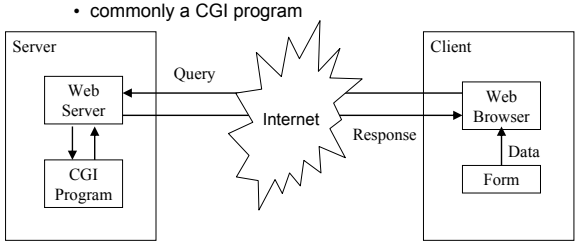
Forms and Form Processing

The HTML Form

- Enables browsers to collect info from users and send the data to a designated server
  - sent via an HTTP POST or GET query
  - The receiving server invokes specified programming to process the data and provide a response
    - commonly a CGI program



The <form> element

- Designed to collect input from Web surfers for processing on the server side
- To create a form, place different types of input-control elements inside a form
  - text input (text fields & text areas)
  - click buttons
  - radio buttons
  - check boxes
  - pull-down menus
- Input elements for creating these controls:
  - <input>, <select>, <textarea>, & <button>

Starting XHTML Document

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
 <head>
  <title>Web Page with Forms</title>
  <link rel="stylesheet" type="text/css" href="./css/MyStyleSheet.css" />
 </head>

 <body>

</body>
</html>
```


Add to your document's body

Must be either *get* or *post*, both in lower case

```
<form method="post"
      action="/cgi-bin/member.cgi">
<pre>
Name:  <input name="name" size="35" />

Email: <input name="email" size="35" />

       <input type="submit" value="Send" />
</pre>
</form>
```


Form Basics

- A typical form element consists of these essential parts:
  1. Instructions for the user on what info is needed & how to fill out the form
  2. Blanks to be completed by the user
  3. Labels for each input control
  4. A submit button
  5. An HTTP query method (post is advisable)
  6. URL of server-side program to receive & process the collected form data
     - given by the required action attribute

Input Control Elements

- Are inline elements
- Must be placed inside block-level elements before being placed inside a form
- We will use them outside forms later
  - with JavaScript
- NOTE: forms may not contain other forms
  - no form nesting

Single line text input

- For 1 line of input (text field), use *input* element with type text, ex:

<input **name**="lastname" **type**="text" **size**="20" **maxlength**="30" />

- The code above would display a text field
  - approximately 20 characters wide
  - max of 30 characters could be entered
  - **name** attribute specifies the key for an input element
  - text entered by user becomes the **value** of the control
- Input for this text field would be sent to the server as a key-value pair, for example:

  lastname=Esmaili
- You may also specify a value attribute if you wish to have an initial text value. Ex, add the following attribute to input:

  value="Enter your last name"

Multi-line text input

- For collecting multiple lines of input (text area), use *textarea* element, ex:

Please let us have your comments:

<textarea **name**="feedback" **rows**="4" **cols**="60">

Tell us what you really think, please.

</textarea>

- Text areas scroll automatically
- **readonly**="readonly" attribute may be specified to make it uneditable

Radio Buttons

- A group of radio buttons allows the user to choose one from a number of choices
- Clicking one choice selects it & deselects all others in the group
- Use input element with type radio for a radio button
- All radio buttons in the same group should have the same **name** attribute
  - use **id** attribute as key for each button
  - use **checked**="checked" attribute to identify initial checked button, which may only be one in the group

Radio Buttons Example

```
<p style="font-size: larger; font-weight: bold;">
Choose a color:
<input type="radio" name="color" value="red" checked="checked"
   />Red
<input type="radio" name="color" value="green" />Green
<input type="radio" name="color" value="blue" />Blue
</p>
```

- Name-value pair sent to server would be **color**=value, ex: **color**=red
- Use label elements to label each button
  - use **for** attribute to tie to button
    - so clicking the label will click the button

Radio Buttons Example (cont'd)

```
<p style="font-size: larger; font-weight: bold;">
Choose a color:
<input id="r" type="radio" name="color" value="red"
   checked="checked" />
  <label for="r" style="color: red">Red</label>
<input id="g" type="radio" name="color" value="green" />
  <label for="g" style="color: green">Green</label>
<input id="b" type="radio" name="color" value="blue" />
  <label for="b" style="color: blue">Blue</label>
</p>
```

- Name-value pair sent to server would be **color**=value, ex: **color**=red

Check Boxes

- Allow users to choose several items from a list of choices
- Use input element with **type** checkbox
- Clicking a check box selects or deselects it without affecting other check boxes
  - A user may select all, none, or any number of check box combinations

Check box Example

```
<p style="font-size: larger; font-weight: bold;">
Your favorite sports:
<input id="t" type="checkbox" name="tennis" checked="checked" />
  <label for="t">Tennis</label>
<input id="b" type="checkbox" name="baseball" />
  <label for="b">Baseball</label>
<input id="w" type="checkbox" name="windsurfing" />
  <label for="w">Wind Surfing</label>
</p>
```

- Each selected item is sent to the server as **name**=on, or **name**=off ex: **tennis**=on, **baseball**=off

Pull-down Menus

- For making combo boxes or lists
- For when there are many choices
- Use select element and include option elements
- Each option presents a different selectable item in the list
- **size** attribute specifies how many options are displayed on the menu at one time
  - **size**="1" for making a *combo box*
  - all other sizes for making a *list*
- To allow multiple choices to be selected, use **multiple**="multiple" attribute

Pull-down Menu Example

```
<p style="font-size: larger; font-weight: bold;">
State:
<select id="statename" name="statename" size="1" />
<option value="0"> Pick One </option>
<option value="Alabama"> Alabama </option>
<option value="Alaska"> Alaska </option>
<option value="Maine"> Maine </option>
<option value="New York"> New York </option>
</select>
 </p>
```

- One name value pair is sent to the server for each option selected, ex: **name**=value, ex: **statename**=Maine

The submit button

- The basic Submit button for a form is:

```
<input type="submit"  value="button-label" />
```

- value attribute will appear on button, ex:

```
<input type="submit"  value="Go" />
```

- You may supply an image instead:

```
<input type="image" src="url" name="key" />
```

- Sent to server as **submit**=value, ex: **submit**=Go
- Other ways of making submit buttons:

```
<button name="submit" value="join">Join</button>
```

- You may place img elements inside a button as well
- Also, button type input elements & button type button elements may be used (we'll see these w/ JavaScript)

File Uploading

- Forms allow us to upload files from the user's computer
- Use the input element with **type**="file"
  - query **method** must be post
  - **enctype**=… is needed when specifying a data format different from the default
  - accept attribute specifies the required MIME type (see http://www.iana.org/assignments/media-types/) for the uploaded file

File Uploading Example

```
<form method="post" action="/cgi-bin/receive.cgi"
   enctype="multipart/form-data">
<p style="font-size: larger; font-weight: bold;">
Submit your paper to the conference:</p>
<p><input type="file" name="paper"
   accept="application/pdf" /></p>
<p><input type="submit" value="Upload" /></p>
</form>
```