

CS472 Web Programming

HTML Forms: Connecting with the Source

Except where otherwise noted, the contents of this document are Copyright 2012 Marty Stepp, Jessica Miller, Victoria Kirst and Roy McElmurry IV. All rights reserved. Any redistribution, reproduction, transmission, or storage of part or all of the contents in any form is prohibited without the author's expressed written permission. Slides have been modified for Maharishi University of Management Computer Science course CS472 in accordance with instructors agreement with authors.

Maharishi University of Management -Fairfield, Iowa © 2016



All rights reserved. No part of this slide presentation may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying or recording, or by any information storage and retrieval system, without permission in writing from Maharishi University of Management.

Wholeness Statement

- ▶ In this lecture we will discuss how to generate and process user input. On the client side, we will create HTML forms with different types of widgets that allow the users to submit different types of data, and on the server side we will look at processing these different types of data. *The first expression of the Unified Field is Rishi (knower), Devata (process of knowing), Chhandas (known).*

Query strings and parameters

► **Query string:** a set of parameters passed from a browser to a web server. Often passed by placing name/value pairs at the end of a URL.

► Below, parameter **username** has value “tina”, and **sid** has value “123456”

<http://mum.edu/login.jsp?username=tina&sid=123456>

<http://www.google.com/search?q=Obama>

<https://www.google.com/?q=HTML+Form>

Request Command

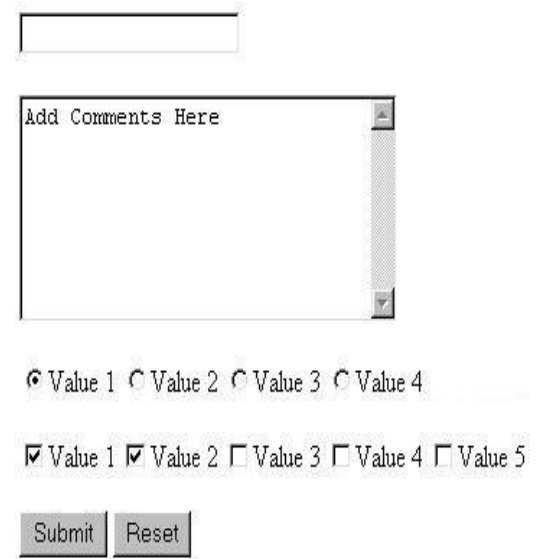
- ▶ There are 3 ways to send a request from a browser tab to the server:
 - ▶ Type url (GET)
 - ▶ Form (GET, POST.. etc)
 - ▶ XHR Request (GET, POST.. etc)
- ▶ Request methods:
 - ▶ GET: only has header (parameter are sent in the header, NO body)
 - ▶ POST: has header and body ([parameters are sent in the body)

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 (request header)
 - ▶ Some older browsers might limit length of URL
 - ▶ URLs cannot contain special characters without encoding
 - ▶ private data in a URL can be seen or modified by users
- ▶ **POST** : submits data to a web server (to be saved in DB or file or updates state in server application)
 - ▶ parameters are embedded in the HTTP request body, not the URL

HTML forms

- ▶ Form: a group of UI controls that accepts information from the user and sends the information to a web server
- ▶ The information is sent to the server as a query string
- ▶ JavaScript can be used to create interactive controls (seen later)



The image shows a web form with the following elements:

- A single-line text input field at the top.
- A text area below it with the placeholder text "Add Comments Here".
- A row of four radio buttons labeled "Value 1", "Value 2", "Value 3", and "Value 4". "Value 1" is selected.
- A row of five checkboxes labeled "Value 1", "Value 2", "Value 3", "Value 4", and "Value 5". "Value 1" and "Value 2" are checked.
- At the bottom, two buttons: "Submit" and "Reset".

HTML form: `<form>`

The **<form>** tag is used to create an HTML form for user input.

The **<form>** element can contain one or more of the following form elements:

<input>, **<textarea>**, **<button>**, **<select>**, **<option>**, **<optgroup>**,
<fieldset>, **<label>**, **<textarea>**, **<datalist>**, **<output>**

```
<form action="sales.html" method="get" novalidate  
autocomplete="on">
```

Form controls

Form Attributes

- ▶ **action** destination URL
- ▶ **method** get, post
- ▶ **novalidate** (HTML5) specifies that the form should not be validated when submitted
- ▶ **autocomplete** (HTML5) on, off

Form Example

```
<form action="http://www.google.com/search">  
  <div> Let's search Google <input name="q" />  
  <input type="submit" /> </div>  
</form>
```

Let's search Google

Submit Query

See example: [lecture4_examples/form.html](#)

Main Points

- ▶ An HTML form allows the user to send data (input parameters) to the server. Forms are created with the `<form>` tag, and can be submitted with either an HTTP GET or POST method.



Form controls: <input>

```
<input type="text" name="q" value="Colbert Report" />  
<input type="submit" value="Booyah!" />
```







- ▶ **input element** is used to create many UI controls
 - ▶ an inline element that **MUST** be self-closed
- ▶ **name attribute** specifies name/key of query parameter to **pass to server**
- ▶ **type** **can be** button, checkbox, file, hidden, password, radio, reset, submit, text, ...
- ▶ **value attribute** specifies control's initial text



Text fields: <input>

```
<input type="text" name="username" size="10" maxlength="8" />
```

```
<input type="text" name="password" size="8" />
```

Attribute	Value(s)	Description
value	text	Initial text to appear in text box
size	integer	Visible length of text box, in characters
maxlength	integer	Maximum number of chars that may be typed into text box
autocomplete 	on, off	Whether to offer suggestions of text to auto-complete the field
autofocus 	autofocus	Makes control initially receive keyboard focus
novalidate 	novalidate	Indicates browser should not check value before submitting
placeholder 	text	A hint or example of what the user should type;
pattern 	regular expr.	A regular expression indicating what input is valid
required 	required	Whether browser should display an error if blank



Checkboxes

- ▶ yes/no choices that can be checked and unchecked (**inline**)
 - ▶ none, 1, or many checkboxes can be checked at same time
 - ▶ Use the **checked** attribute in HTML to initially check the box

```
<input type="checkbox" name="lettuce" /> Lettuce
```

```
<input type="checkbox" name="tomato" checked /> Tomato
```

```
<input type="checkbox" name="pickles" checked /> Pickles
```

☐ Lettuce ☒ Tomato ☒ Pickles



Radio buttons

- ▶ Sets of mutually exclusive choices (**inline**)
 - ▶ Grouped by **name** attribute (only one can be checked at a time)
 - ▶ Must specify a **value** for each one or else it will be sent as value **on**

```
<input type="radio" name="cc" value="visa" checked /> Visa
```

```
<input type="radio" name="cc" value="mastercard" /> MasterCard
```

```
<input type="radio" name="cc" value="amex" /> American Express
```

☒ Visa ☐ MasterCard ☐ American Express

<textarea>

- ▶ The <**textarea**> tag defines a multi-line text input control. (inline)
- ▶ A textarea can hold an unlimited number of characters, and the text renders in a fixed-width font (usually Courier).
- ▶ The size of a textarea can be specified by the **cols** and **rows** attributes, or even better; through CSS' **height** and **width** properties.

```
<textarea rows="4" cols="20">
```

Type your comments
here

Type your comments here.

```
</textarea>
```

See example: `lecture4_examples/textarea.html`

Text labels: `<label>`



- ▶ Associates nearby text with control, so you can **click text to activate control**
- ▶ Can be used with **checkboxes** or **radio** buttons
- ▶ **label** element can be targeted by CSS style rules

```
<label> <input type="radio" name="cc" value="visa"
checked="checked" /> Visa</label>
```

```
<label> <input type="radio" name="cc" value="mastercard" />
MasterCard</label>
```

```
<label> <input type="radio" name="cc" value="amex" /> American
Express</label>
```



- ▶ See example: [lecture4_examples/label1.html](#), [lecture4_examples/label2.html](#)

Drop-down list `<select>` and `<option>`



- ▶ Menus of choices that collapse and expand (inline)
 - ▶ **option** element represents each choice
 - ▶ **select** optional attributes: **disabled**, **multiple**, **size**
 - ▶ optional **selected** attribute sets which one is initially chosen

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

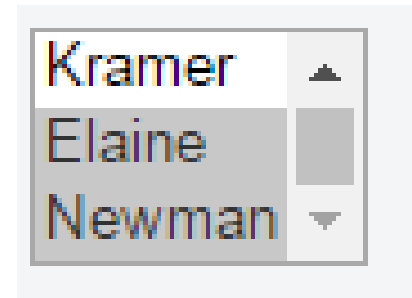
Kramer ▼



Using `<select>` for lists

- ▶ 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**

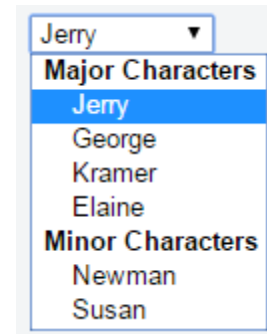
```
<select name="favoritecharacter[]" size="3" multiple>
  <option>Jerry</option>
  <option>George</option>
  <option>Kramer</option>
  <option>Elaine</option>
  <option selected>Newman</option>
</select>
```



Option groups: <optgroup>



```
<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>
```



Reset and Submit buttons

- ▶ When we click **reset** button, it returns all form controls to their initial values
- ▶ When we click **submit** buttons, it sends all data with the specified **method** (Get/Post) to the **action** page in the form
- ▶ Specify custom text on the button by setting its **value** attribute

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



Hidden input parameters



- ▶ An invisible parameter that is still passed to the server when form is submitted, it's useful for passing on additional state that isn't modified by the user

```
<input type="text" name="username" /> Name
```

```
<br />
```

```
<input type="text" name="sid" /> SID
```

```
<br />
```

```
<input type="hidden" name="school" value="MUM" />
```

```
<input type="hidden" name="year" value="2048" />
```

	Name
	SID

Grouping `<fieldset>`, `<legend>`



- ▶ Groups of input fields with optional caption (legend)

```
<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>
```

New Form Controls in HTML5



Input type	Description
color	A color from a palette of available choices
range	A slider for selecting values in a given range
date	A date such as August 29, 2016
time	A time of day such as 11:15 PM
datetime	A date and time such as 11:15 PM, August 29, 2016
month	A month of a particular year, such as August, 2016
week	A week of a particular year, such as August 35, 2016



Styling forms – attribute selector

- ▶ Because most input element are created using input tag, we target each group of elements using this CSS selector:

```
element[attribute="value"] {  
  property: value;  
  property: value;  
  ... property: value;  
}
```

```
input[type="text"] {  
  background-color: yellow;  
  font-weight: bold;  
}
```

CS472

Main Point

- ▶ HTML provides many different types of input widgets, including text fields, text areas, check boxes, radio buttons, and dropdown lists, this is also an area HTML 5 is expanding.

[Bonus] pattern



The **pattern** attribute specifies a regular expression that the `<input>` element's value is checked against. The **pattern** uses the ECMAScript (i.e. JavaScript) flavor of regex.

Note: The **pattern** attribute works with the following input types: **text**, **date**, **search**, **url**, **tel**, **email**, and **password**.

Tip: Use the global **title** attribute to describe the pattern to help the user.

```
<form action="demo_form.jsp">
```

Country code:

```
<input type="text" name="country_code" pattern="[A-Za-z]{3}" title=" Three letter country code ">
```

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

```
</form>
```

Country code:

! Please match the requested format.
Three letter country code

Regular expressions

`^[a-zA-Z_\-]+@([a-zA-Z_\-]+\.)+[a-zA-Z]{2,4}$`

- ▶ Regular expression ("regex"): a description of a pattern of text
- ▶ Can test whether a string matches the expression's pattern
- ▶ Regular expressions are extremely powerful but tough to read
 - ▶ (the above regular expression matches email addresses)
- ▶ Regular expressions are used in all languages:
 - ▶ Java, PHP ,JavaScript, HTML, C#, and other languages
- ▶ Many IDEs allow regexes in search/replace

Basic regular expressions

The simplest regexes simply matches any string that contains that text.

abc

above regular expression matches any string containing "abc":

- ▶ YES: "abc", "abcdef", "defabc", " .=.abc.=.", ...
- ▶ NO: " ABC" , " fedcba", "ab c", "PHP", ...
- ▶ Note that html5 has implicit anchors ^ and \$, so abc is really ^abc\$
- ▶ Regular expressions are case-sensitive by default.

Wildcards

A dot `.` matches exactly **one-character** except a `\n` line break

`.oo.y` matches "Doocy", "goofy", "LooNy", ...

Special characters: |, (), \

| means OR

abc|def|g matches "abc", "def", or "g"

() are for grouping

(Homer|Marge) Simpson

matches "Homer Simpson" or "Marge Simpson"

\ starts an escape sequence

many characters must be escaped to match them

literally: / \ \$. [] () ^ * + ?

<br\/> matches lines containing
 tags

Quantifiers: *, +, ?

***** means 0 or more occurrences

abc* matches "ab", "abc", "abcc", "abccc", ...

a(bc)* matches "a", "abc", "abcbc", "abcbcbc", ...

a.*a matches "aa", "aba", "a8qa", "a!?xyz__9a", ...

+ means 1 or more occurrences

a(bc)+ matches "abc", "abcbc", "abcbcbc", ...

Goo+gle matches "Google", "Goooogle", "Goooooogle",

...

? means 0 or 1 occurrences

a(bc)? matches "a" or "abc"

More quantifiers: {min,max}

{min,max} means between min and max occurrences (inclusive)

a(bc){2,4} matches "abcbc", "abcbcbc", or "abcbcbcbc"

min or **max** may be omitted to specify any number

{2,} means 2 or more

{,6} means up to 6

{3} means exactly 3

Anchors: ^ and \$

^ represents the beginning of the string or line;

\$ represents the end

Jess matches all strings that contain Jess;

^Jess matches all strings that start with Jess;

Jess\$ matches all strings that end with Jess;

^Jess\$ matches the exact string "Jess" only

^Mart.*Stepp\$ matches "MartStepp", "Marty Stepp", "Martin D Stepp", ... but NOT "Marty Stepp stinks" or "I H8 Martin Stepp"

The html5 spec states that ^ and \$ are implicit

Character sets: []

[] group characters into a character set, will match any **single character** from the set

[bcd]art matches strings containing "bart",
"cart", and "dart"
equivalent to **(b|c|d)art** but shorter

inside [], many of the modifier keys act as normal characters

what[!*?]* matches "what", "what!", "what?*!*!",
"what??!", ...

What regular expression matches DNA (strings of A, C, G, or T)?

[ACGT] +

Character ranges: [start-end]

inside a character set, specify a range of characters with -

[a-z] matches any lowercase letter

[a-zA-Z0-9] matches any lower- or uppercase letter or digit

an initial **^** inside a character set negates it

[^abcd] matches any character other than a, b, c, or d

inside a character set, - must be escaped to be matched

[+\-]?[0-9]+ matches an optional + or -, followed by at least one digit

What regular expression matches letter grades such as A, B+, or D- ?

[ABCDF][+\-]?

Escape sequences

Special escape sequence character sets:

\d matches any digit (same as [0-9])

\D any non-digit ([^0-9])

\w matches any word character (same as [a-zA-Z_0-9])

\W any non-word char

\s matches any whitespace character (, \t, \n, etc.)

\S any non-whitespace

What regular expression matches dollar amounts of at least \$100.00 ?

\\$[1-9]\d{2,}\.\d{2}

Example - URL



- ▶ An `<input>` element with `type="url"` that must start with `http://` or `https://` followed by at least one character:

```
<form action="demo_form.jsp">
```

Homepage:

```
<input type="url" name="website"
pattern="https?://.+" title="Please enter a URL">
```

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

```
</form>
```

The screenshot shows a web form with the label "Homepage:" followed by a text input field containing the text "dffd". To the right of the input field is a "Submit" button. Below the input field, a validation error message is displayed in a white box with a grey border. The message starts with an orange exclamation mark icon and the text "Please enter a URL.".

CONNECTING THE PARTS OF KNOWLEDGE WITH THE WHOLENESS OF KNOWLEDGE

HTML Forms: Connecting with the Source

1. Forms let us submit data to the web server, which can then generate a custom response based on server side information.
 2. GET requests are intended to only retrieve information and should be idempotent. POST requests are intended to submit data and not request a direct response.
-

3. Transcendental consciousness is the experience of the source of thought.

4. Impulses within the Transcendental field: duality is created as the boundless interacts with itself, creating the impression of this and that.

5. Wholeness moving within itself: In Unity Consciousness, one becomes aware that the Self is nothing but the boundless unity.



Advanced: Lookaround

Positive lookahead **(?=A) B**

Once a group starts with **?=** it means positive lookahead. Find expression A first, if found then expression B follows.

Negative lookahead **(?!A) B**

Once a group starts with **?!** it means negative lookahead. First check if expression A is not found, then check if expression B follows.

Example - email



- ▶ An `<input>` element with `type="email"` that must be in the following order: characters@characters.domain
(characters followed by an @ sign, followed by more characters, and then a ".")

```
<form action="demo_form.jsp">
```

E-mail:

```
<input type="email" name="email" pattern="[a-z0-9._+\-]+@[a-z0-9.\-]+\.[a-z]{2,3}">
```

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

```
</form>
```

E-mail:

Submit



Please include an '@' in the email address. 'fdfd' is missing an '@'.

Examples - password



- ▶ An `<input>` element with `type="password"` that must contain 8 or more characters that are of at least one number, and one uppercase and lowercase letter:

```
<form action="demo_form.jsp">
```

Password:

```
<input type="password" name="pw" pattern="(=?\d+) (=?[a-z]*) (=?[A-Z]*) .{8,}" title="Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters">
```

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

```
</form>
```

The screenshot shows a web form with a label "Password:" followed by a text input field containing five dots. To the right of the input field is a "Submit" button. Below the input field, a yellow warning icon is displayed next to the text: "Please match the requested format. Must contain at least one number and one uppercase and lowercase letter, and at least 8 or more characters".

Example - Search



- ▶ An <input> element with type="search" that CANNOT contain the following characters: ' or “

```
<form action="demo_form.jsp">
```

Search: <!-- x27 is a single quote and \x22 is a double quote -->

```
<input type="search" name="search"
pattern="^[^\x27\x22]+'" title="Invalid input">
```

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

Search:

! Please match the requested format.
Invalid input