

1

More forms

CSC443: Web Programming

Reset Buttons

2

```
Name: <input type="text" name="name" /> <br />
Food: <input type="text" name="meal" value="pizza" /> <br />
<label>Meat? <input type="checkbox" name="meat" /></label> <br />
<input type="reset" /> HTML
```

Name:

Food:

Meat? ☐

- specify custom text on the button by setting its value attribute

CSC443: Web Programming

Grouping input: <fieldset>, <legend>

3

```
<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> HTML
```

- `fieldset` groups related input fields, adds a border; `legend` supplies a caption

CSC443: Web Programming

Grouping input: <fieldset>, <legend>

4

```
<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> HTML
```

Credit cards:

☒ Visa ☐ MasterCard ☐ American Express

CSC443: Web Programming

Common UI control errors

5

- ❑ “I changed the form's HTML code ... but when I refresh, the page doesn't update!”
- ❑ By default, when you refresh a page, it leaves the previous values in all form controls
 - it does this in case you were filling out a long form and needed to refresh/return to it
 - if you want it to clear out all UI controls' state and values, you must do a full refresh
 - Firefox: Shift-Ctrl-R
 - Mac: Shift-Command-R

CSC443: Web Programming

Styling form controls

6

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

CSS

- ❑ attribute selector: matches only elements that have a particular attribute value
- ❑ useful for controls because many share the same element (input)

CSC443: Web Programming

Styling form controls

7

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

CSS



CSC443: Web Programming

Hidden input parameters

8

```
<input type="text" name="username" /> Name <br />  
<input type="text" name="sid" /> SID <br />  
<input type="hidden" name="school" value="UW" />  
<input type="hidden" name="year" value="2048" />
```

HTML

- ❑ an invisible parameter that is still passed to the server when the form is submitted
- ❑ useful for passing on additional state that isn't modified by the user
- ❑ A hidden field often stores a default value, or can have its value changed by a JavaScript

CSC443: Web Programming

HTML <input> type Attribute

9

- HTML5 has the following new input types:
 - ▣ **color**: Defines a color picker
 - ▣ **date**: Defines a date control (year, month and day)
 - ▣ **datetime-local**: Defines a date and time control
 - ▣ **month**: Defines a month and year control
 - ▣ **week**: Defines a week and year control
 - ▣ **Time**: Defines a control for entering a time
 - ▣ **email**: Defines a field for an e-mail address
 - ▣ number
 - ▣ **range**: Defines a control for entering a number whose exact value is not important
 - ▣ **search**: Defines a text field for entering a search string
 - ▣ **tel**: Defines a field for entering a telephone number
 - ▣ **url**: Defines a field for entering a URL

CSC443: Web Programming

HTML <input> type Attribute

10

```
<form action="demo_form.asp"> Search Google: <input type="search"
name="googlesearch"><br> <input type="submit"></form>
```

```
<form action="demo_form.asp">Points: <input type="range"
name="points" min="0" max="10"><input type="submit"></form>
```

CSC443: Web Programming

11

Submitting data

CSC443: Web Programming

Problems with submitting data

12

```
<form action="http://localhost/test1.php" method="get">
<label><input type="radio" name="cc" /> Visa</label>
<label><input type="radio" name="cc" /> MasterCard</label>
<br />
Favorite Star Trek captain:
<select name="startrek">
  <option>James T. Kirk</option>
  <option>Jean-Luc Picard</option>
</select> <br />
</form>
```

HTML

- the form may look correct, but when you submit it...
 - ▣ `[cc] => on`, `[startrek] => Jean-Luc Picard`
- How can we resolve this conflict?

Recall: The name attribute is used to reference elements in a JavaScript, or to reference form data after a form is submitted.

CSC443: Web Programming

The value attribute

13

```
<label><input type="radio" name="cc" value="visa" />
Visa</label>
<label><input type="radio" name="cc" 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>
<input type="submit" value="submit" />
</select> <br />
```

HTML

- value attribute sets what will be submitted if a control is selected
- [cc] => visa, [startrek] => picard

CSC443: Web Programming

Submitting data to a web server

15

- though browsers mostly retrieve data, sometimes you want to submit data to a server
 - Hotmail: Send a message
 - Flickr: Upload a photo
 - Google Calendar: Create an appointment
- the data is sent in HTTP requests to the server
 - with HTML forms
 - with **Ajax** (seen later)
- the data is placed into the request as parameters

CSC443: Web Programming

URL-encoding

14

- certain characters are not allowed in URL query parameters:
 - examples: " ", "/", "=", "&"
- when passing a parameter, it is URL-encoded
 - "Xenia's cool!?" → "Xenia%27s+cool%3F%21"
- you don't usually need to worry about this:
 - the browser automatically encodes parameters before sending them
 - the PHP \$_REQUEST array automatically decodes them
 - ... but occasionally the encoded version does pop up (e.g. in Firebug)

CSC443: Web Programming

HTTP GET vs. POST requests

16

- GET : asks a server for a page or data
 - if the request has parameters, they are sent in the URL as a query string
- POST : submits data to a web server and retrieves the server's response
 - if the request has parameters, they are embedded in the request's HTTP packet, not the URL

CSC443: Web Programming

HTTP GET vs. POST requests

17

- For submitting data, a POST request is more appropriate than a GET
 - GET requests embed their parameters in their URLs
 - URLs are limited in length (~ 1024 characters)
 - URLs cannot contain special characters without encoding
 - private data in a URL can be seen or modified by users

CSC443: Web Programming

Form POST example

18

```
<form action="http://localhost/app.php" method="post">
<div>
    Name: <input type="text" name="name" /> <br />
    Food: <input type="text" name="meal" /> <br />
    <label>Meat? <input type="checkbox" name="meat" /></label>
<br />
    <input type="submit" />
</div>
</form>
```

HTML

CSC443: Web Programming

GET or POST?

19

```
if ($_SERVER["REQUEST_METHOD"] == "GET") {
    # process a GET request
    ...
} elseif ($_SERVER["REQUEST_METHOD"] == "POST") {
    # process a POST request
    ...
}
```

PHP

- some PHP pages process both GET and POST requests
- to find out which kind of request we are currently processing, look at the global `$_SERVER` array's "REQUEST_METHOD" element

CSC443: Web Programming

Uploading files

20

```
<form action="http://webster.cs.washington.edu/params.php"
method="post" enctype="multipart/form-data">
    Upload an image as your avatar:
    <input type="file" name="avatar" />
    <input type="submit" />
</form>
```

HTML

- add a file upload to your form as an input tag with type of file
- must also set the `enctype` attribute of the form

CSC443: Web Programming

Processing form data in PHP

"Superglobal" arrays

Array	Description
<u>\$_REQUEST</u>	parameters passed to any type of request
<u>\$_GET</u> , <u>\$_POST</u>	parameters passed to GET and POST requests
<u>\$_SERVER</u> , <u>\$_ENV</u>	information about the web server
<u>\$_FILES</u>	files uploaded with the web request
<u>\$_SESSION</u> , <u>\$_COOKIE</u>	"cookies" used to identify the user (seen later)

- PHP superglobal arrays contain information about the current request, server, etc.
- These are special kinds of arrays called associative arrays.

Associative arrays

```
$blackbook = array();
$blackbook["xenia"] = "206-685-2181";
$blackbook["anne"] = "206-685-9138";
...
print "Xenia's number is " . $blackbook["xenia"] . ".\n";
```

PHP

- associative array (a.k.a. map, dictionary, hash table) : uses non-integer indexes
- associates a particular index "key" with a value
 - ▣ key "xenia" maps to value "206-685-2181"

Example: exponents

```
<?php
    $base = $_REQUEST["base"];
    $exp = $_REQUEST["exponent"];
    $result = pow($base, $exp);
?>
<?= $base ?> ^ <?= $exp ?> = <?= $result ?>
```

PHP

- What should we do to run this with xampp?

Example: Print all parameters

25

```
<?php
foreach ($_REQUEST as $param => $value) {
    ?>
    <p>Parameter <?= $param ?> has value <?= $value ?></p>
    <?php
}
?>
```

PHP

- What should we do to run this with xampp?

CSC443: Web Programming

Uploading files

27

```
<input type="file" name="avatar" />
```

HTML

- example: if you upload toby.jpg as a parameter named avatar,
 - `$_FILES["avatar"]["name"]` will be "tobby.jpg"
 - `$_FILES["avatar"]["type"]` will be "image/jpeg"
 - `$_FILES["avatar"]["tmp_name"]` will be something like "/var/tmp/phpZtR4TI"

CSC443: Web Programming

Processing an uploaded file in PHP

26

- uploaded files are placed into global array `$_FILES`, not `$_REQUEST`
- each element of `$_FILES` is itself an associative array, containing:
 - name: the local filename that the user uploaded
 - type: the MIME type of data that was uploaded, such as image/jpeg
 - size : file's size in bytes
 - 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

CSC443: Web Programming

```
Array
(
    [file1] => Array
        (
            [name] => MyFile.txt (comes from the browser, so
            treat as tainted)
            [type] => text/plain (not sure where it gets this
            from - assume the browser, so treat as tainted)
            [tmp_name] => /tmp/php/php1h4j1o (could be anywhere
            on your system, depending on your config settings, but the user
            has no control, so this isn't tainted)
            [error] => UPLOAD_ERR_OK (= 0)
            [size] => 123 (the size in bytes)
        )
    [file2] => Array
        (
            [name] => MyFile.jpg
            [type] => image/jpeg
            [tmp_name] => /tmp/php/php6hst32
            [error] => UPLOAD_ERR_OK
            [size] => 98174
        )
)
```

PHP

28

CSC443: Web Programming

Processing uploaded file example

29

```
$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 {
    print "Error: required file not uploaded";
}
```

PHP

- 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

CSC443: Web Programming

Including files: include

30

```
include("header.php");
```

PHP

- inserts the entire contents of the given file into the PHP script's output page
- encourages modularity
- useful for defining reused functions needed by multiple pages

CSC443: Web Programming

31

Form Validation

CSC443: Web Programming

What is form validation?

32

- **validation:** ensuring that form's values are correct
- some types of validation:
 - preventing blank values (email address)
 - ensuring the type of values
 - integer, real number, currency, phone number, Social Security number, postal
 - address, email address, date, credit card number, ...
 - ensuring the format and range of values (ZIP code must be a 5-digit integer)
 - ensuring that values fit together (user types email twice, and the two must match)

CSC443: Web Programming

A real Form that uses validation

The screenshot shows a web browser window with the WeightWatchers.com registration page. The form is titled "Create Your Registered User Account Login". It contains several input fields: "First name" (filled with "Xenia"), "Last name" (empty), "Your birthdate" (Month, Day, Year dropdowns), "Your gender" (radio buttons for Female and Male), "State" (dropdown menu), "Zip code" (text input), "E-mail" (text input), and "Re-enter e-mail" (text input). Red error messages are displayed next to the "Last name", "Birthdate", "Gender", "State", "Zip code", "E-mail", and "Re-enter e-mail" fields, indicating that the information is required or invalid. A sidebar on the left contains links like "Become a Registered User!", "Start a blog", "Participate in a Challenge", "Save your favorite recipes", and "Post to our boards".

CSC443: Web Programming

Client vs. server-side validation

- Validation can be performed:
 - client-side** (before the form is submitted)
 - can lead to a better user experience, but not secure (why not?)
 - server-side** (in PHP code, after the form is submitted)
 - needed for truly secure validation, but slower
 - both**
 - best mix of convenience and security, but requires most effort to program**

CSC443: Web Programming

An example form to be validated

```
<form action="http://foo.com/foo.php" method="get">
  <div>
    City: <input name="city" /> <br />
    State: <input name="state" size="2" maxlength="2" />
  <br />
    ZIP: <input name="zip" size="5" maxlength="5" /> <br />
    <input type="submit" />
  </div>
</form>
```

HTML

- Let's validate this form's data on the server...

CSC443: Web Programming

Basic server-side validation code

```
...
$city = $_REQUEST["city"];
$state = $_REQUEST["state"];
$zip = $_REQUEST["zip"];
if (!$city || strlen($state) != 2 || strlen($zip) != 5) {
  ?>
    <h2>Error, invalid city/state submitted.</h2>
  <?php
  }
  ?>
```

PHP

- basic idea: examine parameter values, and if they are bad, show an error message and abort

CSC443: Web Programming

Basic server-side validation code

37

- validation code can take a lot of time / lines to write
 - How do you test for integers vs. real numbers vs. strings?
 - How do you test for a valid credit card number?
 - How do you test that a person's name has a middle initial?
 - How do you test whether a given string matches a particular complex format?

CSC443: Web Programming

Delimiters

39

```
[a-z]at           #cat, rat, bat...
[aeiou]#
[a-zA-Z]/
~[^a-z]~          #not a-z
[[:alnum:]]+       #at least one alphanumeric char
#(very) *#large    #large, very very very large...
~(very){1, 3}~     #counting "very" up to 3
/^bob/            #bob at the beginning
/com$/            #com at the end

/http://\
// #http://#      #better readability
```

PHPRegExp

- Used for Perl regular expressions (preg)

CSC443: Web Programming

Regular expressions

38

```
[a-z]at           #cat, rat, bat...
[aeiou]
[a-zA-Z]
[^a-z]            #not a-z
[[:alnum:]]+       #at least one alphanumeric char
(very) *large     #large, very very very large...
(very){1, 3}      #counting "very" up to 3
^bob              #bob at the beginning
com$              #com at the end
```

PHPRegExp

- Regular expression: a pattern in a piece of text
- PHP has:
 - POSIX
 - Perl regular expressions

CSC443: Web Programming

Basic Regular Expression

40

```
/abc/
```

- in PHP, regexes are strings that begin and end with /
- the simplest regexes simply match a particular substring
- the above regular expression matches any string containing "abc":
 - YES: "abc", "abcdef", "defabc", "=".abc.="", ...
 - NO: "fedcba", "ab c", "PHP", ...

CSC443: Web Programming

Wildcards

41

- A dot `.` matches any character except a `\n` line break
 - `"/.oo.y/"` matches "Doocy", "goofy", "LooNy", ...
- A trailing `i` at the end of a regex (after the closing `/`) signifies a case-insensitive match
 - `"/xen/i"` matches "Xenia", "xenophobic", "Xena the warrior princess", "XEN technologies" ...

CSC443: Web Programming

Special characters: `|`, `()`, `^`, `\`

42

- `|` means OR
 - `"/abc|def|g/"` matches "abc", "def", or "g"
 - There's no AND symbol. Why not?
- `()` are for grouping
 - `"/(Homer|Marge) Simpson/"` matches "Homer Simpson" or "Marge Simpson"
- `^` matches the beginning of a line; `$` the end
 - `"/^<!--$/"` matches a line that consists entirely of "<!--"

CSC443: Web Programming

Special characters: `|`, `()`, `^`, `\`

43

- `\` starts an escape sequence
 - many characters must be escaped to match them literally: `/ \ $. [] () ^ * + ?`
 - `"/<br \>/"` matches lines containing `
` tags

CSC443: Web Programming

Quantifiers: `*`, `+`, `?`

44

- `*` 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!?_a", ...
- `+` means 1 or more occurrences
 - `"/a(bc)+/"` matches "abc", "abcbc", "abcbcbc", ...
 - `"/Goo+gle/"` matches "Google", "Gooogle", "Gooooogle", ...
- `?` means 0 or 1 occurrences
 - `"/a(bc)?/"` matches "a" or "abc"

CSC443: Web Programming

More quantifiers: {min,max}

45

- {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

CSC443: Web Programming

Character ranges: [start-end]

47

- 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

CSC443: Web Programming

Character sets: []

46

- [] 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)?

CSC443: Web Programming

Character ranges: [start-end]

48

- 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- ?

CSC443: Web Programming

Escape sequences

49

- 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 ?

CSC443: Web Programming

Regular expressions example

51

```
echo preg_match ('/test/', "a test of preg_match");
echo preg_match ('/tutorial/', "a test of preg_match");

$matchesarray[0] = "http://www.tipsntutorials.com/"
$matchesarray[1] = "http://"
$matchesarray[2] = "www.tipsntutorials.com/"
preg_match ('/(http://)(.*)/', "http://www.tipsntutorials.com/", $matchesarray)
```

PHP

CSC443: Web Programming

Regular expressions in PHP (PDF)

50

- regex syntax: strings that begin and end with `/`, such as `"/[AEIOU]+/"`

function	description
<code>preg_match(regex, string)</code>	returns TRUE if string matches regex
<code>preg_replace(regex, replacement, string)</code>	returns a new string with all substrings that match regex replaced by replacement
<code>preg_split(regex, string)</code>	returns an array of strings from given string broken apart using the given regex as the delimiter (similar to explode but more powerful)

CSC443: Web Programming

Regular expressions example

52

```
# replace vowels with stars
$str = "the quick brown fox";
$str = preg_replace("/[aeiou]/", "*", $str);
# "th* q**ck br*wn f*x"
# break apart into words
$words = preg_split("/[ ]+/", $str);
# ("th*", "q**ck", "br*wn", "f*x")
# capitalize words that had 2+ consecutive vowels
for ($i = 0; $i < count($words); $i++) {
    if (preg_match("/\\*{2,}/", $words[$i])) {
        $words[$i] = strtoupper($words[$i]);
    }
}
# ("th*", "Q**CK", "br*wn", "f*x")
```

PHP

CSC443: Web Programming

PHP form validation w/ regexes

53

```
$state = $_REQUEST["state"];  
if (!preg_match("/[A-Z]{2}/", $state)) {  
    ?>  
    <h2>Error, invalid state submitted.</h2>  
    <?php  
}
```

PHP

- using `preg_match` and well-chosen regexes allows you to quickly validate query parameters against complex patterns

Another PHP experiment

54

- Write a PHP script that tests whether an e-mail address is input correctly. Test using valid and invalid addresses
- Use array
- Use function