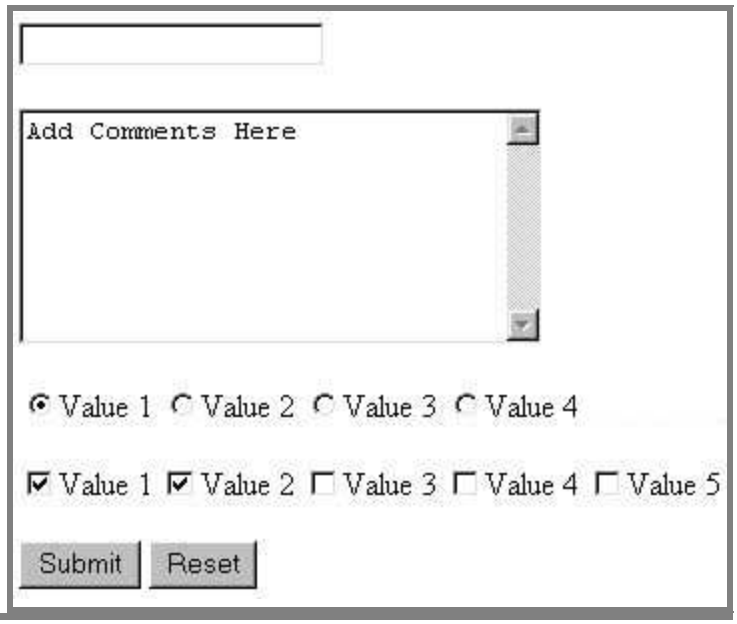
XHTML Forms

CSE 190 M (Web Programming), Spring 2008
University of Washington

References: JavascriptKit, w3schools

HTML forms

- an HTML **form** is a group of UI controls that accepts information from the user and sends the information to a web server
- forms use HTML UI controls (buttons, checkboxes, text fields, etc.)
- the information is sent to the server as a **query string**
- this is the other way, besides Ajax requests, to submit information to a server-side web service



HTML form: `<form>`

```
<form action="web service URL" method="GET or POST">
  <fieldset>
    form controls
  </fieldset>
</form>
```
HTML

- required `action` attribute gives the URL of the server web service that will process this form's data
- `method` attribute specifies whether the server should use an HTTP GET or POST request

Form example

```
<form action="http://www.foo.com/app.php" method="GET">
  <fieldset>
    <label>Name: <input type="text" name="name" /></label>
    <label>Meal: <input type="text" name="meal" /></label>
    <label>Meat?
      <input type="checkbox" name="meat" />
    </label>
    <input type="submit" />
  <fieldset>
</form>
```

HTML

- should wrap the form's controls in one or more `fieldset`s

Recall: the HTML UI controls

```
<textarea name="message" rows="4" cols="20">
Type your comments here.
</textarea>
```
HTML

Type your comments here.

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

Jerry

```
<input type="text" name="email" /><br />
<input type="password" name="pw" size="12" />
```
HTML

```
<label><input type="radio" name="cc" />Visa</label>
<label><input type="radio" name="cc" />MasterCard</label>
```
HTML

Visa MasterCard

```
<label>
<input type="checkbox" name="lettuce" />
  Lettuce</label>
<label>
<input type="checkbox" name="tomato" />
  Tomato</label>
```
HTML

Lettuce Tomato

Submit

`submit` and `reset` buttons

```HTML
<form action="http://www.foo.com/app.php" method="GET">
  <fieldset>
    ...
    <input type="submit" />
    <input type="reset" />
  </fieldset>
</form>
```

Submit Reset

- an `input` element with a `submit` attribute is displayed as a button that, when clicked, will send the parameters to the server and show the response
- an `input` element with a `reset` attribute is displayed as a button that, when clicked, will change the controls back to their original state

The `name` attribute

```HTML
<form action="http://foo.com/app.php" method="GET">
  <fieldset>
    Name: <input type="text" name="name" />
    Meal: <input type="text" name="meal" />
    <label>Meat? <input type="checkbox" name="meat" /></label>
    <input type="submit" />
  <fieldset>
</form>
```

- each control's `name` specifies the query string parameter to pass
- if user types "Sue" as name, "pizza" as meal, and checks Meat? box, then clicks Submit button, the browser will go to this URL:
  `http://foo.com/app.php?name=Sue&meal=pizza&meat=on`

`submit`, `reset` example

Name: Marty
Meal: pizza
Meat?
Submit Reset

- specify custom text on the buttons by setting their `value` attribute

```HTML
<input type="submit" value="Order Meal" />
```

Problems with checkboxes, radio buttons

```php
<form method="GET"
 action="http://webster.cs.washington.edu/params.php" >
  <label><input type="checkbox" name="meat" /> Meat</label> <br />
  <label><input type="radio" name="creditcard" /> Visa</label>
  <label><input type="radio" name="creditcard" /> MasterCard</label> <br />

  Favorite Star Trek captain:
  <select name="startrek">
    <option>Kirk</option>
    <option>Picard</option>
  </select> <br />

  <input type="submit" />
</form>
```

- the following form may look correct, but when you submit it...
- `[meat] => on, [creditcard] => on, [startrek] => Jean-Luc Picard`

The `value` attribute

```php
<form method="GET"
 action="http://webster.cs.washington.edu/params.php" >
  <label><input type="checkbox" name="meat" value="heckyeah"/> Meat</label> <br />
  <label><input type="radio" name="creditcard" value="visa" /> Visa</label>
  <label><input type="radio" name="creditcard" value="mastercard"/> MasterCard</label> <br />

  Favorite Star Trek captain:
  <select name="startrek">
    <option value="kirk">James T. Kirk</option>
    <option value="picard">Jean-Luc Picard</option>
  </select> <br />

  <input type="submit" />
</form>
```

- the `value` attribute controls what will be submitted if that control is selected (important for radio buttons)
- `[meat] => heckyeah, [creditcard] => visa, [startrek] => picard`

GET vs. POST

```HTML
<form action="http://www.foo.com/app.php" method="POST">
```

- a `GET` request passes the parameters to the server as a query string
- a `POST` request embeds the parameters in HTTP request, not in the URL
- advantages of `POST` :
  - `GET` is limited to browser's URL length, around 100-200 characters
  - information is more private (not shown in browser address bar)
- disadvantages of `POST` :
  - can't be bookmarked
  - browser can't easily go back (the famous POSTDATA message)

Uploading files

```PHP
Upload an image as your avatar:
<input type="file" name="avatar" />
<input type="submit" />
```

- add a file upload to your form as an `input` tag with `type` of `file`
- Note: not allowed to set the file path using the JS DOM (why?)

More about uploading files

```PHP
<form action="http://foo.com/app.php"
 method="POST" enctype="multipart/form-data">
  <fieldset>
    Upload an image as your avatar:
    <input type="file" name="avatar" />
    <input type="submit" />
  </fieldset>
</form>
```

- form's request method must be `POST` (an entire file can't be put into a URL!)
- form's `enctype` (data encoding type) must be set to `multipart/form-data` or else the file will not arrive at the server

Processing an uploaded file in PHP

- uploaded files are placed into a global associative array named `$_FILES`
- each element of `$_FILES` is an assoc. array, containing these elements:
  - `name` - the local filename that the user uploaded
  - `type` - the MIME type of data that was uploaded, such as `image/jpeg`
  - `tmp_name` - a filename where PHP has temporarily saved the uploaded file
    - to permanently store the file, move it from this location into some other file
  - `size` - file's size in bytes
- example: if you upload `example.txt` as a parameter named `logfile`,
  - `$_FILES["logfile"]["name"]` will be `example.txt`
  - `$_FILES["logfile"]["type"]` will be `text/plain`
  - `$_FILES["logfile"]["tmp_name"]` will be something like `/var/tmp/phpZtR4TI`

Processing uploaded file, example

```php
$username = $_REQUEST["username"];
if (is_uploaded_file($_FILES["avatar"]["tmp_name"])) {
  move_uploaded_file($_FILES["avatar"]["tmp_name"], "$username/avatar.jpg");
  print "Saved uploaded file as $username/avatar.jpg\n";
} else {
  die "Error: required file not uploaded";
}
```

- functions for dealing with uploaded files:
  - `is_uploaded_file(`*filename*`)`
    returns `TRUE` if the given filename was uploaded by the user
  - `move_uploaded_file(`*from*`, `*to*`)`
    moves from a temporary file location to a more permanent file
- proper idiom: check `is_uploaded_file`, then do `move_uploaded_file`

Hidden input parameters

```
Name: <input type="text" name="username" size="16" /> <br />
SID: <input type="text" name="sid" size="8" /> <br />

<input type="hidden" name="school" value="UW" />
<input type="hidden" name="quarter" value="08sp" />
<input type="submit" value="Retrieve student info" />
```
PHP

Name:
SID:
Retrieve student info

- an `input` tag with `type` of `hidden` is a parameter that won't appear on the page, but is still passed to the server when form is submitted
- useful for passing on additional state that isn't modified by the user

Recall: Styling form controls

```
element[attribute="value"] {
  ...
}
```
CSS

```
input[type="text"] {
  color: blue;
  font-style: italic;
  border: 4px solid yellow;
}
```
CSS

example text

- a CSS **attribute selector** affects an element only if it has the given attribute set to the given value
- often used with forms, because `input` represents many controls