Week 7: Midterm Review

CIS 086 • PHP and MySQL • Mission College

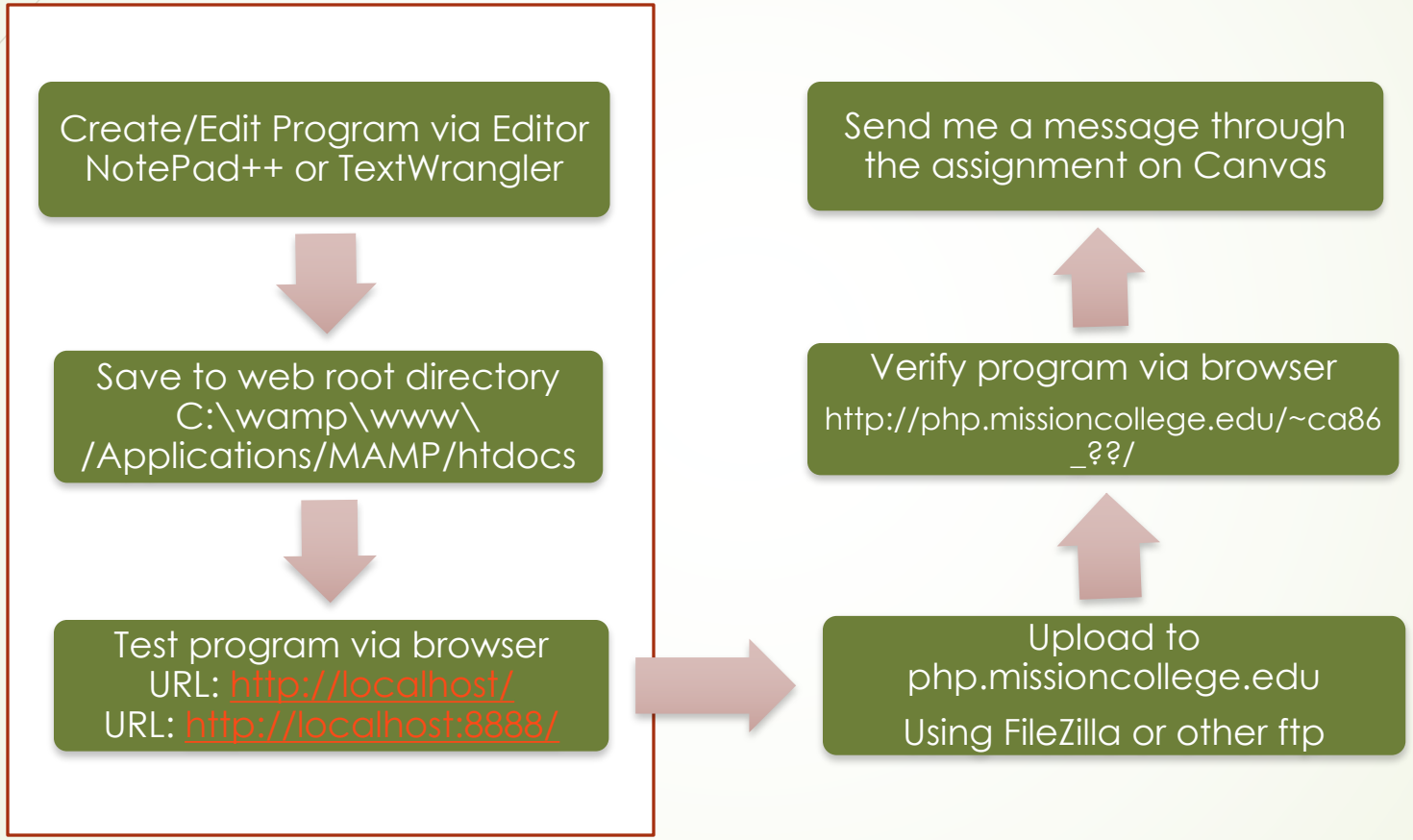
Midterm exam

- Two parts:
  - Multiple choice part
  - Coding part
- Multiple choice part
  - On Canvas, scored automatically
- Coding part
  1. Write the code to process a form
  2. Post the form on the Mission College PHP server
  3. Link the form to your home page

Homework Protocol

1. Upload all assigned exercises and projects to the Mission College server.
2. Make sure to test after you upload!
3. Link all exercises and projects to your personal home page.
4. Do not upload any files or post file attachments in the drop box.

Homework Flow Chart



Static vs Dynamic Web Page

- A **static web page** is a web page that is delivered to the user exactly as stored. Consequently a static web page displays the same information for all users, from all contexts.
- A **dynamic web page** is a kind of web page that has been prepared with fresh information (content and/or layout), for each individual viewing. It is not static because it changes with the time, the user, the user interaction, the context, or any combination of the above.

Programming Topics

Basic Topics

- Numbers
- Data types
- Operators
  - Math operators
  - Logic operators
- Strings
- Expressions
- Statements

Intermediate Topics

- Conditionals
  - If
  - Switch
- Loops
  - For
  - While
- Arrays
- Strings
- Files
- Functions

Images

- Use images of the appropriate size (in both pixels and bytes).
- Use an image format that is suitable for the content you're displaying.
- Use images that load quickly.
- When resizing images, preserve the aspect ratio.

Basic HTML

- A complete basic web page **must** have these elements:
  - DOCTYPE
  - HTML tag
  - HEAD tag
  - charset
  - TITLE tag
  - BODY tag
- Any page that does not have these basic tags will not validate.
- See the blog for a quick template. http://cis086.blogspot.com

Tags you should learn

IMG	Image
A	Link to another page
TABLE, TR, TH, TD	Table, row, header, data
UL	Unordered (bullet) list
OL	Ordered (numbered) list
LI	List item

- Combining A with IMG and LI
  - You can put an A tag inside an LI tag (a list of links)
  - You can put an IMG tag inside an A tag (image link)

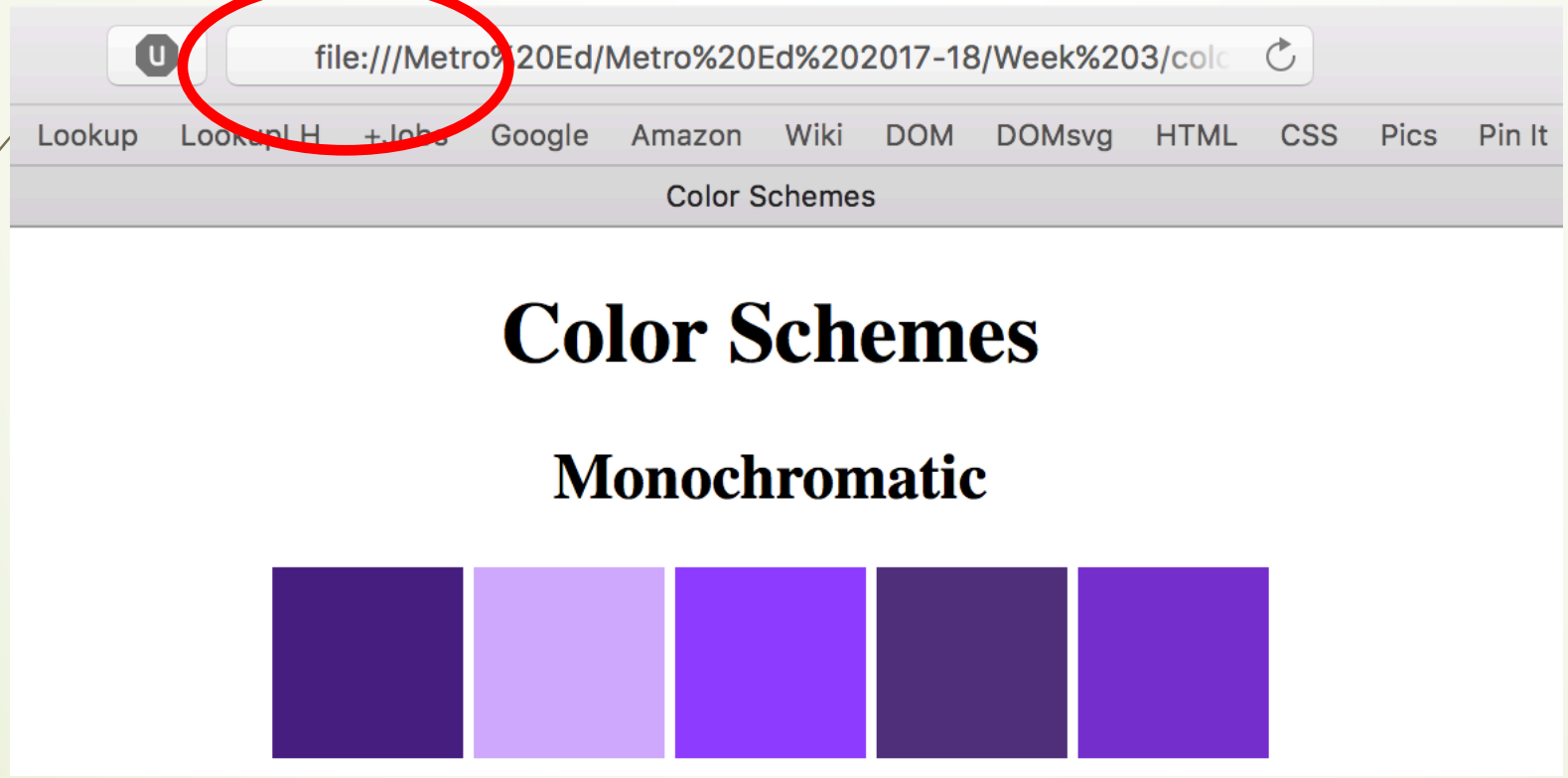
Basic CSS

- CSS = Cascading Style Sheets
- selector { attribute: value; }
- Six elements of a style declaration. All are required.

selector	Which HTML elements will this style apply to
curly braces	{ }
attribute	Which kind of style do you want to set, such as margin, padding, font-family, background-color, color
colon	: aka “eyeballs”
value	What do you want to set the style to, such as 8px, 1em, “Century Gothic”, red, chartreuse, etc.
semicolon	; aka “winky-face”

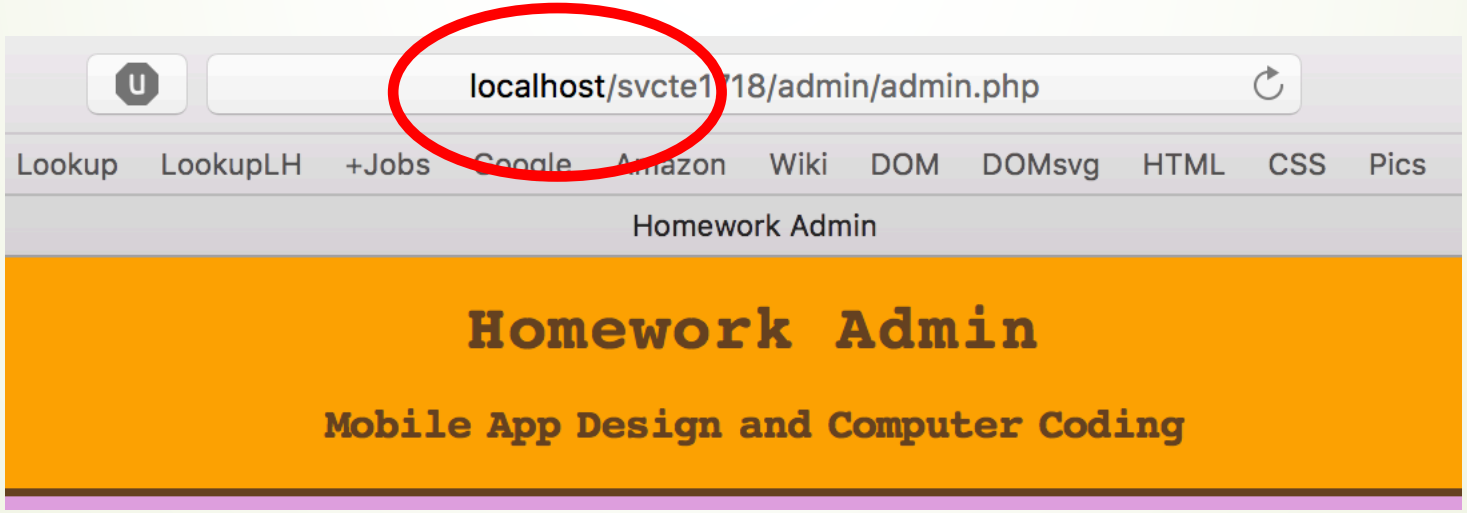
File Protocol

- You can double click on a pure HTML file and open it in the browser. This uses the file:// protocol.



HTTP Protocol

- Opening a PHP file in the browser is more difficult. You must:
  1. Start the xAMP server.
  2. Put the file in the www or htdocs folder.
  3. Navigate in the URL bar to the localhost.
  4. Find your file and click on it.
  5. The browser used to say http:// but now it just says localhost.



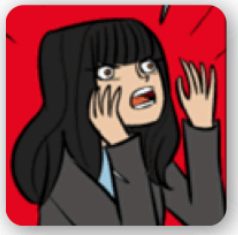
Example: 12 Images


Dina


Claire


Dorothy


Dale


Lottie


Marigold


Sonny


Hannelore


Mildred

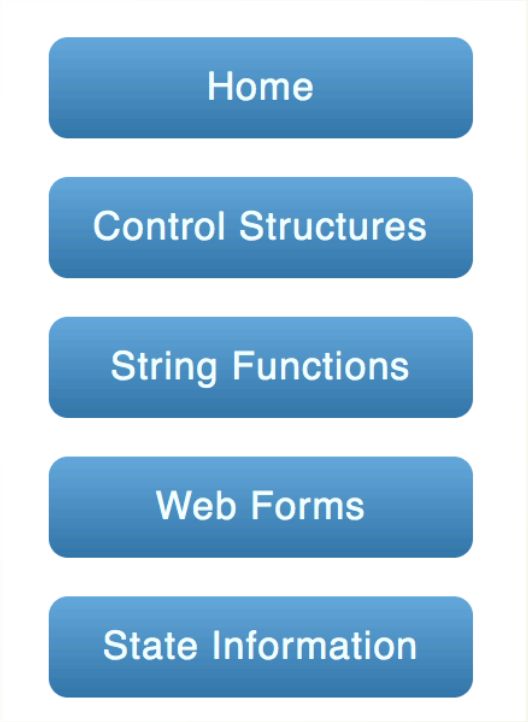

Walky


Joyce


Shauna

Example: Buttons

- You can make the buttons using Photoshop.
- You can make the buttons using CSS by applying a shadow, linear gradient, hover effect, or other.
- The buttons should all be the same width and height, and use the same font size, even if the shorter text items do not fill the button.



PHP Code Blocks

PHP code blocks can go directly inside HTML code.

```
<?php
  echo “I wrote this in code block one.\n”;
?>
<p>This paragraph is HTML.</p>
<?php
  echo “I wrote this in code block two.\n”;
?>
```

Comments

```
// This one line is commented out


/*
  This whole block is commented out.
  The comment can span many lines.
  Fourscore and seven years ago.
 */
```

Constants

```
define (“FAVORITE_LANGUAGE”, “PHP”);

echo “My favorite programming language is “ .
    FAVORITE_LANGUAGE;
```

PHP Language Constructs

- echo
- print
- include
- require
- isset
- Language constructs don't need parentheses (but you may use them; they are not prohibited either)
- http://www.phpknowhow.com/basics/language-constructs-vs-built-in-functions/

Commonly-used functions

- `print_r ()`
- `array ()`
- `file ()`
- `count ()`
- `date ()`
- `substr ()`
- `header ()`
- `strlen ()`
- `str_replace()`
- `printf ()`
- `trim ()`
- `sprintf ()`
- `strpos ()`
- `preg_match()`
- `min ()`
- `exec ()`

- http://www.boyter.org/2011/03/list-of-most-commonly-used-php-functions/
- http://php.net/manual/en/function.print-r.php

PHP Data Types

- Primitive or Scalar
  - Integer
  - Floating Point
  - Boolean
  - String
- Compound
  - Array
  - Object
- Special
  - Resource
  - NULL

- http://php.net/manual/en/language.types.intro.php

Operator Precedence

```
$a = 5;
$b = 6;
$x = 7;
$y = 8;
$z = $a * $x + $b * $y; // z = ax + by
print_r ($z);
```

- First : * (both of them)
- Second : +
- Last : =

Logical Operator Precedence

```
$a = true;
$b = true;
$x = true;
$y = false;
$z = $a && $x || $b && $y;
print_r ($z);
```

- First : && (both of them)
- Second : | |
- Last : =

Associativity

- **Left to Right**
  - `$a + $b + $c + $d; //` is the same as
  - `(($a + $b) + $c) + $d;`

  - `$a – $b – $c – $d; //` is the same as
  - `(($a – $b) – $c) – $d;`
  - `// BUT NOT the same as`
  - `$a – ($b – ($c – $d));`

- **Right to Left**
  - `$x = $y = $z; // is the same as`
  - `$y = $z; // THEN`
  - `$x = $y;`

Increment and Decrement

```
$x = 3;
$y = 2;

$x = $y++; // POST-increment is the same as
$x = y; // THEN
$y = $y + 1;

$x = ++$y; // PRE-increment is the same as
$y = $y + 1; // THEN
$x = $y;
```

Conditional Operator

```
$BlackjackPlayer1 = 20;
$Result = "Player 1 is " .
    ($BlackjackPlayer1 <= 21 ?
        "still in the game." :
        "out of the action.");
echo "<p>", $Result, "</p>";
```

- Prototype:
- condition ? value_if_true : value_if_false
- value_if_true and value_if_false may be full statements but they may also be simply expressions

Variable names

```
// Must start with a letter or underscore
// May use letters, underscores, and numbers
// http://php.net/manual/en/language.variables.basics.php

$xyz123 = 17; // ok to start with a letter and have number
$_startsWithUnderscore = 42; // ok to start w/underscore

$123startsWithNumber = "xyz"; // NO, cannot start w/number

$inside_underscore = "Can I do it?"; // ok underscores
$double__underscore = "Really"? // ok to have 2 underscore

$embedded space = 3.14e2; // NO cannot have space inside
$"embedded space" = 3.14e2; // NO this does not help
```

Arrays

```
$cities = array (
    "Alviso", "Belmont", "Campbell",
    "Danville", ”Eureka", "Fremont",
    "Gilroy", "Hayward", ”Irvine",
    "Jackson” );
print_r ($cities);
```

Bracket Syntax

(PHP 5.4+ only)

```
$countySeats = [
    [ "Alameda", "Oakland"],
    [ "Alpine", "Markleeville" ],
    [ "Amador", "Jackson" ],
    [ "Butte", "Oroville" ],
    [ "Calaveras", "San Andreas" ],
    [ "Colusa", "Colusa" ],
    [ "Contra Costa", "Martinez" ],
    [ "Del Norte", "Crescent City" ] ];
print_r ($countySeats);
```

Functions

```
$numberOfCities = count ($cities);

echo $numberOfCities;
```

Variable Scope

- A variable declared inside a function is visible only within that function.
- A variable that is a function parameter is visible only within that function.
- Global variables, declared outside any function, are visible outside all functions.
- Global variables can be made visible within a function by using the keyword *global*.

Global variable example

```
$languages = [ “php”, “python”, “perl”, “pascal”,
    “prolog” ];

function print_languages () {
    global $languages;

    for ($i=0; $i<count($languages); $i++) {
        echo $languages . “<br />\n”;
    }
}
```

Superglobals

Basic information	$_SERVER
Forms	$_GET $_POST
Uploading files	$_FILES
State information	$_COOKIE $_SESSION

Relational Operators

Operator	What it does
==	Equal value
===	Equal value and type
!=	Not equal value
!==	Not equal value or not the same type
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Logical Operators

Operator	Name	What it does
&&	AND	True if both operands are true
||	OR	True if either operand is true
XOR	XOR	True if one and only one of the operands is true
!	NOT	True if its operand is false

Conditionals

```
if (a) {
  do_stuff_b();
}
else if (c) {
  do_stuff_d();
}
```

- `do_stuff_b()` happens only if `a` is true.
- `do_stuff_d()` happens only if `a` is false and `c` is true.

SWITCH statements

```
switch ($colorName)
{
  case "red"     : $colorHex = "ff0000"; break;
  case "orange"  : $colorHex = "ff8000"; break;
  case "yellow"  : $colorHex = "ffff00"; break;
  case "green"   : $colorHex = "00ff00"; break;
  case "cyan"    : $colorHex = "00ffff"; break;
  case "blue"    : $colorHex = "0000ff"; break;
  case "magenta" : $colorHex = "ff00ff"; break;
  case "violet"  : $colorHex = "8000ff"; break;
  default        : $colorHex = "000000"; break;
}
```

Loops

- **for**
  - When you know how many times you want to go through the loop.
  - Example: printing 12 icons onto the web page.
- **while**
  - When you don't know how many times you'll go through the loop, but you have a way to know when you're finished.
  - For example: reading from a file. You don't know how many lines the file has, but you can detect an EOF marker.
  - Not guaranteed to ever go through the loop.
- **do…while**
  - Same as while, but guaranteed to go through the loop at least once.

Loop equivalence

```
for ($i=0; $i<12; $i++) {
    echo $i . “<br />\n”;
}
```

```
$i = 0;
do {
    echo $i . “<br />\n”;
    $i++;
}
  while ($i<12);
```

```
$i = 0;
while ($i<12) {
    echo $i . “<br />\n”;
    $i++;
}
```

Pass by value vs. reference

- When a variable is passed by value, the function makes a local copy of the variable and uses the local copy for all calculations. When the function returns, the local copy is thrown away, and any changes made to it are thrown away too.

- When a variable is passed by reference, the called function gets the actual variable to manipulate. This means it can both use and change the value of that variable. If the function changes the variable, after the variable is returned, the calling function will see the changed value.

Pass by value vs. reference

```
// Example: Swap 2 values
// If passed by value, this will not work.

function swap (&$value1, &$value2)
{
  $temp = $value1;
  $value1 = $value2;
  $value2 = $temp;
}

if ($colors[$i] > $colors[$i+1])
  swap ($colors[$i], $colors[$i+1]);
```

What is a string?

- A string is a sequence of characters.
- A character is a grapheme or symbol that is part of a written language.
- Examples of characters include numbers, letters, punctuation, white space, and control characters.
- White space includes spaces, hard spaces, tabs, and vertical tabs.
- Control characters include carriage return, line feed, and others.

Commonly-used string functions

- explode
- htmlentities
- htmlspecialchars
- implode
- join
- lcfirst
- levenshtein
- money_format
- number_format
- printf
- similar_text
- sprintf
- str_replace
- strcasecmp
- strchr
- strcmp
- strip_tags
- stripos
- stristr
- stripslashes
- stristr
- strlen
- strpos
- strrpos
- strtolower
- strtoupper
- substr
- trim
- ucfirst
- ucwords

Splitting a string into parts

- `explode == (split)`
- `implode == join`
- `strip_tags`
- `substr`
- `trim`
- `preg_split`

Comparing strings

- `levenshtein`
- `similar_text`
- `strcasecmp`
- `strcmp`

Searching within a string

- `str_replace`
- `strchr`
- `stristr`
- `strstr`
- `strpos`
- `strrpos`

Formatting a string

- lcfirst
- ucfirst
- ucwords
- printf
- sprintf

- money_format
- number_format
- strtolower
- strtoupper

similar_text

- Calculates the similarity between two strings.
- http://php.net/manual/en/function.similar-text.php

levenshtein

- Calculates the minimal number of characters you have to replace, insert or delete to transform one string into the other.
- http://php.net/manual/en/function.levenshtein.php

HTML Forms

- Required elements:
  - Opening and closing <form> tags
  - Method
    - GET
    - POST
  - Action: the URL of the form handler
  - Input fields

Forms

- Forms **must** include:
  - ACTION attribute – this tells which PHP file will process the form input.
  - METHOD attribute – this can be POST or GET.
- Form **may** include:
  - ID – which may be useful if you have 2 forms on one page.
  - CLASS – which can be used with CSS to style the form.

Form fields

Input field types

- Text boxes
- Checkboxes
- Radio buttons
- Hidden fields
- Submit button

Other field types:

- Text areas
- Select menu

New features in HTML5:

- placeholder
- color picker
- number
- range
- date and time pickers

Form Fields

- All form fields **must** have:
  - **NAME** – tells the form processor which variable goes with which value.
- Some form fields must have:
  - **VALUE** – tells the form processor what is the value of a checkbox, radio button, menu option, etc.

Autoglobals

- $_GET
  - Use $_GET when you want to GET information from the server: such as a specific web page, a specific inventory item, or a specific category
- $_POST
  - Use $_POST when you want to POST information to the server: such as a change to the database, or an uploaded file.
- It takes a long time to get a feel for the difference in usage. I try to give you tips.

GET vs. POST

- GET puts your form data in the URL string.
- **If** you don't want the form data to be **visible** in the URL string, **or** if there is **a lot** of form data, use POST instead of GET.
- If the form is going to **query** information from a database, GET is usually OK.
- But if the form is going to **change** information in a database, you always want to use POST instead.
- Think of the difference between **looking up** a book on Amazon and **ordering** the book on Amaaon.

GET vs. POST

- $_GET
  - Puts the parameters in the URL (**visible** to the user).
  - Lets you **bookmark** your URL.
  - You want users to be able to **bookmark** their favorite products or categories in your inventory.
  - www.myfavoritebookstore.com/?isbn=1234567890&category=programming
  - The user can add more parameters to the URL manually, which is a **security** flaw.

GET vs. POST

- $_POST
  - Passes the parameters in a separate data structure that is **not visible** to the browser or the URL.
  - You **cannot bookmark** your URL.
  - Won't let users bookmark URLs that cause change to the database or that upload files.
  - The user cannot add parameters because they can come only from the form, which partially closes the security problem.
  - However, a sophisticated user can create their own form that adds parameters. This is hard to automate, though.