Forms & Design

Chapters 6 & 7

- **Form** - group of controls (buttons, boxes, text fields) that allow the user to submit information to a web server.
- Server-side programs accept parameters to affect the way they run.
- Consider the following:
  https://www.google.com/search?q=oshkosh+computer+science
  - **q** is called a *query parameter*.
  - The set of all parameters is called a *query string*, for example, https://www.google.com/search?q=oshkosh&start=10, the ?q=oshkosh&start=10 is the query string.
  - Query strings have the form **name=value** where each pair is separated by a &.

We can create our own Google homepage. The *action* attribute is required.

Example

```
<form action="http://www.google.com/search">
    <div>
        Enter your Google query:
        <input name="q" />    <!-- no type = textbox -->
        <input type="submit" />
    </div>
</form>
```

- A form must have a submit button.
- When submit is clicked, a query string is built automatically.
- The name attribute is critical for each input.
- From the example below, if “computers” were entered in the textbox, hitting submit takes us to: http://www.google.com/search?q=computers
- Form attributes: type, name, value.

Example

```
<form action=“http://www.google.com/search”>
    <div>
        Enter your Google query:
        <input name=“q” />     <!– no type = textbox –>
        <input type=“submit” />
    </div>
</form>
```

- http://www.w3schools.com/tags/tag_input.asp explains the attributes for all input types.
- **Text Box**
  - <input type="text" name="name" attributes />
  - Common attributes for text boxes include: value, size, maxlength, autocomplete, autofocus, pattern, placeholder, required.
- **Text Area**
  - <textarea name="name" attributes>Initial Text</textarea>
  - http://www.w3schools.com/tags/tag_textarea.asp for textarea attributes.
  - Common attributes for textareas include: autofocus, rows, cols, wrap, required.
- HTML5 has new input types including email, number, search, tel and url.

- http://www.w3schools.com/tags/tag_input.asp explains the attributes for all input types.
- **Checkboxes**
  - <input type="checkbox" name="name" />
  - All checkboxes that are checked are submitted to the next page with a value of *on*.
  - Common attributes for checkboxes include: checked, disabled, readonly.
- **Radio Buttons**
  - <input type="radio" name="group" value="param"/>
  - The **name** attribute specifies a grouping of buttons. At most 1 of a group may be checked.
  - Omitting the value attribute causes that group's button to have a value of *on*.
  - Common attributes for radio buttons include: checked, disabled, readonly.

- Don’t you hate websites that make you click on the radio button or checkbox instead of the text associated with that option?
- **Labels**
  - <label>Control and label text</label>
  - <label><input type=“checkbox” name=“twenties” />20-29</label>
- **Drop-down Menus**
  - <select name=“name”><option>Option1</option>. . . <option>OptionN</option></select>
  - Common attributes for select include: size, multiple.
  - If choosing multiple, you essentially need to make name an array. <select name=“name[]” multiple=“multiple” . . .
  - You can also choose to have an option selected by using the selected attribute.
  - optgroup can also be used to group your options.

Example

```
<select name="classes" size="6">
    <optgroup label="Fun classes">
        <option>Web Software Dev</option>
        <option>Artificial Intelligence</option>
    </optgroup>
    <optgroup label="Extremely fun classes">
        <option>Algorithms</option>
        <option>Theory of Computation</option>
    </optgroup>
</select>
```

- Checkboxes, radio buttons and menus essentially do the same thing.
- Menus take up less space if there are many options (e.g. Countries, States).
- We'll breifly talk about interface design in chapter 7.

- **Reset**
  - <input type="reset" value="Reset Me!"/>
  - Resets the form back to it's initial state.
- **New HTML5 Input Types**
  - http://www.w3schools.com/html/html5_form_input_types.asp
  - **Color** - color from available palette.
  - **Range** - Slider for selecting values in a range.
  - **Date** - select a date.
  - **Time** - select a time.
  - Others include datetime, month, week, tel, url...
- New HTML5 elements include datalist, keygen, meter, output and progress.

- If you have a lot of input on a page, it's best to group similar things together.
- **Fieldset**
    - <fieldset><legend>Caption</legend>Controls</fieldset>
- The fieldset element will draw a box around the controls it encompasses.
- Information may need to be sent from page to page without the user having to enter it.
- <input type="hidden" name="something" value="someVal" />

- **File Upload**
  - <input type=“file” name=“param” attr/>
- In order to provide a file upload, you must change the following: <form action=“out.php” **enctype=“multipart/form-data” method=“post”**>
- HTML5 allows you to restrict the file types being uploaded. <input type=“file” name=“param” **accept=“image/jpeg”**/>

- Many forms use the same tag: input.
- Want to use **attribute selectors** to apply formatting to certain elements.
- Do not use tables to format your forms.

Example

```
input[type="text"] {
    background-color: red;
    border: 2px solid blue;
    color: white;
}
input[type="submit"] {
    background-color: blue;
    border: 2px dotted red;
    color: green;
}
```

The following were introduced in CSS3.

- **attr = "value"** - attribute exactly equals the value
- **attr ^= "value"** - attribute begins with value
- **attr $= "value"** - attribute ends with value
- **attr *= "value"** - attribute contains value

Example

```
input[name^="school"] {
    background-color: red;
    border: 2px solid blue;
    color: white;
}
input[name*="test"] {
    background-color: blue;
    border: 2px dotted red;
    color: green;
}
```

- **GET request** - used when browser is requesting information from a web server. Transmits information as query parameters in the URL.
  - Easier to understand.
  - Bookmarked and send.
- **POST request** - used when a browser is submitting new information to a web server. Embeds information into the header of an HTTP packet.
  - URL length is limited.
  - Private or sensitive information needn't show up in the URL.
  - Resending POST requests generally bring up prompts.
- <form action="someFile.php" method="post">...

- PHP has **superglobals** that are visible in all code.
- Superglobals are all uppercase and begin with an underscore.
  - **$_SERVER** - contains information about the current web server.
  - **$_POST** - contains any POST parameters.
  - **$_GET** - contains any GET parameters.
  - **$_FILES** - contains any files the user uploaded.
- Each of the globals are an **associative array**. You know this as a hash table. Array indices can be anything.
- For example, $a = array(); $a[“pi”] = 3.14;
- PHP stores query parameters into global associative arrays, e.g $_GET[“paramName”].

Form.php

Example

```
<form action="process.php" method="post">
    <div>
        Enter your name:
        <input name="name" />
        Enter your age:
        <input name="age" />
        <input type="submit" />
    </div>
</form>
```

process.php

Example

```
<?php
   $name = $_POST["name"];
   $age = $_POST["age"];
?>
<p>
   Hello <?= $name> who is <?= $age> years old!
</p>
```

Potpurri

- print_r is useful for debugging GET and POST arrays.
- foreach loop can get keys and data:
  foreach($array as $index =>$element){ . . . }
- Useful sorting functions for associative arrays include: ksort and asort.
- **array_keys** returns an array holding all of the keys.
- **array_values** returns an array holding all of the values.
- To check if a checkbox was checked, you must check if the array at that index holds a value. You want to use **isset**.
  if(isset($_POST["someName"])){ . . . }
- Echoing user-entered values is a huge security risk.
- If you are displaying values back to the user, be sure to use **htmlspecialchars**. We will cover this more in depth at the end of the semester.

- $_FILES is a 2-dimensional array with the first dimension being the name given here: <input type="file" **name="myFile"** . . .
- The second dimension can access the following items: name, type, tmp_name, error and size. For example, to print out the name, one would write: print $_FILES["myFile"]["name"];
- Uploaded files are sent to a temporary directory with a nonsensical name for obvious security reasons.
- **is_uploaded_file** returns TRUE if the file name exists. is_uploaded_file($_FILES["myFile"]["tmp_name"]
- **move_uploaded_file** moves it to a new location.

- Create a form that has several features.
- Allow the user to choose from a favorite color given a list of 3 radio buttons.
- Allow the user to choose a favorite number from a drop-down menu.
- Allow the user to enter in his or her name.
- When the user submits the answers, ensure that the user entered something in the favorite class box. If not, display an error message and link back to the form.
- Assuming all went well, display the following text in the users chosen color: "AAA's favorite color is YYY! Your favorite number is ZZZ." where YYY was the selected color, ZZZ was the selected number and AAA was the entered name.
- Along with displaying this information, save the favorite color into a text file for future use.

- Create a form that allows the user to enter a name. Let's assume the user entered a name.
- Display the following text: "Welcome back AAA!" where AAA was the entered name.
- If the user has already entered in a name and color, the text should appear in that user's favorite color.

- Let's say you download a new program and it's unintuitive and hard to use. What do you do?
- There are entire classes (CS347) devoted to design and usability. The interaction between machine and human is called **human-computer interaction**.
- design $\neq$ aesthetics. Rather, aesthetics $\subset$ design.

- Step 1 is almost always defining who the user is.
- Regardless of the user, most users will ignore instructions and lots of text.
- Functionality of your page should be immediately clear.
- You should assume the average user is much more illiterate and impatient than you think.

- Keep pages short by organizing content.
- Put important content on the top of the page.
- Use formatting to convey importanance.
- Choose fonts, colors and backgrounds carefully.
- Consistency.
- Don't be too creative.

- When programming, your first step should always be to write pseudocode. Solve the problem in English first, then write code.
- When designing an interface, work on the design first before writing code. Use **storyboards** or **flowcharts**. Create a heirarchy of pages called a **site map**.
- **Prototyping** - act of creating a rough version of a site.

- Keep your website uncluttered.
- Whitespace is king. AltaVista vs Google
- Use left-aligned or justified text.

Do not center everything.

- Choose columns and widths carefully.
- Insert “breaks” in text with either a picture, heading or table.

- Home page should contain:
  - name and/or logo of the site.
  - some information about the purpose of the site.
  - navigation links to other parts of the site.
- Your home page should be a billboard.

- Do not use frames, splash screens or strange navigation systems.
- **Horizontal menus** are rows placed at the top of a web page.
- **Vertical menus** are rows placed on either the left or right side of a web page.
- **Fly-out menu** have additional submenus.
- **Breadcrumbs** are a series of links to navigate to “higher” levels of a site.
- Links in the text should be colored, clear and easy to identify. Don’t underline text that isn’t a link.
- Visited links should have a distinct color.
- Links should be short and you should never say: click here.

- Minimize the number of fields and forms.
- Place descriptive names for fields to help auto-fillers.
- Group related fields or put them in a reasonable order.
- Use labels with checkboxes and radio buttons.
- Use JavaScript to validate the input.
- Do not force the user to re-enter *everything* if one mistake is made.
- Use default values.

- Use proper HTML elements. Use newest HTML5 controls like email, tel or date.
- These are simply guidelines, not hard rules. For a single choice:
  - checkboxes for yes/no or true/false answers.
  - radio buttons for less than 5 choices.
  - list box for 5 to 10 choices.
  - text input for more than 10 choices, preferably with autocomplete.
- These are simply guidelines, not hard rules. For a multiple choice:
  - checkboxes for finite options.
  - text boxes for short lists, text areas for long lists.

- Consider users with poor or no vision. They use a **screen reader**.
- Small fonts and icons can make a web page hard to read.
- Provide **alt** attributes for things like images and videos.
- Ensure your content reads well from top to bottom. Most browsers can easily disable CSS.
- Color-blindness affects a non-trivial amount of people. Red-green is the most common. Provide colorless feedback as well.
- If audio is used, a deaf person will need some other kind of feedback.
- Account for poor motor skills.
- Account for other input devices: televisions, tablets, phones.
- Account for slow Internet connectivity.
- **Localization** - adapting a web page to the local language/culture.