CSE 154

LECTURE 9: SUBMITTING DATA (POST)
Common UI control errors

• “I changed the form's HTML code ... but when I refresh, the page doesn't update!”
  • By default, when you refresh a page, it leaves the previous values in all form controls
  • it does this in case you were filling out a long form and needed to refresh/return to it
  • if you want it to clear out all UI controls' state and values, you must do a full refresh
    • Firefox: Shift-Ctrl-R
    • Mac: Shift-Command-R
Drop-down list: `<select>`, `<option>`

*menus of choices that collapse and expand (inline)*

```
<select name="favoritecharacter">
  <option>Jerry</option>
  <option>George</option>
  <option selected="selected">Kramer</option>
  <option>Elaine</option>
</select>
```

- option element represents each choice
- select optional attributes: disabled, multiple, size
- optional selected attribute sets which one is initially chosen
Using `<select>` for lists

```html
<select name="favoritecharacter[]" size="3" multiple="multiple">  
  <option>Jerry</option>  
  <option>George</option>  
  <option>Kramer</option>  
  <option>Elaine</option>  
  <option selected="selected">Newman</option> 
</select>
```

- optional multiple attribute allows selecting multiple items with shift- or ctrl-click
  - must declare parameter's name with [] if you allow multiple selections
- option tags can be set to be initially selected
Option groups: `<optgroup>`

```html
<select name="favoritecharacter">
  <optgroup label="Major Characters">
    <option>Jerry</option>
    <option>George</option>
    <option>Kramer</option>
    <option>Elaine</option>
  </optgroup>
  <optgroup label="Minor Characters">
    <option>Newman</option>
    <option>Susan</option>
  </optgroup>
</select>
```

• What should we do if we don't like the bold appearance of the optgroups?
Grouping input: `<fieldset>, `<legend>`

*groups of input fields with optional caption (block)*

```html
<fieldset>
  <legend>Credit cards:</legend>
  <input type="radio" name="cc" value="visa" checked="checked" /> Visa
  <input type="radio" name="cc" value="mastercard" /> MasterCard
  <input type="radio" name="cc" value="amex" /> American Express
</fieldset>
```

- **Credit cards:**
  - Visa ○ MasterCard ○ American Express

- fieldset groups related input fields, adds a border; legend supplies a caption
Styling form controls

- attribute selector: matches only elements that have a particular attribute value
- useful for controls because many share the same element (input)
Problems with submitting data

This form submits to our handy params.php tester page. The form may look correct, but when you submit it...

- [cc] => on, [startrek] => Jean-Luc Picard
The value attribute

- `<label><input type="radio" name="cc" value="visa" /> Visa</label>`
- `<label><input type="radio" name="cc" value="mastercard" /> MasterCard</label>`
- `<select name="startrek">
  <option value="kirk">James T. Kirk</option>
  <option value="picard">Jean-Luc Picard</option>
</select>`

- `Value attribute sets what will be submitted if a control is selected`
- `[cc] => visa, [startrek] => picard`
URL-encoding

- certain characters are not allowed in URL query parameters:
  - examples: " ", ",/", ",=", ",&"  

- when passing a parameter, it is URL-encoded (reference table)  
  - “Allison's cool!?” → “Allison%27s+cool%3F%21"

- you don't usually need to worry about this:  
  - the browser automatically encodes parameters before sending them  
  - the PHP $_GET and $_POST arrays automatically decode them  
  - ... but occasionally the encoded version does pop up (e.g. in Firebug)
Submitting data to a web server

• though browsers mostly retrieve data, sometimes you want to submit data to a server
  • Hotmail: Send a message
  • Flickr: Upload a photo
  • Google Calendar: Create an appointment

• the data is sent in HTTP requests to the server
  • with HTML forms
  • with Ajax (seen later)

• the data is placed into the request as parameters
HTTP GET vs. POST requests

- **GET**: asks a server for a page or data
  - if the request has parameters, they are sent in the URL as a query string

- **POST**: submits data to a web server and retrieves the server's response
  - if the request has parameters, they are embedded in the request's HTTP packet, not the URL

For submitting data to be saved, POST is more appropriate than GET

- GET requests embed their parameters in their URLs
- URLs are limited in length (~1024 characters)
- URLs cannot contain special characters without encoding
- private data in a URL can be seen or modified by users
Form POST example

```html
<form action="http://foo.com/app.php" method="post">
  <div>
    Name: <input type="text" name="name" /> 
    Food: <input type="text" name="meal" />
    <label>Meat? <input type="checkbox" name="meat" /></label>
  </div>
  <input type="submit" />
</form>
```

Output:

Name: 
Food: 
Meat? 

Submit Query
GET or POST?

```php
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    # process a GET request
    ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    # process a POST request
    ...
}
```

- some PHP pages process both GET and POST requests
- to find out which kind of request we are currently processing,
- look at the global $_SERVER array's "REQUEST_METHOD" element
### Including files: include

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>include(&quot;filename&quot;)</code></td>
<td>inserts the entire contents of the given file into the PHP script's output page</td>
</tr>
<tr>
<td><code>include(&quot;header.html&quot;)</code></td>
<td>encourages modularity</td>
</tr>
<tr>
<td><code>include(&quot;shared-code.php&quot;)</code></td>
<td>useful for defining reused functions needed by multiple pages</td>
</tr>
<tr>
<td></td>
<td>related: include_once, require, require_once</td>
</tr>
</tbody>
</table>

**PHP**