



Internet and Web-Based Systems Development

Lecture 5

XHTML Forms & More on HTTP



Lecture 5

- XHTML Form
 - Text Field
 - Textarea
 - Check Box
 - Button
 - Radio Button
 - Drop-down List
- HTTP Protocol



Introduction

- When browsing Web sites, users often need to input information, e.g.,
 - Email addresses, search keywords, choices
- XHTML provides a mechanism, called a *form*, for collecting such user information
- Any data entered in the form is sent to the Web server using the HTTP protocol

XHTML Forms

- Template

```
<form method="post" action="register.php">  
  <!-- Some form elements -->  
  <!-- Some form elements -->  
  <!-- Some form elements -->  
  <!-- Submit Button -->  
</form>
```

Action

- action = "register.php"
- The data entered in the form will be transmitted to the indicated server-side program
 - register.php is written in PHP (Hypertext Pre-Processor) embedded in XHTML tags (more on this later)

Form Elements

- Common attribute

 - NAME

 - Identifies individual elements by the server when the form is submitted

- Text Control

 - Text Field

 - `<input type="text" name="..." />`

 - For single-line text input
 - Other attributes

 - VALUE – The initial content
 - SIZE – The width of the text field displayed on the screen
 - MAXLENGTH – The maximum number of characters allowed
 - DISABLED, READONLY – Make the text field disabled or read-only

Form Elements

- Password Field

`<input type="password" name="..." />`

- Attributes: The same as text field
- The characters will appear as * when being typed in this field

Form Elements

- Text Area
 - <Textarea NAME="..." ROWS=xxx COLS=xxx>
InitialText
 - </textarea>
 - For multiple-line text input
 - Attributes
 - ROWS (required) – number of visible lines of text
 - COLS (required) – number of visible width of the text area
 - WRAP
 - OFF: no word wrap
 - HARD: word wrap, add line break when submit the form
 - SOFT: word wrap, no extra line breaks to be transmitted
 - DISABLED, READONLY

Form Elements

○ Button

- Submit Button

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

- After clicked, the form is sent to server

- Reset Button

- `<input type="reset" />`

- After clicked, all values of the form is reset to initial status

- Normal Button

- `<input type="button" />`

- Common attributes

- VALUE – the text shown on the button
 - DISABLED

Form Elements

○ Checkboxes

<label>

<input type="checkbox" name="..." />The Checkbox String

</label>

- Attributes

- VALUE

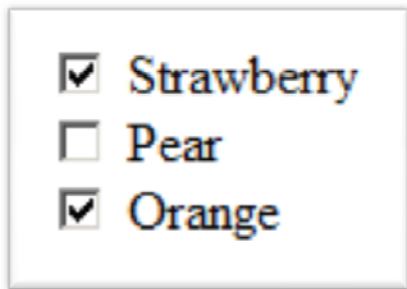
- Default is ON. If the button is selected, the `name=value` pair will be added to data string in the HTTP request message

- CHECKED

- Indicate the checkbox is initially checked when the page is loaded

- DISABLED

- The `<label>` tag provides a usability improvement for mouse users, because if the user clicks on the text within the label element, it toggles the control



☒ Strawberry
☐ Pear
☒ Orange

Form Elements

○ Radio Buttons

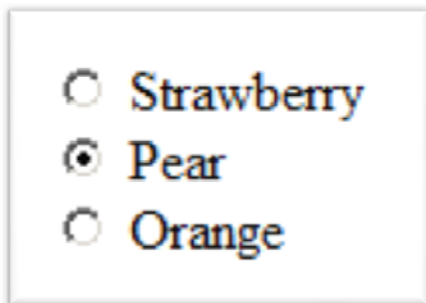
<label>

```
<input type="radio" name="..." value="..." />
```

The Radio Button String

</label>

- Different from checkbox that only a single radio button in a given group can be selected
- A group of radio buttons are those having the same value in the **name** attribute
- Attributes
 - **VALUE**
 - If the button is selected, the name=value pair will be added to data string
 - **CHECKED**
 - Indicate the radio button is initially checked
 - **DISABLED** and **READONLY**



☐ Strawberry

☒ Pear

☐ Orange

Form Elements

- Combo boxes and List boxes

- `<select name="...">`

- `<option value="value1">Choice 1 Text`

- `<option value="value2">Choice 2 Text`

- `<option value="value3">Choice 3 Text`

- `</select>`

- Combo Boxes (drop-down list)

- One option can be chosen

- List Boxes

- Multiple selections can be made

Form Elements

SELECT

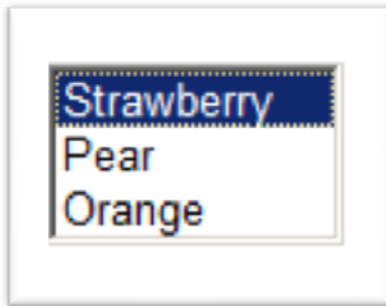
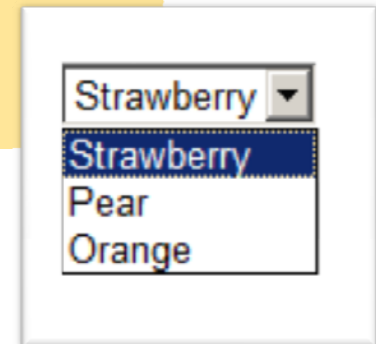
- NAME – name of this list
- SIZE – number of visible rows
- MULTIPLE – allow multiple selections
- DISABLED

OPTION

- SELECTED – the item is initially selected
- VALUE – the `name=value` pair to be transmitted to server
- DISABLED

Form Elements

```
<select name="fruit">  
  <option value="Strawberry">Strawberry</option>  
  <option value="Pear">Pear</option>  
  <option value="Orange">Orange</option>  
</select>
```



```
<select name="fruit" size="3" multiple>  
  <option value="Strawberry">Strawberry</option>  
  <option value="Pear">Pear</option>  
  <option value="Orange">Orange</option>  
</select>
```

Form Elements

- Hidden Fields

```
<input type="hidden" name="..."  
value="...">
```

- Do not affect the appearance of the page that is presented to the user
- Purposes
 - Tracking users as they move around within a site
 - Provide predefined input to a server-side program
 - Stored contextual information in pages that are generated by the server
 - E.g., an online shopping cart information

Form Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Form Demo</title>
</head>
<body>
<form action="feedback.php" method="post" name="feedbackForm">
  <p><b>Feedback
    Form</b></p>
  <p>Please fill out the following form.</p>
  <p>Name:
    <input type="text" name="name" />
  </p>
  <p>Comments:<br />
    <textarea name="comments"></textarea>
  </p>
```

form.html (Page 1 of 4)

<p>E-mail Address:

<input type="text" name="email" />

</p>

<p>Which are your favourite fruit(s)?

<label>

<input type="checkbox" name="star" value="banana" />

Banana</label>

<label>

<input type="checkbox" name="star" value="apple" />

Apple</label>

<label>

<input type="checkbox" name="star" value="pear" />

Pear</label>

<label>

<input type="checkbox" name="star" value="grape" />

Grape</label>

<label>

<input type="checkbox" name="star" value="watermelon" />

Watermelon</label> </p>

<p>How did you get to our site?

<label>

<input type="radio" name="howto" value="searchengine"
checked="checked" />

Search engine</label>

<label>

<input type="radio" name="howto" value="links" />

Links from another site</label>

<label>

<input type="radio" name="howto" value="forum" />

Forum</label>

<label>

<input type="radio" name="howto" value="magazines" />

Magazines</label>

<label>

<input type="radio" name="howto" value="others" />

Others</label>

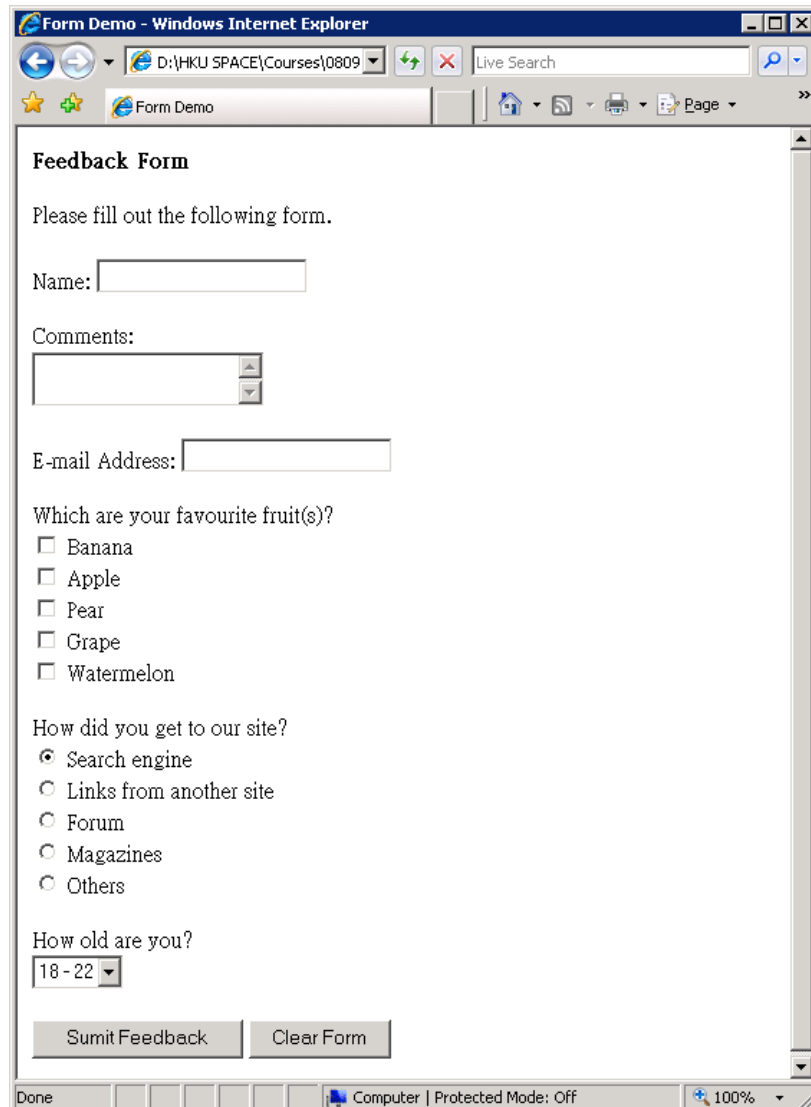
</p>



```
<p>How old are you?<br />
  <select name="age">
    <option value="18" selected="selected">18 - 22</option>
    <option value="23">23 - 26</option>
    <option value="27">27 - 30</option>
  </select>
</p>
<p>
  <input type="hidden" name="subject" value="feedbackForm" />
</p>
<p>
  <input type="submit" name="Submit" value="Sumit Feedback" />
  <input type="reset" name="Clear" value="Clear Form" />
</p>
</form>
</body>
</html>
```

form.html (Page 4 of 4)

Form Example



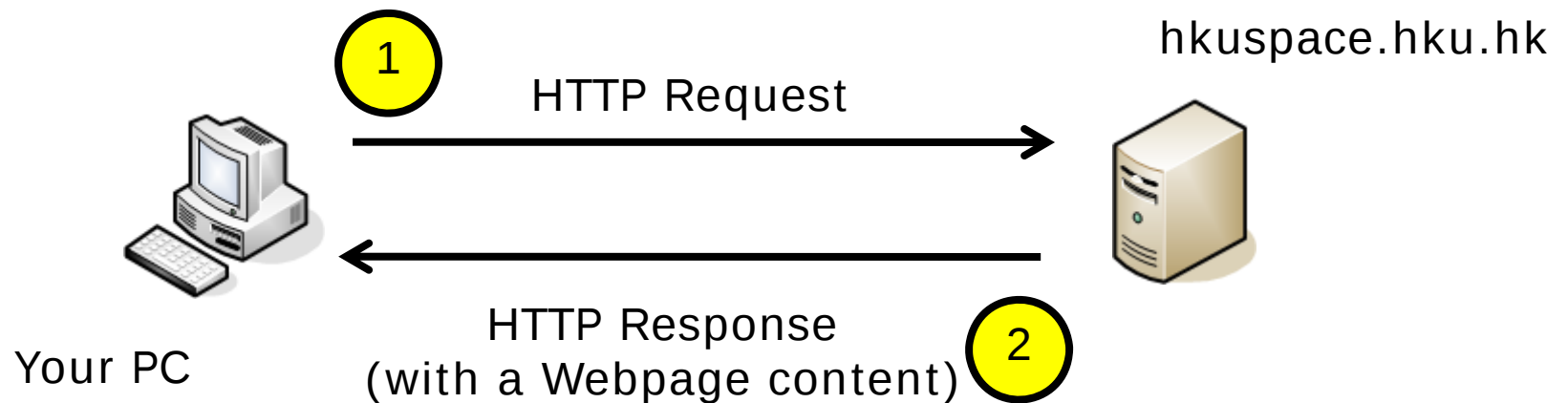
The screenshot shows a Windows Internet Explorer browser window titled "Form Demo - Windows Internet Explorer". The address bar displays "D:\HKU SPACE\Courses\0809" and the search bar contains "Live Search". The page content is titled "Feedback Form" and includes the instruction "Please fill out the following form.".

The form fields and options are as follows:

- Name:
- Comments:
- E-mail Address:
- Which are your favourite fruit(s)?
 - ☐ Banana
 - ☐ Apple
 - ☐ Pear
 - ☐ Grape
 - ☐ Watermelon
- How did you get to our site?
 - ☒ Search engine
 - ☐ Links from another site
 - ☐ Forum
 - ☐ Magazines
 - ☐ Others
- How old are you?

At the bottom of the form are two buttons: "Sumit Feedback" and "Clear Form".

Recall: HTTP request-response behaviour



HTTP – Background

- An application layer protocol
 - used between user agents (browser) and servers (web) for communications
 - first version developed in around 1990 HTTP/0.9, followed by HTTP/1.0, prevailing version is HTTP/1.1
 - HTTP/1.1 adds in more functionality, has more stringent requirement, allows multiple requests/responses exchange over a single connection
 - <http://www.research.att.com/~bala/papers/h0vh1.html>

HTTP Request Example

GET /form.htm HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/xaml+xml, application/vnd.ms-xpsdocument, application/x-ms-xbap, application/x-ms-application, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*

Accept-Language: zh-hk

UA-CPU: x86

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322; .NET CLR 2.0.50727; .NET CLR 3.0.04506.30; .NET CLR 3.0.04506.648; InfoPath.2; OfficeLiveConnector.1.3; OfficeLivePatch.0.0; .NET CLR 3.0.4506.2152; .NET CLR 3.5.30729)

Host: localhost:8080

Connection: Keep-Alive

Another HTTP Request Example

POST /form.php HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/x-ms-application, application/vnd.ms-xpsdocument,
application/xaml+xml, application/x-ms-xbap, application/x-
shockwave-flash, application/vnd.ms-excel, application/vnd.ms-
powerpoint, application/msword, */*

Accept-Language: zh-hk

Content-Type: application/x-www-form-urlencoded

UA-CPU: x86

Accept-Encoding: gzip, deflate

User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.0;
SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.04506; InfoPath.2)

Host: localhost:8080

Content-Length: 29

Connection: Keep-Alive

Cache-Control: no-cache

firstName=Dennis&lastName=Liu

HTTP Request Message Structure

Request-Line

`* (general-header | request-header |
entity-header)`

CRLF

`[ message-body ]`

HTTP Request Message

Request-Line =

Method *Space* Request-URI *Space* HTTP-Version *CRLF*

- Method
 - OPTION, GET, HEAD, POST, PUT, DELETE, TRACE or extension method
- Request-URI
 - identify the relative path of a resource on the Web server
- HTTP-Version
 - indicate the HTTP version adopted in the message

HTTP Request Message

- General header
 - applicable to both request & response messages, apply to the message itself but not the content (entity)
 - E.g., Cache-Control
- Request header
 - gives additional info about the request, about the client; like the parameters in invoking the Method
 - E.g., Accept-Encoding, Accept-Language
- Entity-header
 - fields define optional meta-information about the entity-body
 - E.g., Content-Length



HTTP Response Message Example

HTTP/1.1 200 OK

Server: Apache-Coyote/1.1

Last-Modified: Fri, 18 Jan 2008
14:04:02 GMT

Content-Type: text/html

Content-Length: 730

Date: Fri, 18 Jan 2008 15:31:50 GMT

Connection: close

(The response message continues in the next slide)

HTTP Response Example

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>A sample page</title>
</head>
<body>
<form action="form.php" method="post">
  <table align="center" border="0">
    <tr>
      <td width="150">First Name:</td>
      <td><input name="firstName" type="text" /></td>
    </tr>
    <tr>
      <td>Last Name:</td>
      <td><input name="lastName" type="text" /></td>
    </tr>
    <tr>
      <td colspan="2" align="center"><input name="" type="submit" value="Submit" /></td>
    </tr>
  </table>
</form>
</body>
</html>
```

HTTP Response Message Structure

Status-Line

* (general-header | response-header |
entity-header)

CRLF

[message-body]

Status-Line =

HTTP-Version *Space* Status-Code
Space Reason-Phrase *CRLF*

Response Message Types

Status Code	Reason Phrase
200	OK
401	Unauthorized
403	Forbidden
404	Not Found
408	Request Timeout
500	Internal Server Error