Web Forms

CSE/ISE 102: Intro to Web Design
Stony Brook University

The Interactive Web

- The Web started out as a collection of static pages
- Now it's a place where people go to get things done
- We use *forms* to handle these interactions with users

The Parts of a Form

- Every Web form has two parts:
  - HTML markup that provides buttons, input fields, and drop-down menus
  - an application or server-side script that processes form information and returns some kind of response

Step-by-Step

1. Visitor opens a page with the Web form
2. User enters data into the form fields
3. Browser encodes data, sends it to server
4. Web application processes the data
5. Web application returns a response
6. Server sends response back to browser

The `form` Element

- `form` is a container for the form content
  - this includes form controls and other HTML (but not another form!)
- `form` also has several attributes that are required for interacting with the server

Sample Form

```
<form action="/mailinglist.php" method="post">
<fieldset>
<legend>Join our e-mail list</legend>
<p>descriptive text</p>
<ol>
<li><label for="firstlast">Name:</label>
      <input type="text" name="username" id="firstlast"></li>
<li><label for="email">E-mail:</label>
      <input type="text" name="email" id="email"></li>
</ol>
<input type="submit" value="Submit">
</fieldset>
</form>
```

Form Attributes

- `action`
  - provides URL of the application or script (*action page*) that will process the form
- `method`
  - specifies how the info should be sent
    - POST or GET (default)

POST vs. GET

- POST
  - browser sends a separate server request containing special headers and data
  - best for sensitive information or lots of data
- GET
  - encoded form data is added to URL sent to server, following a question mark
  - www.foo.com/maillist.php?name=Sally&email=...

Variables and Content

- Variables: information collected by the form
  - user-entered data is the value/content of the variable
- `name` attribute provides the name for a control
  - User info is sent to server as name-value pairs
    - `name=Sally%20Strongarm`
  - All form control elements MUST include a name

Form Controls

- Text entry controls
- Specialized text entry controls
- Submit and reset buttons
- Radio and checkbox buttons
- Pull-down and scrolling menus
- File selection and upload control
- Hidden controls

Text Entry Controls

- input
  - single-line text field
  - set `type` to "text"
  - use `value` to specify default text
  - use maxlength to limit the # of characters that can be entered

Multiline Text Entry

- Use `textarea` to display a scrollable, multiline text entry box
  - has a closing tag
  - can use `rows` and `cols` to set size
  - `wrap` attribute specifies whether to keep or lose line breaks (can be *soft* or *hard*)

Specialized Text Entry

- Password entry field
  - use normal input element, but type is "password"
  - obscures whatever is typed there
    - Note: does NOT encrypt the contents!

Submit and Reset Buttons

- Use the `input` element to add these
- Specify the type as "submit" or "reset"
- The `value` attribute determines the button text

Radio Buttons

- Radio buttons only permit a single selection
  - use input type "radio"
  - `name` attribute is required
    - radio buttons with the same name are grouped together into a set
      - Use a unique `value` for each button
  - the `checked` attribute sets a default

Checkboxes

- Allow multiple simultaneous selections
  - Use input type "checkbox"
- Assign same name, as with radio buttons
- May be set to checked by default

Menus

- More compact than groups of buttons and checkboxes
- Two forms: drop-down and scrolling
  - this depends on size and whether multiple options can be selected
- Uses the `select` element

Drop-down Menus

- `select` displays as a drop-down menu (or pull-down menu) when `size` attribute is set to 1 or omitted
  - Use `option` to identify menu items
    - add `value` attribute to send a different value to the server

```
<p>What is your favorite band?
<select name="Fave">
   <option>The Cure</option>
   <option>Cocteau Twins</option>
   <option value="DM">Depeche Mode</option>
   <option>New Order</option>
</select>
</p>
```

Scrolling Menus

- Add a `size` attribute to `select` indicating number of lines to make visible
  - Use the `multiple` attribute to allow multiple selections
- Add the `selected` attribute to options to select them by default
  - this can also be used with pull-down menus

Grouping Menu Options

- Use `optgroup` to create conceptual groups of elements
  - label attribute is required to provide a heading for the group

```
<select ...>
<optgroup label="traditional">
    <option>...</option>
</optgroup>
</select>
```

File Selection

- Can be used to select a file to upload
- Use the "file" type with `input`
- If you do this, you must use POST, and you must set the encoding type (enctype) to "multipart/form-data" in your form element

Hidden Controls

- Used to send additional information to the form-processing application
- Use the "hidden" type with `input`
  - passes a name/value pair to the server

Form Accessibility

- Problem: what about users who are not using visual browsers?
- Need to clarify the semantic connections between form components
  - `label`, `fieldset`, and `legend`

Labels

- A label associates descriptive text with a form field
  - a label is associated with exactly 1 control
- *Implicit association* nests the control and description inside the `label` element
  - this is the only way to label radio buttons and checkboxes

Labels, part 2

- *Explicit association* matches the label with the control's `id` reference
  - uses the `for` attribute

```
<label for="username">Login account</label>
<input type="text" name="login"
id="username">
```

`fieldset` and `legend`

- `fieldset` indicates a logical group of form controls
- A `fieldset` may include a `legend` that provides a caption for the contained fields

```
<fieldset>
<legend>Mailing List Sign-up</legend>
<ul>
  <li><Label>Add me to your list
    <input type="radio" name="list"
     value="yes" checked="checked">
  </label></li>
  <li><label>No thanks <input type="radio"
      name="list" value="no">
  </label></li>
</ul>
</fieldset>
```

Form Layout and Design

Usable Forms

- Goal: make the process as smooth as possible
  - Avoid unnecessary questions
  - Consider impact of label placement
    - put labels above their respective fields
  - Choose input types carefully
  - Group related inputs
  - Clarify primary and secondary actions

Styling Forms

- We can use CSS to create a clean form layout
  - e.g., a consistent look
- Form elements can be styled in terms of colors, dimensions, fonts, and background effects
- This can be tricky, though...

Available Options

- Text inputs: width, height, background-color, color, background-image, border, margin, padding
- Textareas: line-height, resize (for resize handles), plus anything for text input controls
- Button inputs: width, height, margin, padding, background, box-shadow
  - CSS syntax: input[type="submit"]

More Options

- Radio and checkbox buttons: leave these alone, or use Javascript
- Drop-down and select menus: width, height, or just leave them to be rendered as-is by the browser
- Fieldsets and legends: border, background, margin, padding

```
ul { list-style-type: none; }

ul li { clear: both; }

form { width: 40em;
       border: 1px solid #666;
       border-radius: 10px;
   box-shadow: .2em .2em .5em #999;
       background-color: #d0e9f6;
       padding: 1em;
       overflow: hidden; }
```

```
label { display: block;
        float: left;
        width: 10em;
        text-align: right;
        margin-right: .5em;}

input.textinput { width: 30em;
                  height: 2em;
     border: 1px solid #666; }
```

Next Time

- Form processing with Perl and CGI