see if we can figure out a project for you. If you fall in this category then I want to know all the times during the week you'd generally be available for a meeting. I'm not available from 11:30-12:30, and 1:30-4:00 MWF but other than that I can generally schedule appointments anywhere in the range 8:30am-7:00pm Monday through Friday.

Problem 4. Extra Credit `observations.html`

Submit a styled-as-you-like HTML file named `observations.html` with...

(a) (1 point extra credit) An estimate of how long it took you to complete this assignment. To facilitate programmatic extraction of the hours from all submissions, please enclose the number of hours in a `span` with `class=hours`, like this:

```
I spent <span class=hours>8.5</span> hours on this assignment.
```

Other comments about the assignment are welcome, too. Was it too long, too hard, too detailed? Speak up! I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth. Think of me saying "Good!" as one point, "Interesting!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

Turning in your work

Use the D2L Dropbox named `a7` to submit a zip file named `a7.zip` that contains all your work. If you submit more than one `a7.zip`, we'll grade your final submission. Here's the full list of deliverables:

`ttt.php`
`blog3.php`
`load_entries.php` (or just incorporate that code in `blog3.php`)
`project2.html`
`observations.html` (for extra credit)

Note that all characters in the file names are lowercase.

Have all the deliverables in the uppermost level of the zip. It's ok if your zip includes other files, too.

Miscellaneous

The HTML slides, the CSS slides, PHP slides through 183, in-class examples, and a little reading on `php.net` about functions like `readdir(...)` should be all you need to complete this assignment.

Point values of problems correspond directly to assignment points in the syllabus.

`$a7` follows the convention of `$a`*N* on the previous assignments.

Remember that late assignments are not accepted and that there are no late days; but if circumstances beyond your control interfere with your work on this assignment, there may be grounds for an extension. See the syllabus for details.

My estimate is that it will take a typical CS junior from three to six hours to complete this assignment, not counting the project-related work

Keep in mind the point value of each problem; don't invest an inordinate amount of time in a problem or become incredibly frustrated before you ask for a hint or help. Remember that the purpose of the assignments is to build understanding of the course material by applying it to solve problems. If you reach the seven-hour mark, regardless of whether you have specific questions, it's probably time to touch base with us. Give us a chance to speed you up! **Our goal is that everybody gets 100% on this assignment AND gets it done in an amount of time that is reasonable for them.**

None of your results need to validate with either the HTML5 validator or the "Jigsaw" CSS validator, but you may find the CSS validator to be helpful in diagnosing mysterious problems with CSS. (The typical mysterious CSS problem is that a declaration is ineffective, like those on slide CSS slide 16.)

Remember that I award Bug Bounty points; if you find problems, there's a potential reward for reporting them. I prefer that bugs be reported by mail to me. I'll put up a " FAQs and Corrections" post on Piazza and update it as things come in.

Use the `a7` folder for any Piazza posts about the assignment.