Web Design and Programming

Week 7

6 June2024

Instructor: Dr. Peeraya Sripian

Course schedule

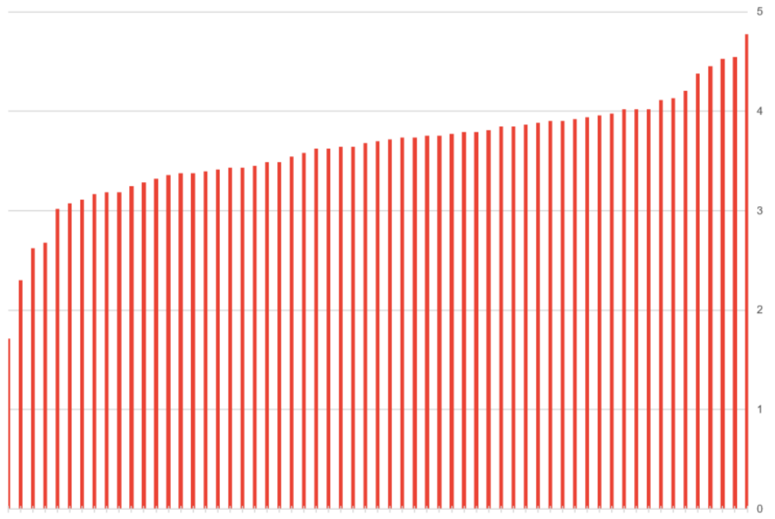
Week	Date	Topic
1	4/18	Intro to WWW, Intro to HTML
2	4/25	CSS Fundamental
	5/2	Holiday (GW)
3	5/9	CSS and Bootstrap
4	5/16	Work on midterm project
5	COIL 22 MAY 18:00-19:30 (counted as 1 class, replacing 23 May)	
6	5/30	Midterm project presentation week
7	6/6	PHP fundamentals + Installation XAMPP
8	6/13	PHP fundamentals 2
9	6/20	mySQL fundamentals
10	6/27	Assessing MySQL using PHP, MVC pattern + Intro of Final project
11	7/4	Cookies, sessions, and authentication + Proposal of final project
12	7/11	Javascript and PHP validation
13	7/18	Final project development
14	7/25	Final project presentation

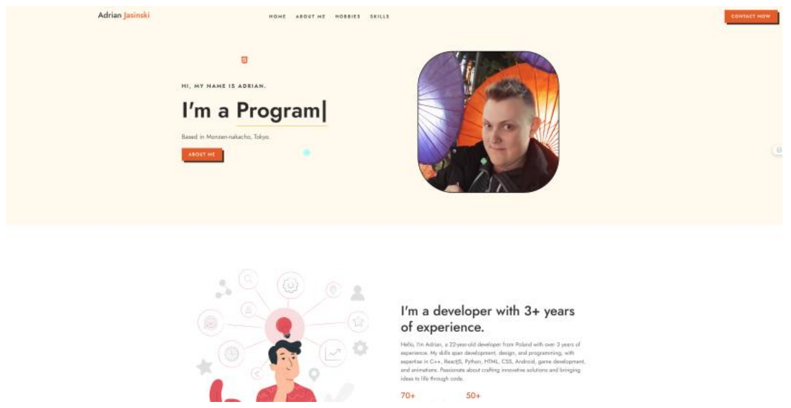
Midterm voting result

Top 3

ID	Score
Z123291	4.528302
Z524026	4.54717
Z124008	4.773585

Congratulations!!





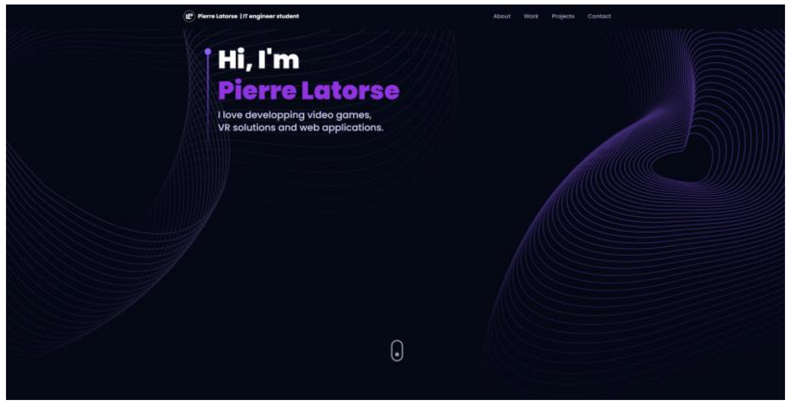
HI, MY NAME IS ADRIAN.

I'm a Program|

Based in Monzen-nakacho, Tokyo.

I'm a developer with 3+ years of experience.

Hello, I'm Adrian, a 22-year-old developer from Poland with over 3 years of experience. My skills span development, design, and programming, with expertise in C++, ReactJS, Python, HTML, CSS, Android, game development, and animations. Passionate about crafting innovative solutions and bringing ideas to life through code.



Hi, I'm Pierre Latorse

I love developping video games, VR solutions and web applications.

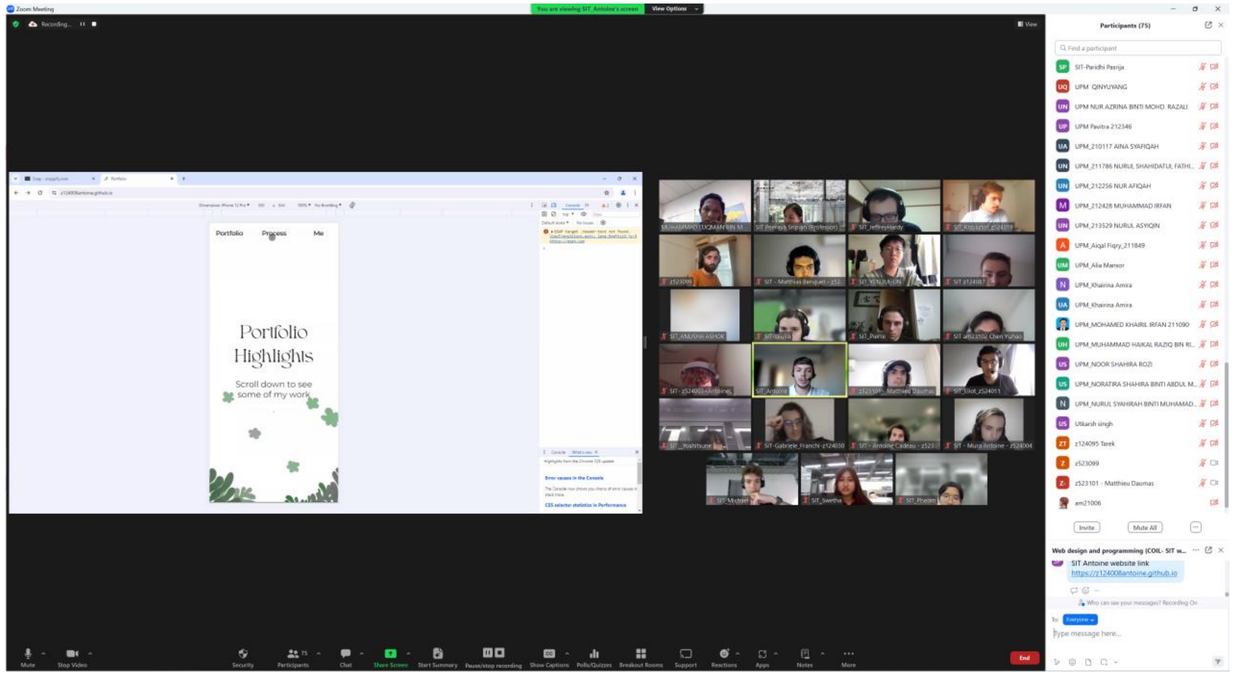
Portfolio Highlights

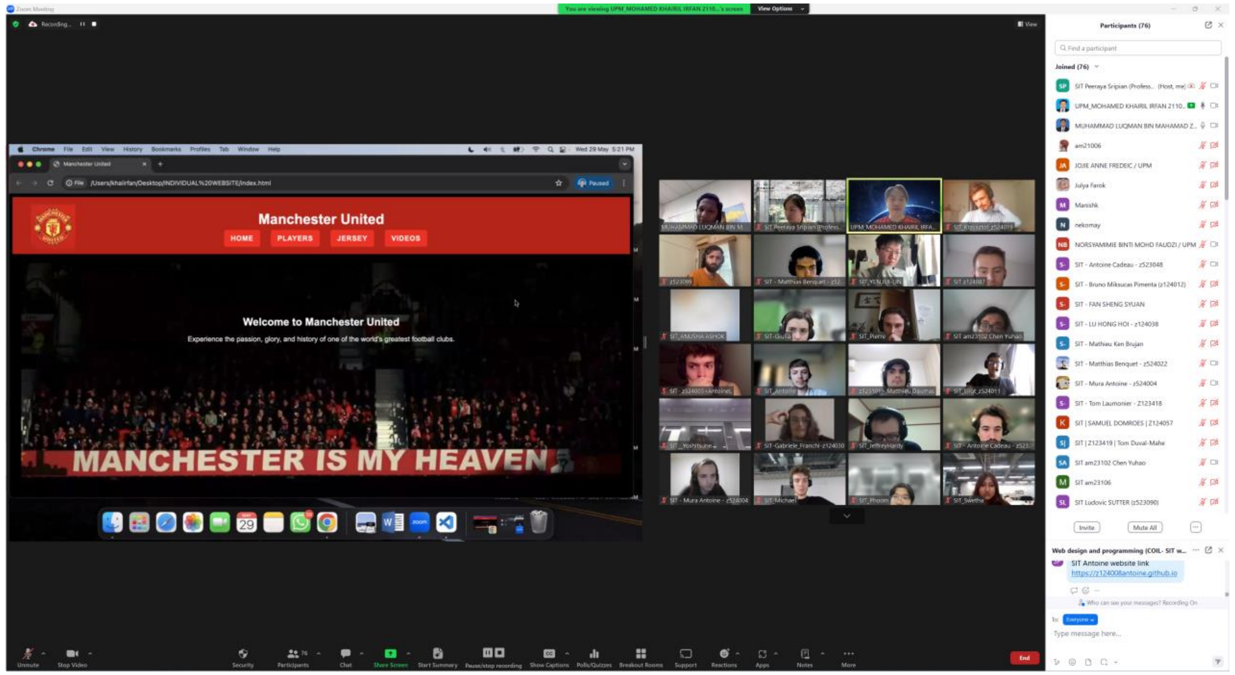
Scroll down to see some of my work

Result on COIL implementation

2 Professor (SIT, UPM)
33 UPM students
144 SIT students

Total: 179 participants




Manchester United
HOME
PLAYERS
JERSEY
VIDEOS
Welcome to Manchester United
Experience the passion, glory, and history of one of the world's greatest football clubs.
MANCHESTER IS MY HEAVEN
Participants (76)

Today's topic

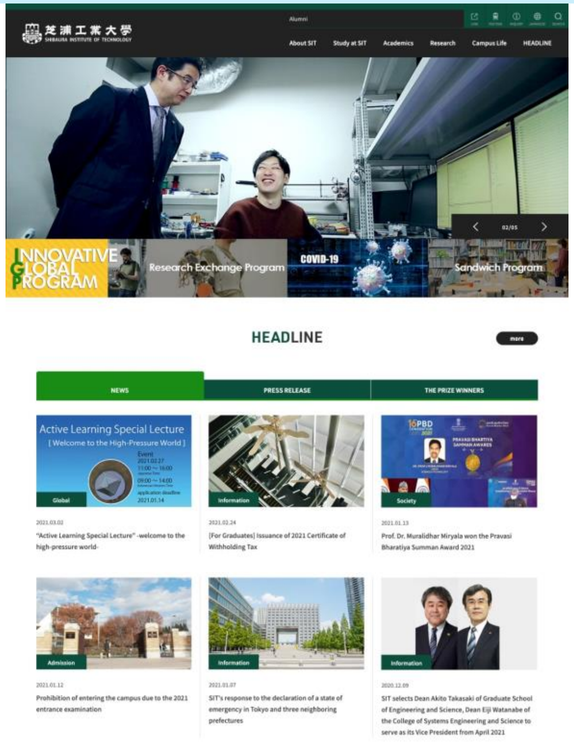
- Introduction to server side scripting
- PHP
  - Break for 10 minutes
- HTML Forms
- Submitting Form

- In class activity 6 – PHP Fundamentals

Two types of webpages

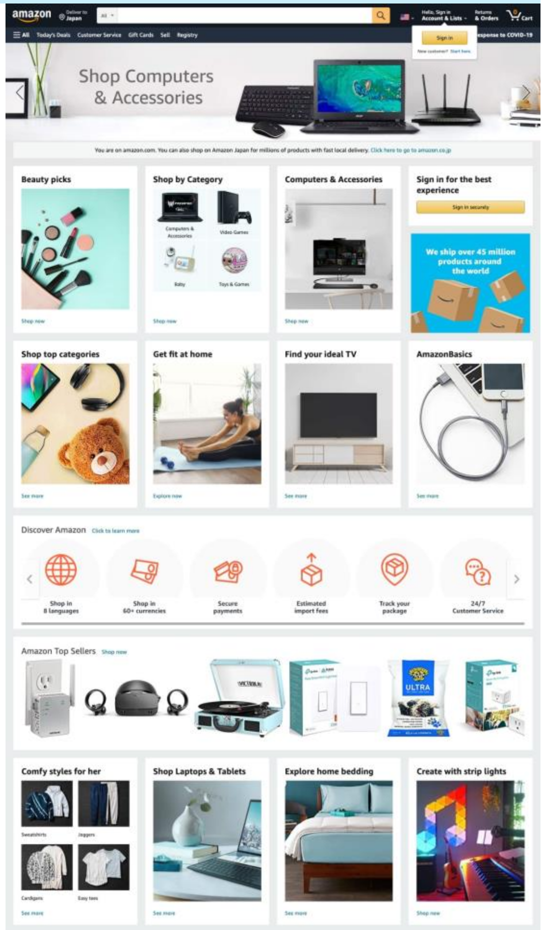
Static

https://www.shibaura-it.ac.jp/en/

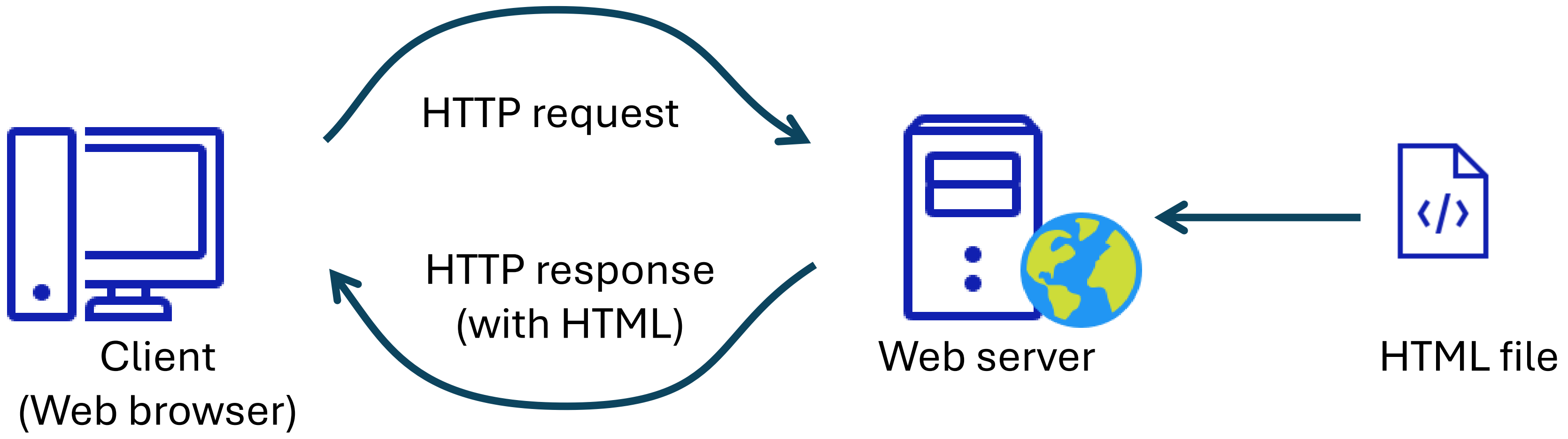


Dynamic

https://www.amazon.com/

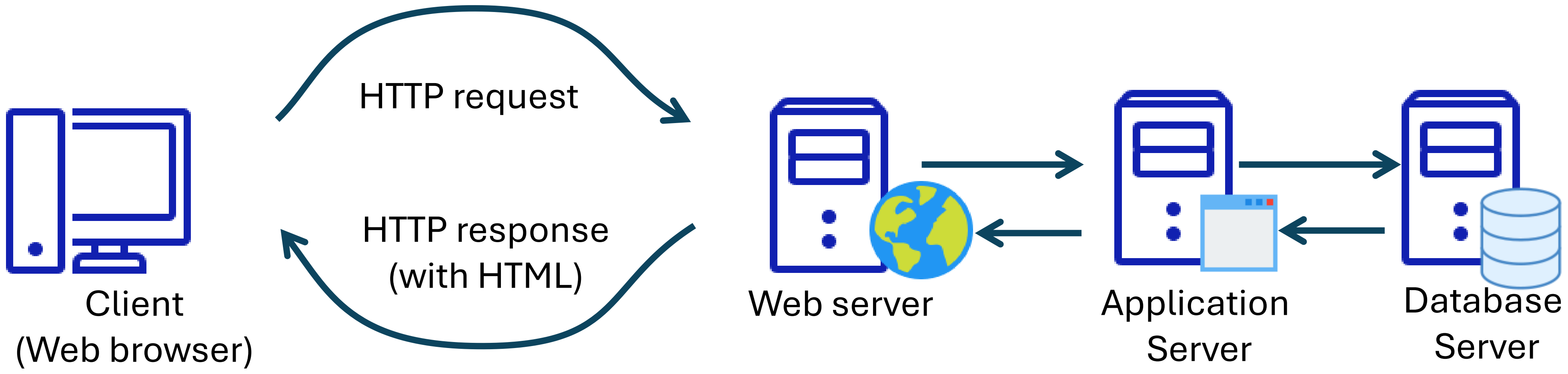


How static web pages are processed



- A static webpage is an HTML document stored on a web server and does not change
- Usually with filenames .htm .html
- When user request a static webpage, the browser send HTTP request include the filename
- Web server retrieve HTML from webpage and send it back to browser as part of HTTP response
- Browser then renders the HTML into a webpage and display

How dynamic web pages are processed



- A dynamic webpage is a webpage that is generated by a server-side program or script.
- A web server would looks up the **extension** of the requested file to find out which **application server** should process the request
- Application server runs the specified script. Often uses the data get from web browser to get the appropriate data from a **database server**.
- When application server finishes processing the data, it generates the **HTML** for a web page and returns to web server, then return to web browser as part of HTTP response.

Server-side scripting languages

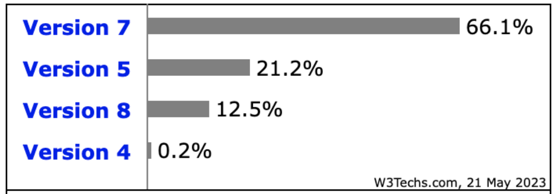
- To develop dynamic web pages, we need a server-side scripting language.
- Scripting languages run on specific web servers such as IIS and Apache.

Language	Description	Used by
Node.js	Quickest growing. JavaScript code on server-side. No need to learn a new language for back-end development. Good for real time application	Paypal, Uber, LinkedIn, Netflix
PHP	The most used (almost 80% of websites now). Good for content-based web pages. Embed in HTML code. Easy to write.	Facebook*, Wikipedia, Zoom
Java	Good for enterprise, performance.	Google, Amazon, eBay
Ruby	Developing applications fast. Adopted by a lot of startups.	Twitter (early days), Github, Airbnb
Python	Clear and easy to read syntax. Popular for statistical analysis.	Youtube, Instagram, Dropbox, Quora

https://twm.me/best-programming-languages-and-frameworks-for-server-side-web-development/

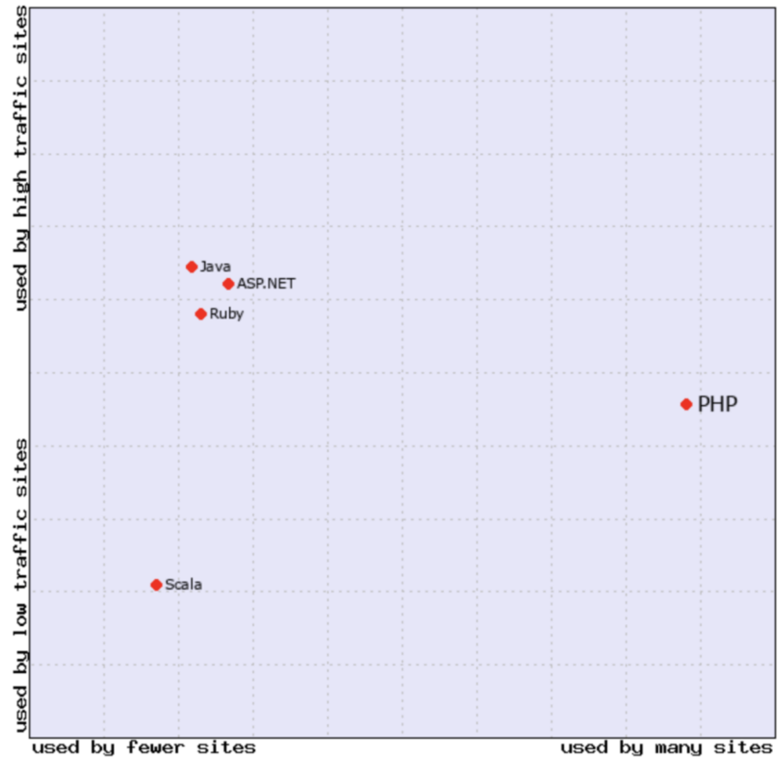
Server-side scripting languages usage

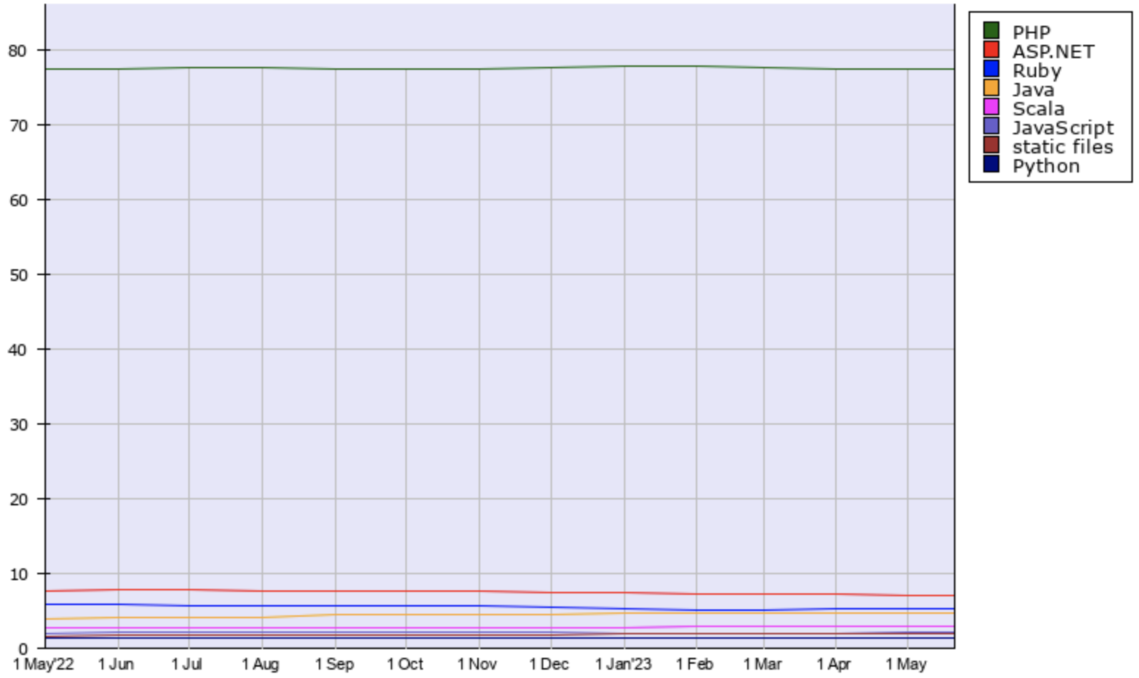
- Yes! PHP is not dead yet
- Although it is not the best or the most modern language
- PHP is used by 77.5% of all the websites whose server-side programming language we know



Percentages of websites using various versions of PHP

PHP Market Position, 21 May 2023, W3Techs.com



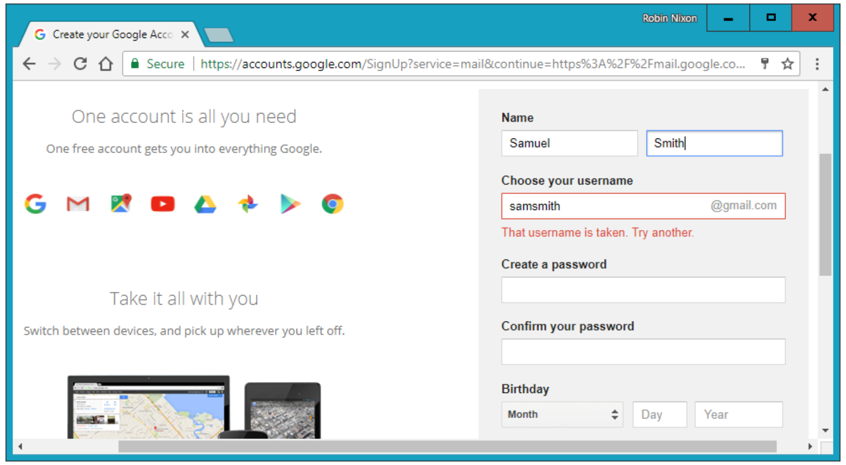


Usage of server-side programming languages for websites, 21 May 2023, W3Techs.com

https://w3techs.com/technologies/details/pl-php/all/all

Bringing it all together

- PHP, MySQL, JavaScript (sometimes aided by jQuery or etc.), CSS, and HTML all work together to produce “dynamic web content”
  - PHP handles all main work on web server
  - MySQL manages all the data
  - CSS + JavaScript looks after web page presentation
  - JavaScript → Talk to PHP code whenever it need to update something



Create your Google Acco

Robin Nixon

Secure | https://accounts.google.com/SignUp?service=mail&continue=https%3A%2F%2Fmail.google.co...

One account is all you need

One free account gets you into everything Google.

Take it all with you

Switch between devices, and pick up wherever you left off.

Name

Samuel | Smith

Choose your username

samsmith @gmail.com

That username is taken. Try another.

Create a password

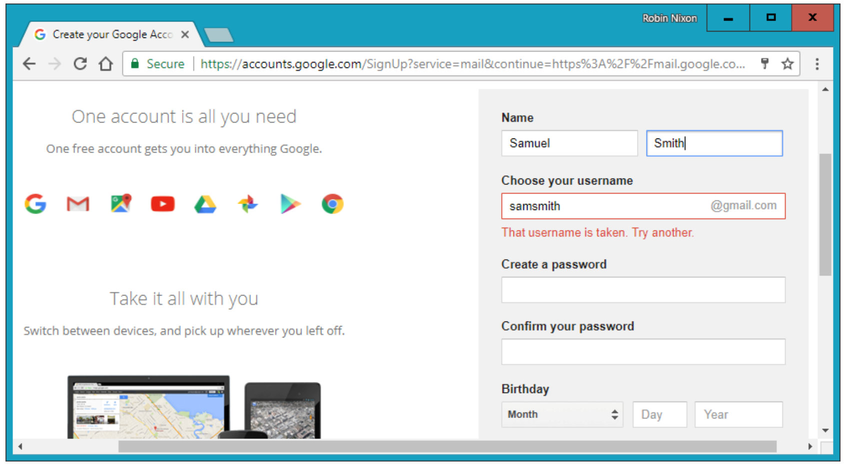
Confirm your password

Birthday

Month | Day | Year

Asynchronous communication in Gmail

Gmail username example


Create your Google Acco
Robin Nixon
Secure
https://accounts.google.com/SignUp?service=mail&continue=https%3A%2F%2Fmail.google.co...

One account is all you need

One free account gets you into everything Google.

Take it all with you

Switch between devices, and pick up wherever you left off.

Name

Samuel

Smith

Choose your username

samsmith @gmail.com

That username is taken. Try another.

Create a password

Confirm your password

Birthday

Month Day Year

1. Server outputs HTML to create web form
2. Server attaches some JavaScript to monitor username input box, check for
   - Typed text or not
   - Input deselected or not
3. When text has been entered and the field deselected, the JavaScript pass username to PHP script, wait for response
4. Web server looks up username (taken yet?) and replies back to JavaScript
5. JavaScript place indication text next to the username input box (available or not)
6. If the username is not available, JavaScript interrupts the submission and reemphasizes (alert box?)
7. Additional: suggest alternative username

* All of these take place in the background quitedly

Development Server

- WAMP, MAMP, LAMP :
  - W: Windows | M: Mac | L: Linux
  - A: Apache
  - M: MySQL
  - P: PHP

- Packages that bind the bundles programs together to a single program

- Recommended program: XAMPP

https://www.apachefriends.org/download.html

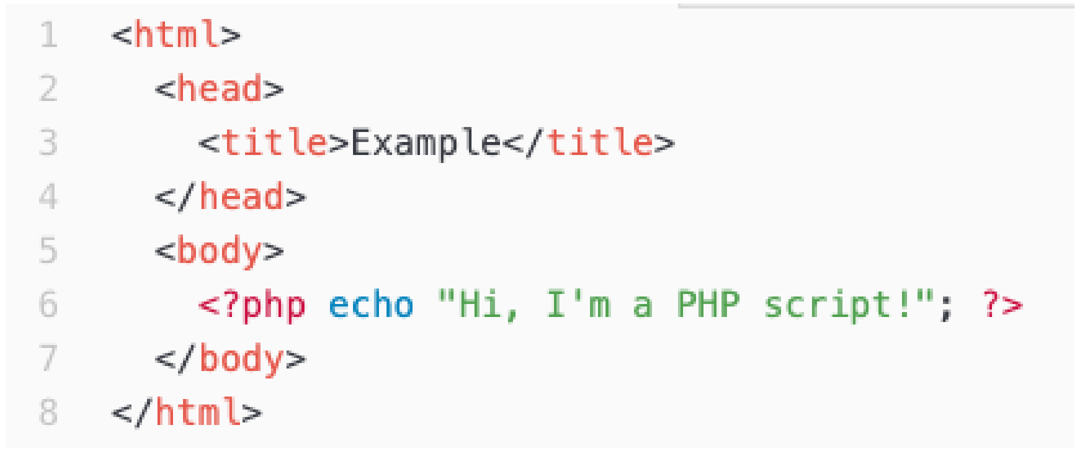
Note: Use Safari to open the link in Mac (I have a problem when use Chrome or Edge in Mac)

PHP

PHP

- PHP
  - recursive acronym for "PHP: Hypertext Preprocessor"
  - widely-used Open Source general-purpose scripting language
  - especially suited for Web development
  - can be embedded into HTML.

- PHP scripts reside between reserved PHP tags
  - This allows the programmer to embed PHP scripts within HTML pages



```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php echo "Hi, I'm a PHP script!"; ?>
  </body>
</html>
```

Save as demo.php

What is PHP (cont’d)

- Interpreted language, scripts are parsed **at run-time** rather than compiled beforehand
- Executed on the **server-side**
- Source-code **not visible** by client
    - ‘View Source’ in browsers **does not** display the PHP code
- Various built-in functions allow for fast development
- Compatible with many popular databases

PHP Overview

- Easy learning
- Syntax Perl- and C-like syntax. Relatively easy to learn.
- Large function library
- Embedded directly into HTML
- Interpreted, no need to compile
- Open Source server-side scripting language designed specifically for the web.

PHP Overview (cont.)

- Conceived in 1994
- Outputs not only HTML but can output XML, images (JPG & PNG), PDF files and even Flash movies all generated on the fly. Can write these files to the file system.
- Supports a **wide-range** of databases (20+ODBC).
- PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP.
- **PHP is used by 77.5%*** of all the websites whose server-side programming language we know.

*Updated May 2023

What does PHP code look like?

- Structurally similar to C/C++
- Supports procedural and object-oriented paradigm (to some degree)
- All PHP statements end with a semi-colon
- Each PHP script must be enclosed in the reserved PHP tag

```
<?php
    …
?>
```

Comments in PHP

- There are 3 ways to write comments in PHP
- Standard C, C++, and shell comment symbols

```
// C++ and Java-style comment (single line comment)

# Shell-style comments (single line comment)

/* C-style comments (block comment)
     These can span multiple lines */
```

Variables in PHP

- PHP variables must begin with a “`$`” sign
- Case-sensitive `$Foo != $foo != $fOo`
- Global and locally-scoped variables
  - Global variables can be used anywhere
  - Local variables restricted to a function or class
- Certain variable names reserved by PHP
  - Form variables `$_POST, $_GET`
  - Server variables `$_SERVER`
  - Etc.

Variable usage

```php
<?php
$foo = 25;    // Numerical variable
$bar = "Hello"; // String variable
$foo = ($foo * 7);  // Multiplies foo by 7
//$bar = ($bar * 7);  // Invalid expression
echo "Here";
?>
```

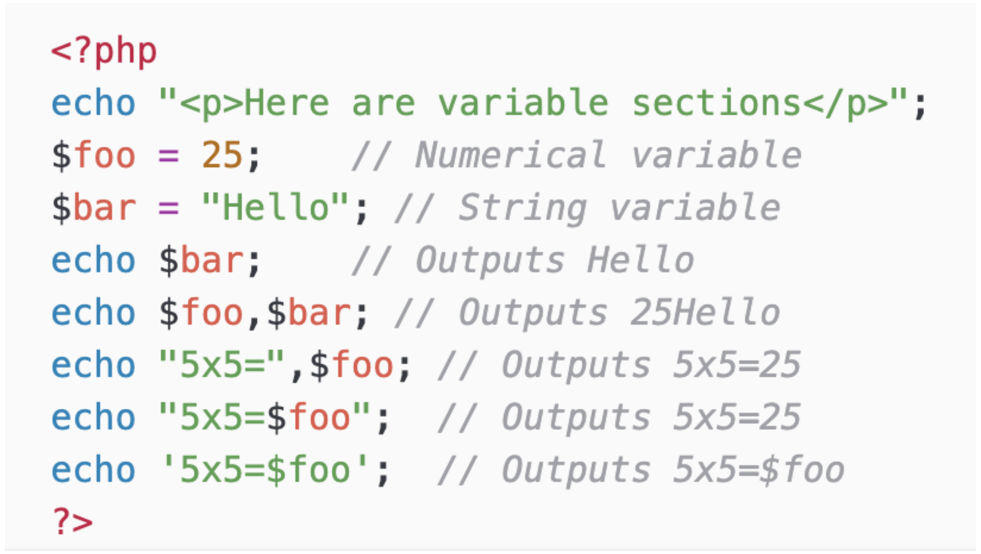
Echo

- The PHP command '**echo**' is used to output the parameters passed to it
  - The typical usage for this is to send data to the client's web-browser
- Syntax

```
void   echo (string arg1 [, string argn...])
```

  - In practice, arguments are not passed in parentheses since echo is a language construct rather than an actual function

Echo example



```php
<?php
echo "<p>Here are variable sections</p>";
$foo = 25;     // Numerical variable
$bar = "Hello"; // String variable
echo $bar;     // Outputs Hello
echo $foo,$bar; // Outputs 25Hello
echo "5x5=",$foo; // Outputs 5x5=25
echo "5x5=$foo";  // Outputs 5x5=25
echo '5x5=$foo';  // Outputs 5x5=$foo
?>
```

- Notice how **`echo ‘5x5=$foo’`** outputs **`$foo`** rather than replacing it with **25**
- Strings in single quotes **`(‘ ’)`** are not interpreted or evaluated by PHP
- This is true for both variables and character escape-sequences (such as **`\n`** or **`\\`**)

We can also use `print, printf` instead of `echo`

Arithmetic Operations

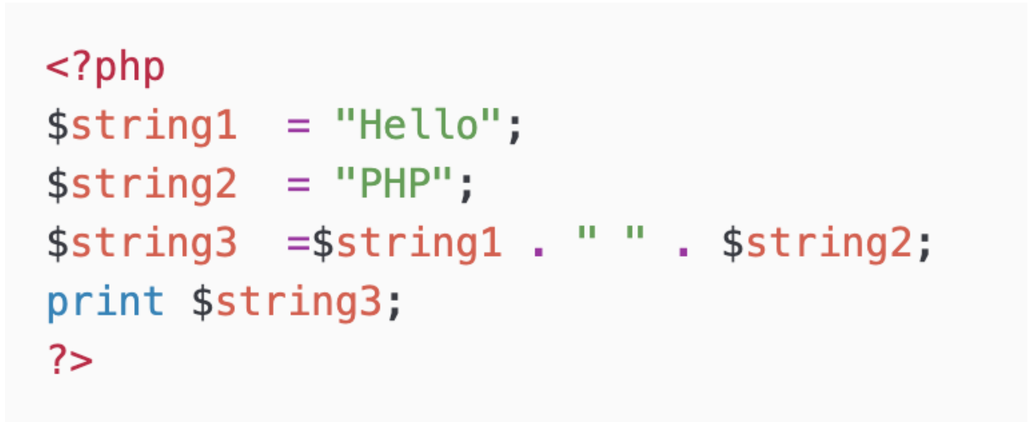
- `$a - $b` // subtraction
- `$a * $b` // multiplication
- `$a / $b` // division
- `$a += 5` // $a = $a+5 Also works for *= and /=

```
<?php
  $a  = 15;
  $b  = 30;
  $total  = $a + $b;
  print $total;
  print "<p><h1>$total</h1>";
  // total is 45
?>
```

`echo` and `print` are similar

Concatenation

- Use a period “ . ” to join strings into one.



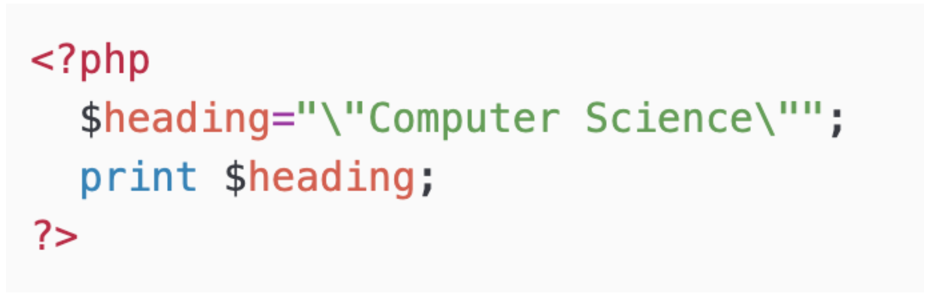
```php
<?php
$string1  = "Hello";
$string2  = "PHP";
$string3  =$string1 . " " . $string2;
print $string3;
?>
```

Output

Hello PHP

Escaping the Character

- If the string has a set of double quotation marks that must remain visible, use the \ [backslash] before the quotation marks to ignore and display them.



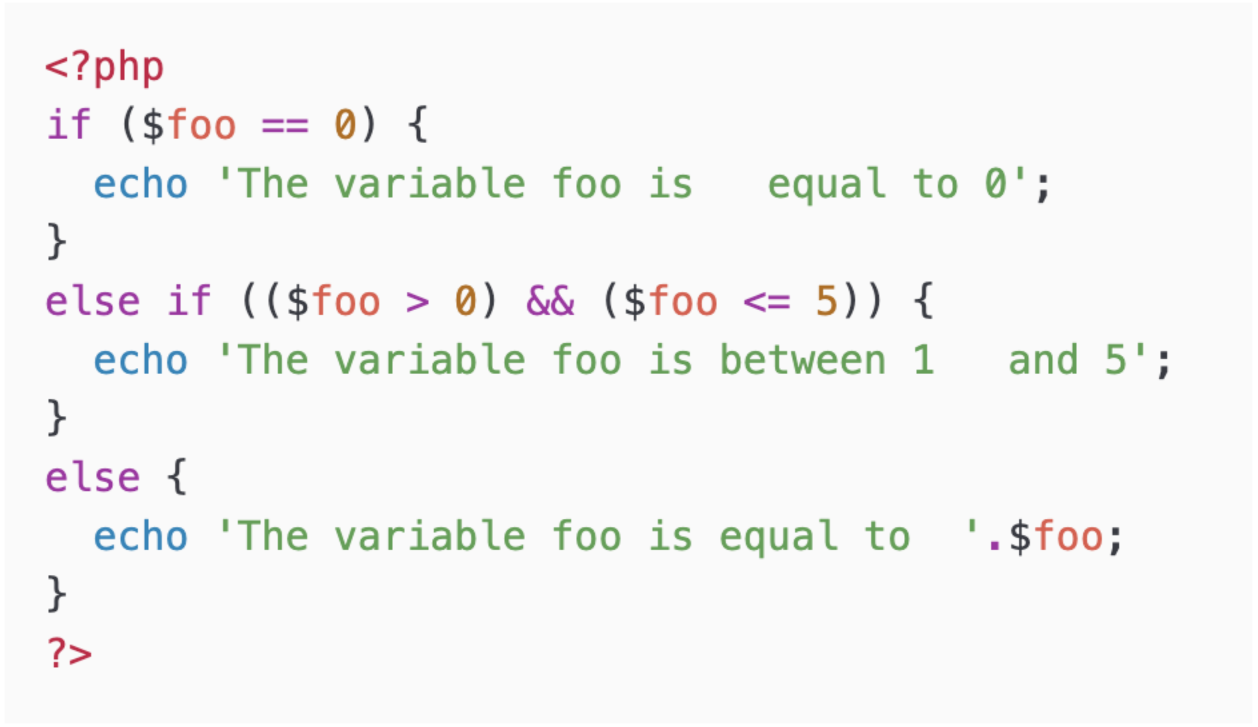
```php
<?php
  $heading="\"Computer Science\"";
  print $heading;
?>
```

Output

“Computer Science”

PHP Control Structures

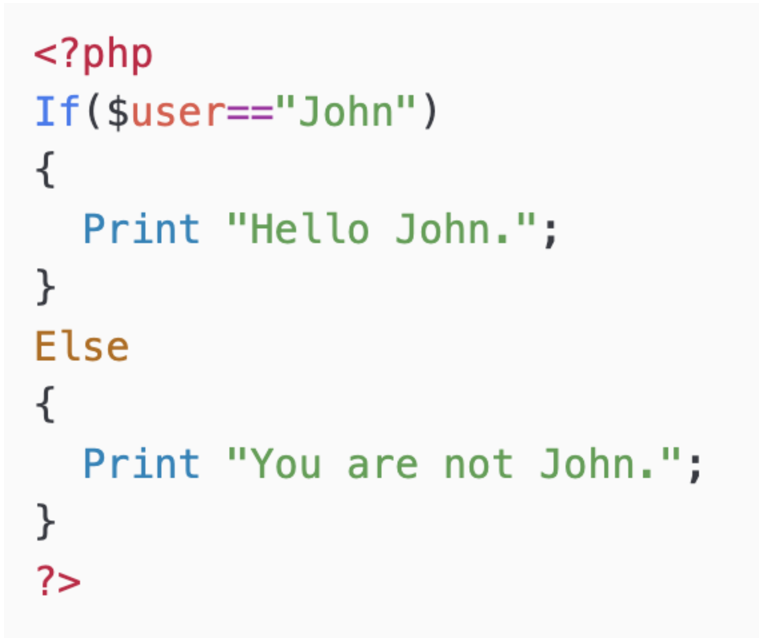
- Control Structures
- Grouped into conditional (branching) structures (e.g. if/else) and repetition structures (e.g. while loops).
- Example of if/else statement →



```php
<?php
if ($foo == 0) {
  echo 'The variable foo is  equal to 0';
}
else if (($foo > 0) && ($foo <= 5)) {
  echo 'The variable foo is between 1  and 5';
}
else {
  echo 'The variable foo is equal to  '.$foo;
}
?>
```

If … Else…

```
if (condition)
{
        Statements;
}
else
{
        Statement;
}
```



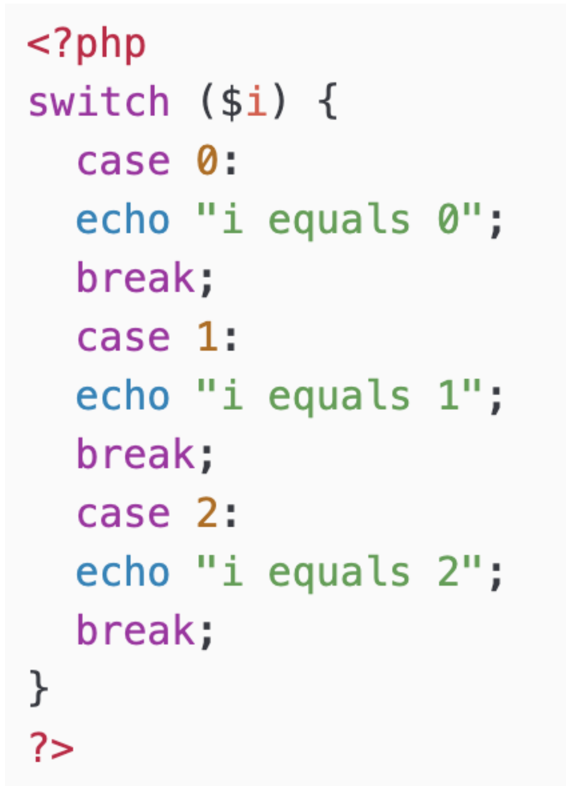
```
<?php
If($user=="John")
{
  Print "Hello John.";
}
Else
{
  Print "You are not John.";
}
?>
```

No THEN in PHP

Switch....case....

Can also use with string eg... case("boy"), case("girl")

```
switch(expr)
{
   case (condition):
        Statements;
        break;
   case (condition):
        Statements;
        break;
   default :
        Statements;
}
```



```
<?php
switch ($i) {
  case 0:
  echo "i equals 0";
  break;
  case 1:
  echo "i equals 1";
  break;
  case 2:
  echo "i equals 2";
  break;
}
?>
```

While Loops

```
while (condition)
{
   Statements;
}
```

```
<?php
$count=0;
while($count<3)
{
  print "hello PHP. ";
  $count += 1;
  // $count = $count + 1;
  // or
  // $count++;
}
?>
```

Output

hello PHP. hello PHP. hello PHP.

For

```
for(expr1; expr2; expr3)
{
  Statements;
}
```

```php
<?php
for ($i = 1; $i <= 10; $i++) {
  echo $i;
}
for ($i2 = 1; ; $i2++){
  if ($i2 > 10){
    break;
  }
}
$i3 = 1;
for ( ; ; ){
  if ($i3 > 10) {
    break;
  }
  echo $i3;
  $i3++;
}
?>
```

Comparing operator

`==`	equal
`!=` OR `<>`	not equal
`<`	less than
`>`	more than
`<=`	less than or equal to
`>=`	more than or equal to

Logical operator

• && OR and	And
• || OR or	Or
• !	Not
• ^ OR xor	Exclusive Or

Functions

- A set of statements that performs a particular function and optionally returns a value.
- Advantages
  - Less typing
  - Reduce syntax and other programming errors
  - Decrease the loading time of program files
  - Decrease execution time (function is compiled only once)
  - Accept arguments and can be used for general or specific cases

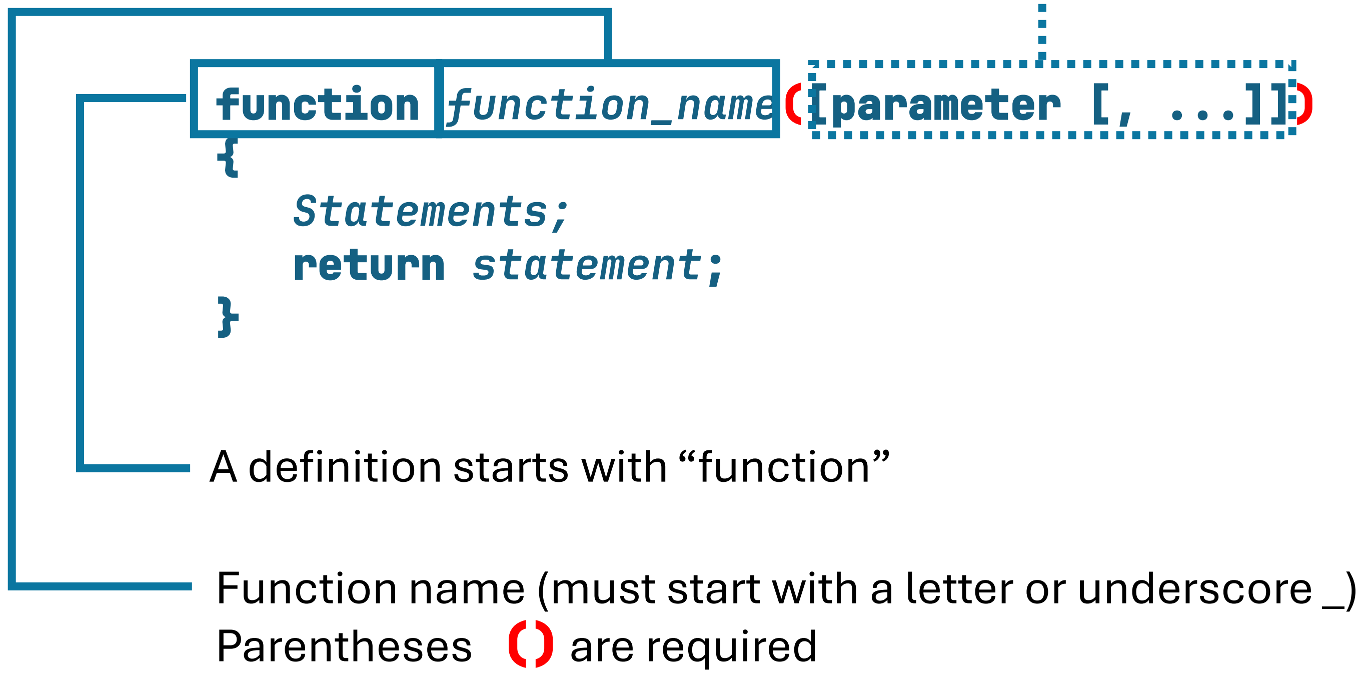
PHP Functions

- PHP comes with hundreds of ready-made, built in function
  - **Example:**

```
echo date("l");          // Displays the day of the week
phpinfo();
echo strrev(" .dlrow olleH"); // Reverse string // Hello world.
echo str_repeat("Hip ", 2);   // Repeat string // Hip Hip
echo strtoupper("hooray!");   // String to uppercase // HOORAY!
echo strtolower(“HOOrAY!");   // String to lowercase // hooray!)
echo ucfirst(“hello”); // First character of a string to uppercase // Hello
```

- Can take any number of arguments, including zero

Defining a function

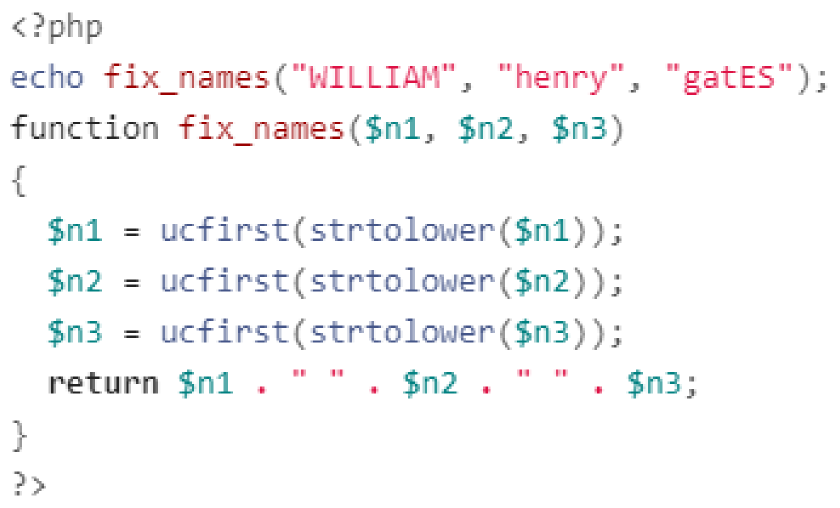


Optional
One or more parameters
Separated by commas

```
function function_name([parameter [, ...]])
{
    Statements;
    return statement;
}
```

A definition starts with “function”

Function name (must start with a letter or underscore _)
Parentheses ( ) are required



```
<?php
echo fix_names("WILLIAM", "henry", "gatES");
function fix_names($n1, $n2, $n3)
{
  $n1 = ucfirst(strtolower($n1));
  $n2 = ucfirst(strtolower($n2));
  $n3 = ucfirst(strtolower($n3));
  return $n1 . " " . $n2 . " " . $n3;
}
?>
```

Output

William Henry Gates

Users often leave their Caps Lock key on, or accidentally insert capital letters in the wrong places.

PHP Arrays

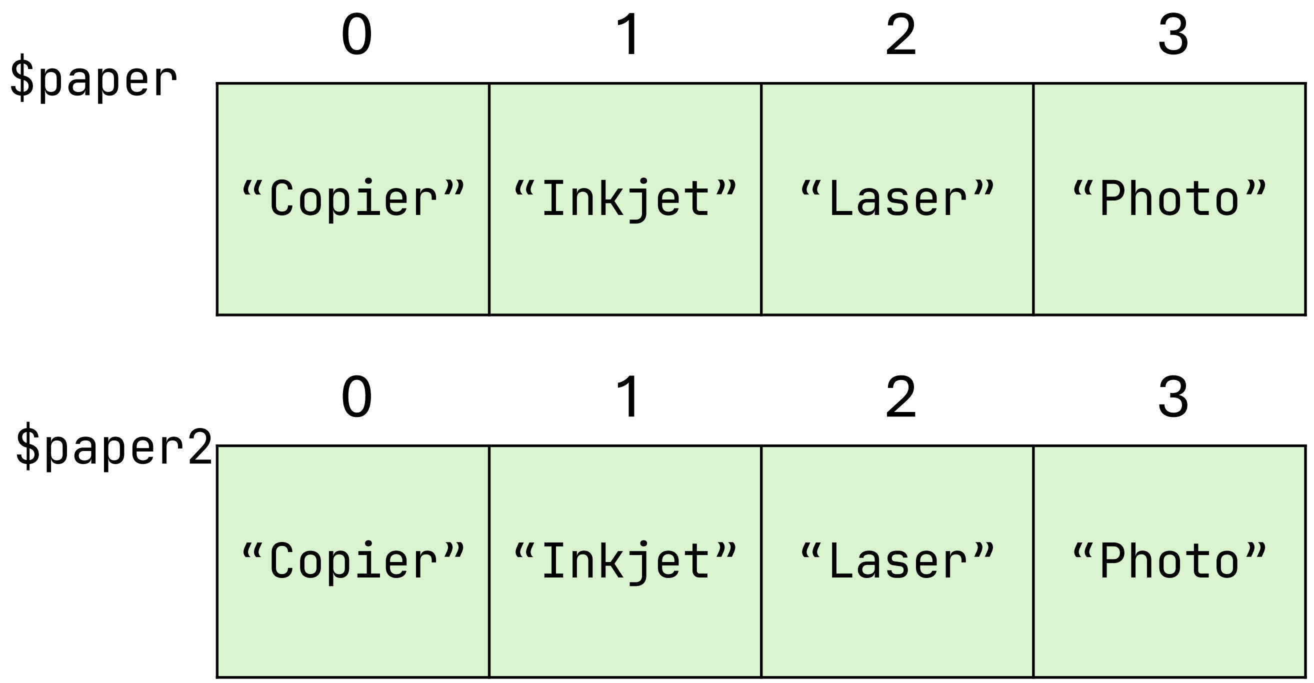
An array can store one or more values in a single variable name.

Each element in the array is assigned its own ID so that it can be easily accessed.

```
$array[key] = value;
```

Numerical Indexes

```php
<?php
echo "<p>";
$paper[] = "Copier";
$paper[] = "Inkjet";
$paper[] = "Laser";
$paper[] = "Photo";
print_r($paper);
echo "<p>";
$paper2[0] = "Copier";
$paper2[1] = "Inkjet";
$paper2[2] = "Laser";
$paper2[3] = "Photo";
print_r($paper2);

for ($j = 0 ; $j < 4 ; ++$j)
echo "$j: $paper[$j]<br>";

?>
```



Output

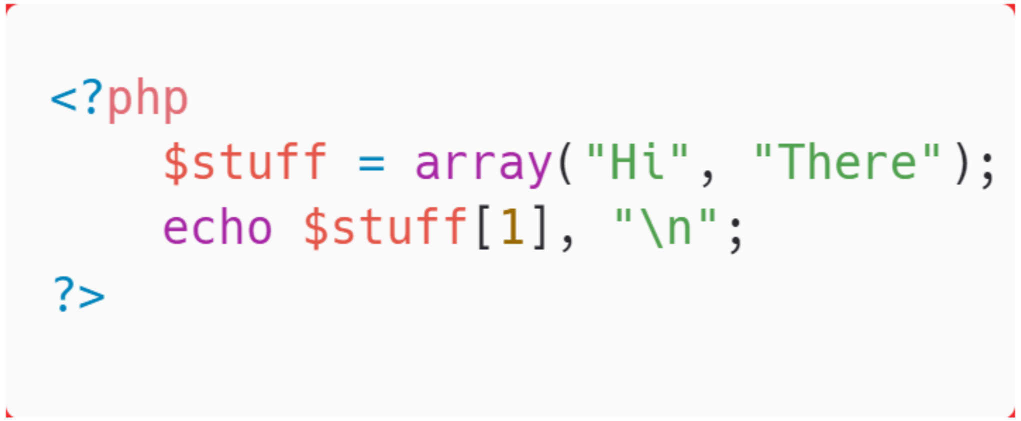
```
Array ( [0] => Copier [1] => Inkjet [2] => Laser [3] => Photo )

Array ( [0] => Copier [1] => Inkjet [2] => Laser [3] => Photo )
```

Numerical array

- Automatically
- Manually

```
$names =
array("Peter","Quagmire","Joe");
```

```
$names[0] = "Peter";
$names[1] = "Quagmire";
$names[2] = "Joe";
```



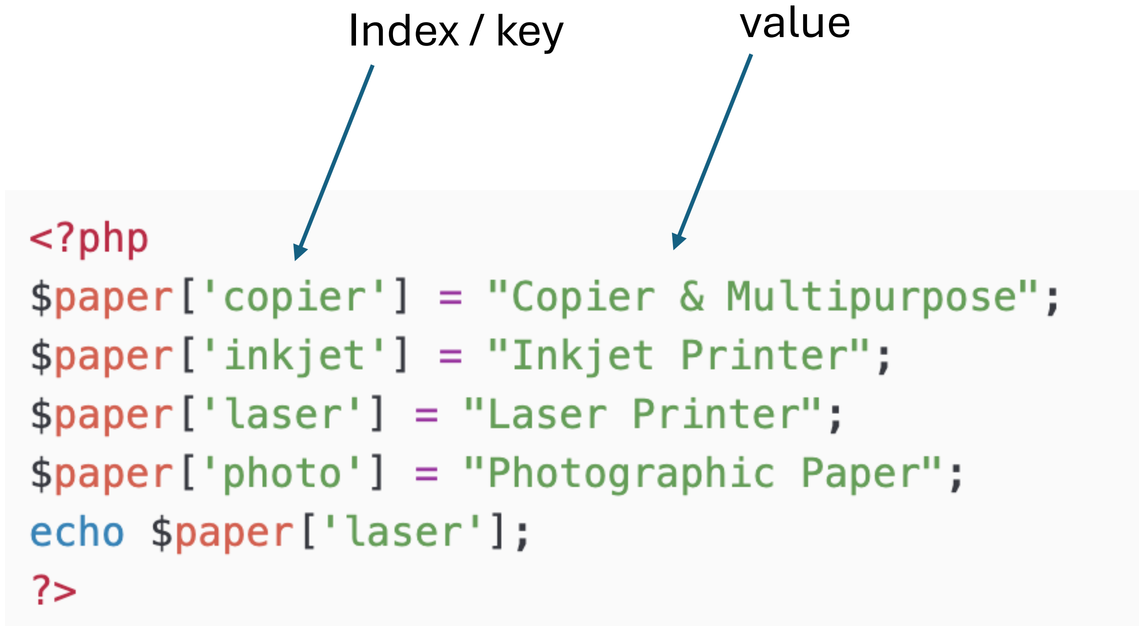
```
<?php
    $stuff = array("Hi", "There");
    echo $stuff[1], "\n";
?>
```

Output

There

Associative Arrays

- An associative array, each ID key is associated with a value.
- When storing data about specific named values, a numerical array is not always the best way to do it.
- With associative arrays we can use the values as <u>index</u> or <u>keys</u> and assign <u>values</u> to them.



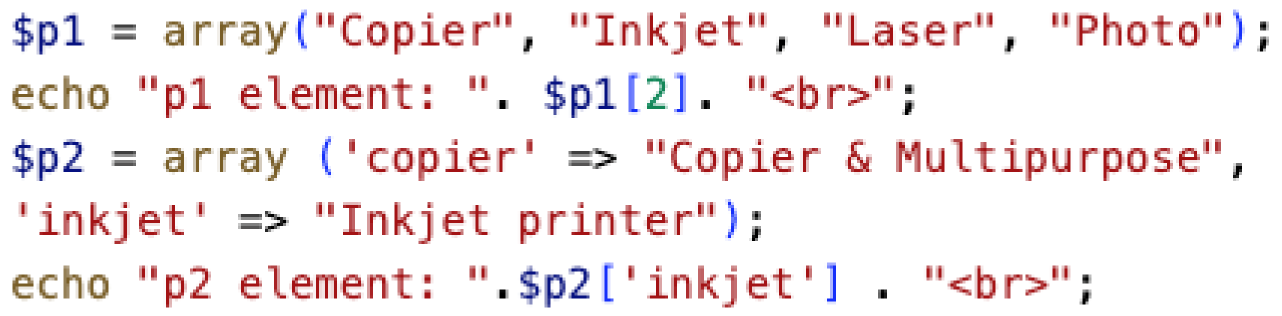
```php
<?php
$paper['copier'] = "Copier & Multipurpose";
$paper['inkjet'] = "Inkjet Printer";
$paper['laser'] = "Laser Printer";
$paper['photo'] = "Photographic Paper";
echo $paper['laser'];
?>
```

Not possible in C

Associative Arrays

- Often used when you are extracting information from XML and HTML.
- For example, the HTML parser (used by search engine)
  - Place all the elements of a web page into an associative array
    - `$html['title'] = "My web page";`
    - `$html['body'] = "... body of web page ...";`
  - Break down all the links in a page into another array
  - Also break all headings and subheadings into another array
- Easy to write and debug

Assignment using **array** keyword



```
$p1 = array("Copier", "Inkjet", "Laser", "Photo");
echo "p1 element: ". $p1[2]. "<br>";
$p2 = array ('copier' => "Copier & Multipurpose",
'inkjet' => "Inkjet printer");
echo "p2 element: ".$p2['inkjet'] . "<br>";
```

The old way and shorten style of assignment

Assignment using format
key => value

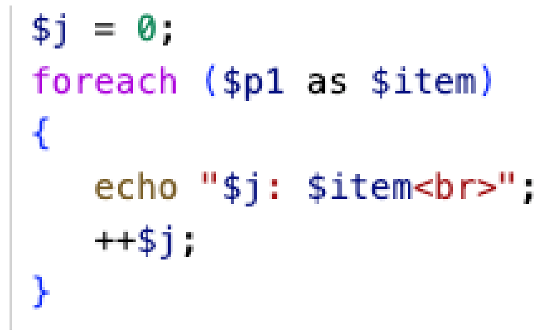
```
echo $p1['inkjet']; // Undefined index
echo $p2[3];     // Undefined offset
```

foreach .. As loop

Created in PHP especially for arrays

```
$p1 = array("Copier", "Inkjet", "Laser", "Photo");
```

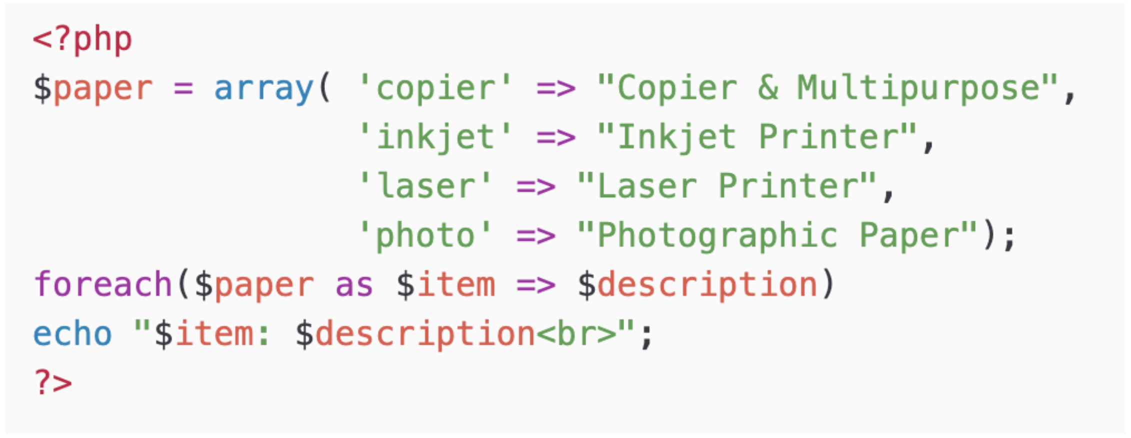
Numerical index



```
$j = 0;
foreach ($p1 as $item)
{
    echo "$j: $item<br>";
    ++$j;
}
```

```
$p2 = array ('copier' => "Copier & Multipurpose",
'inkjet' => "Inkjet printer");
```

Associative arrays



```
<?php
$paper = array( 'copier' => "Copier & Multipurpose",
                'inkjet' => "Inkjet Printer",
                'laser' => "Laser Printer",
                'photo' => "Photographic Paper");
foreach($paper as $item => $description)
echo "$item: $description<br>";
?>
```

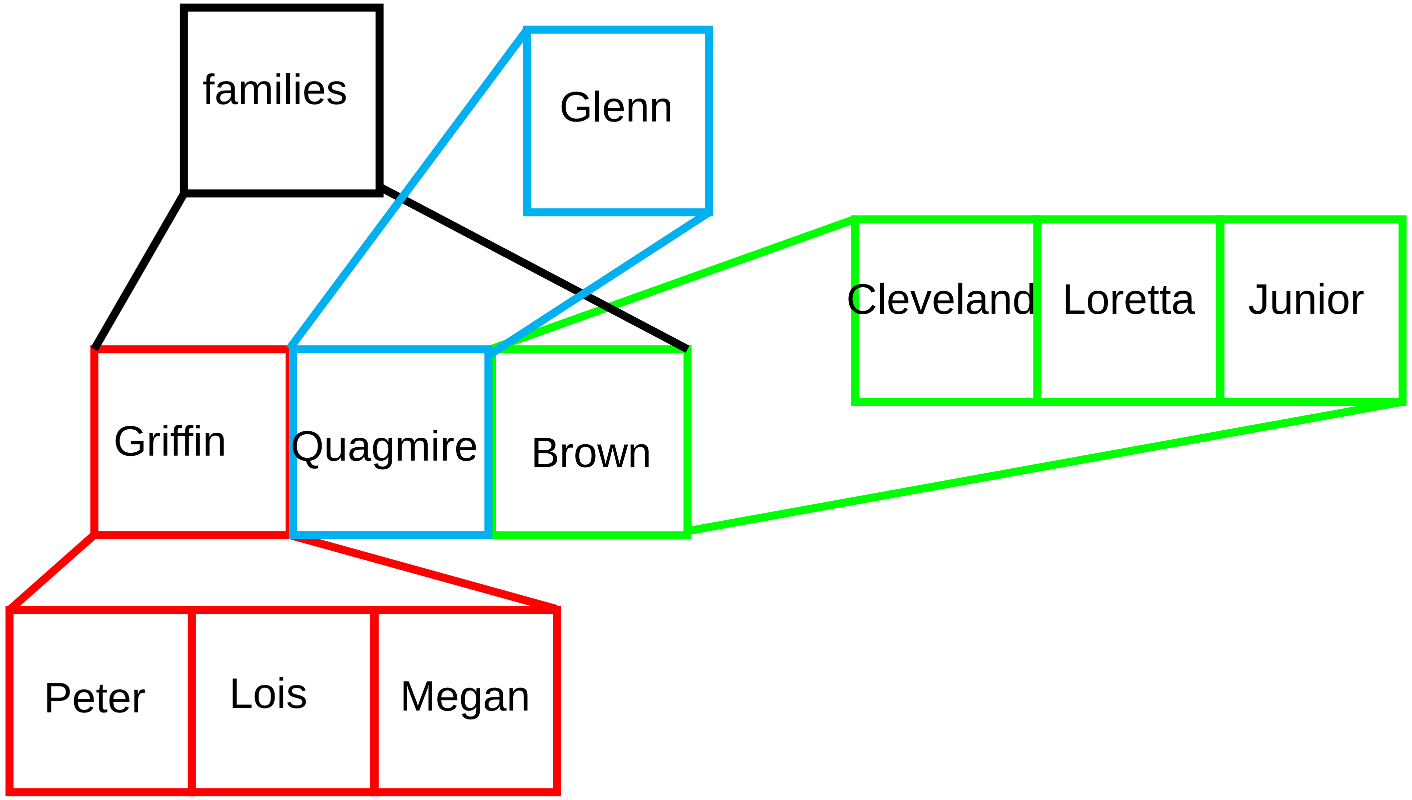
Multidimensional Arrays

- In a multidimensional array, each element in the main array can also be an array.
- Each element in the sub-array can be an array, and so on.

Example

demo

```php
$families = array
(
  "Griffin"=>array(
    "Peter",
    "Lois",
    "Megan"
  ),
  "Quagmire"=>array ("Glenn"),
  "Brown"=>array(
    "Cleveland",
    "Loretta",
    "Junior"
  )
);
?>
```



```php
<?php
$products = array(
  'paper' => array(
    'copier' => "Copier & Multipurpose",
    'inkjet' => "Inkjet Printer",
    'laser' => "Laser Printer",
    'photo' => "Photographic Paper"
  ),
  'pens' => array(
    'ball' => "Ball Point",
    'hilite' => "Highlighters",
    'marker' => "Markers"
  ),
  'misc' => array(
    'tape' => "Sticky Tape",
    'glue' => "Adhesives",
    'clips' => "Paperclips"
  )
);
echo "<pre>";
foreach($products as $section => $items)
foreach($items as $key => $value)
echo "$section:\t$key\t($value)<br>";
echo "</pre>";
?>
```

Output

```
paper: copier (Copier & Multipurpose)
paper: inkjet (Inkjet Printer)
paper: laser (Laser Printer)
paper: photo (Photographic Paper)
pens: ball (Ball Point)
pens: hilite (Highlighters)
pens: marker (Markers)
misc: tape (Sticky Tape)
misc: glue (Adhesives)
misc: clips (Paperclips)
```

```php
echo $products['misc']['glue'];
```

Output

```
Adhesives
```

Creating a multidimensional numeric array

- Create the board for a chess game with the pieces in their starting positions
  - Lowercase letter: black pieces
  - Uppercase letter: white piece

```php
<?php
$chessboard = array(
  array('r', 'n', 'b', 'q', 'k', 'b', 'n', 'r'),
  array('p', 'p', 'p', 'p', 'p', 'p', 'p', 'p'),
  array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
  array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
  array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
  array(' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '),
  array('P', 'P', 'P', 'P', 'P', 'P', 'P', 'P'),
  array('R', 'N', 'B', 'Q', 'K', 'B', 'N', 'R')
);
echo "<pre>";
foreach($chessboard as $row)
{
  foreach ($row as $piece)
  echo "$piece ";
  echo "<br>";
}
echo "</pre>";
?>
```

Output

```
r n b q k b n r
p p p p p p p p
P P P P P P P P
R N B Q K B N R
```

Dumping an Array

- The function print_r() dumps out PHP data - it is used mostly for debugging

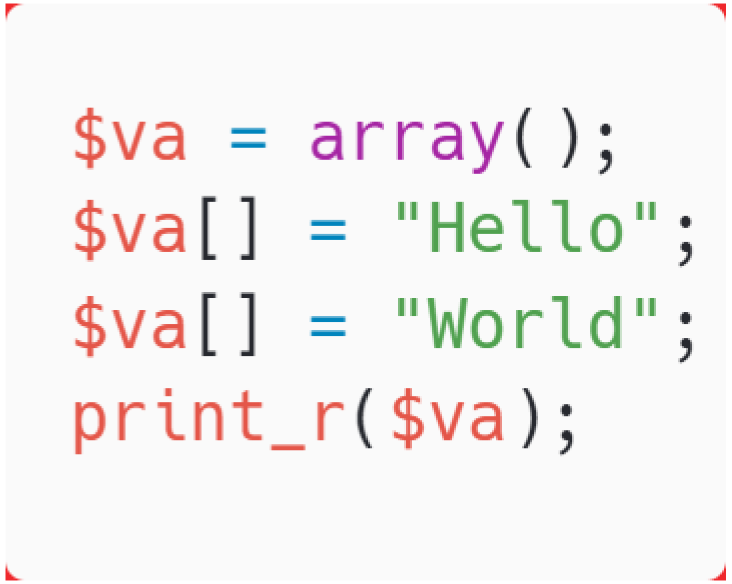
```php
<?php
    $stuff = array("name" => "Chuck",
      "course" => "PHPIntro");
    print_r($stuff);
?>
```

Output

```
Array(
     [name] => Chuck
     [course] => PHPIntro
)
```

Building up an Array

- You can allocate a new item in the array and add a value at the same time using empty square braces [] on the right hand side of an assignment statement



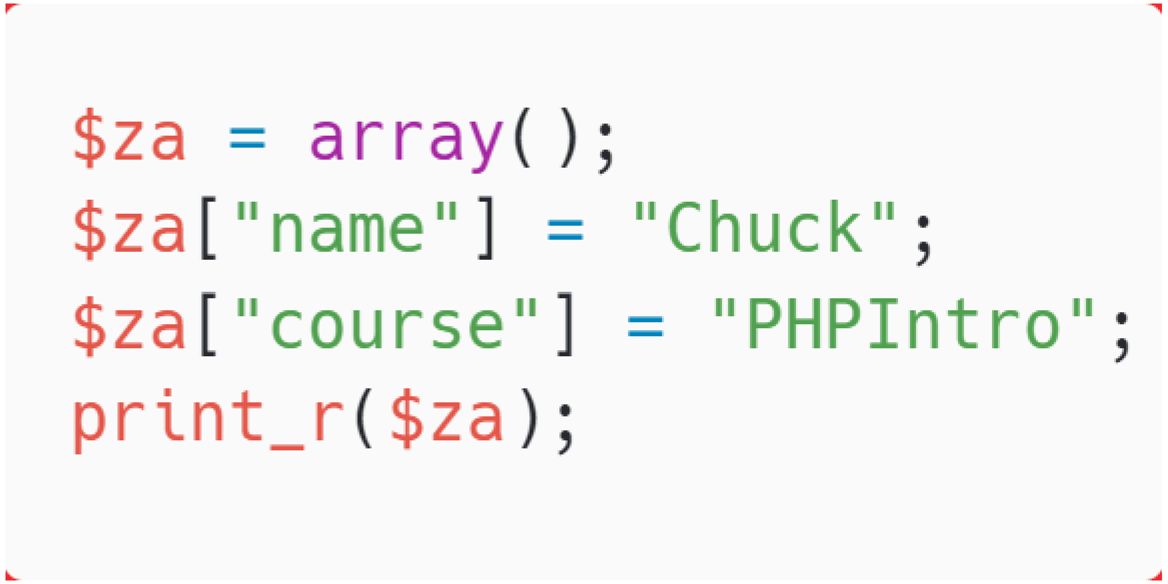
```php
$va = array();
$va[] = "Hello";
$va[] = "World";
print_r($va);
```

Output

```
Array(
   [0] => Hello
   [1] => World
)
```

Building up an Array

- You can also add new items in an array using a key as well



```php
$za = array();
$za["name"] = "Chuck";
$za["course"] = "PHPIntro";
print_r($za);
```

Output

```
Array(
   [name] => Chuck
   [course] => PHPIntro
)
```

Array Functions

Full list at: https://www.php.net/manual/en/ref.array.php

- **`count($ar)`** - How many elements in an array
- **`is_array($ar)`** - Returns TRUE if a variable is an array
- **`isset($ar['key'])`** - Returns TRUE if key is set in the array
- **`sort($ar)`** - Sorts the array values (loses key)
- **`ksort($ar)`** - Sorts the array by key
- **`asort($ar)`** - Sorts array by value, keeping key association
- **`shuffle($ar)`** - Shuffles the array into random order

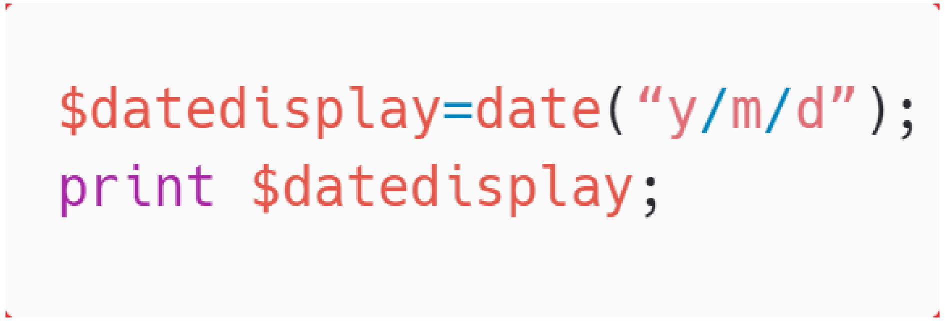
Date Display

More info about this function: https://www.php.net/manual/en/datetime.format.php

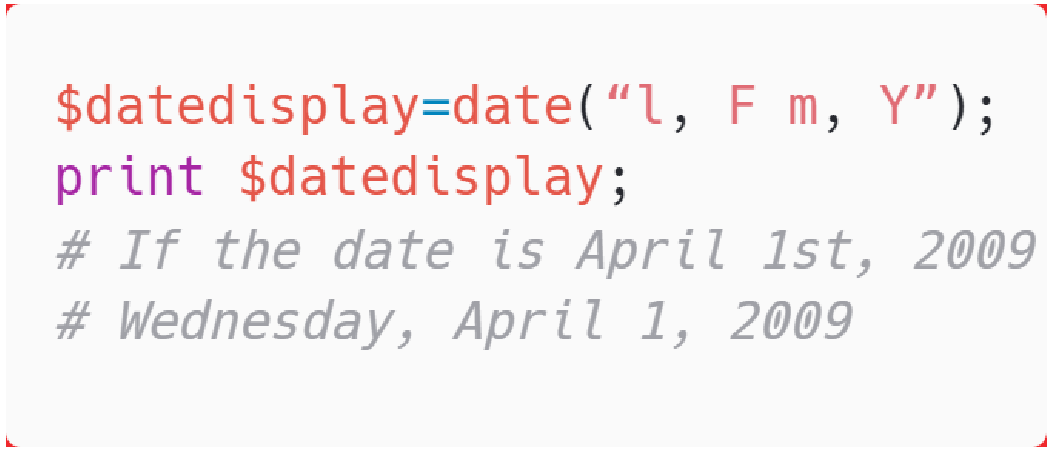
Month, Day & Date format symbols

M	Jan
F	January
m	01
n	1

Day of Month	d	01
Day of Month	J	1
Day of Week	l	Monday
Day of Week	D	Mon



```
$datedisplay=date(“y/m/d”);
print $datedisplay;
```



```
$datedisplay=date(“l, F m, Y”);
print $datedisplay;
# If the date is April 1st, 2009
# Wednesday, April 1, 2009
```

Output

14/1/30

Output

Wednesday, April 1, 2009

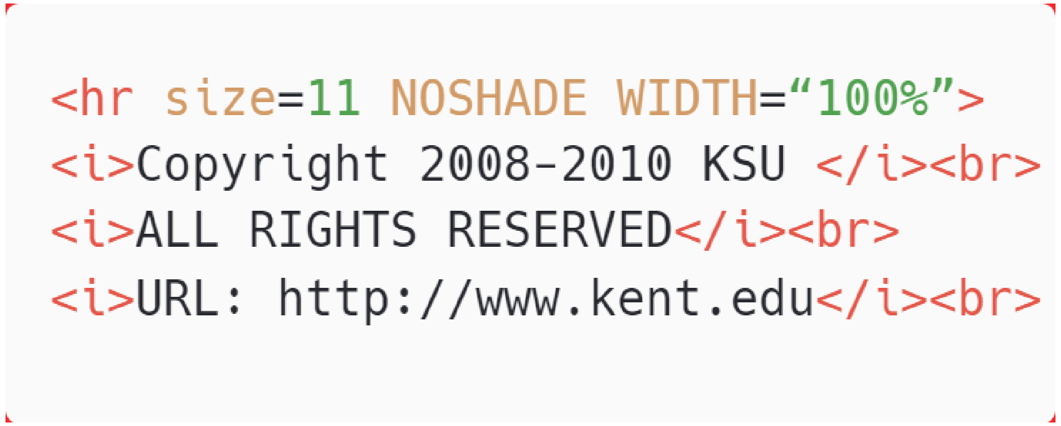
Include Files

```
include “opendb.php”;
include “closedb.php”;
```

- This inserts files; the code in files will be inserted into current code. This will provide useful and protective means once you connect to a database, as well as for other repeated functions.

```
include (“footer.php”);
```

- The file footer.php might look like:



```
<hr size=11 NOSHADE WIDTH=“100%”>
<i>Copyright 2008-2010 KSU </i><br>
<i>ALL RIGHTS RESERVED</i><br>
<i>URL: http://www.kent.edu</i><br>
```

PHP - Forms

- Access to the HTTP POST and GET data is simple in PHP
- The global variables **`$_POST[]`** and **`$_GET[]`** contain the request data

On the form page

```
<form action="form.php" method="post">
    <input type="submit" name="submit" value="Submit">
    <input type="submit" name="cancel" value="Cancel">
</form>
```

form.php page

```
<? php
if ($_POST["submit"])
  echo "<h2>You clicked Submit!</h2>";
else if ($_POST["cancel"])
  echo "<h2>You clicked Cancel!</h2>";
?>
```

http://www.cs.kent.edu/~nruan/form.php

HTML Forms

Web Data

- Most interesting web pages revolve around data
  - examples: Google, IMDB, Digg, Facebook, YouTube, Rotten Tomatoes
  - can take many formats: text, HTML, XML, multimedia
- Many of them allow us to access their data
- Some even allow us to *submit our own new data*
- Most server-side web programs accept parameters that guide their execution

Reading/writing an entire file

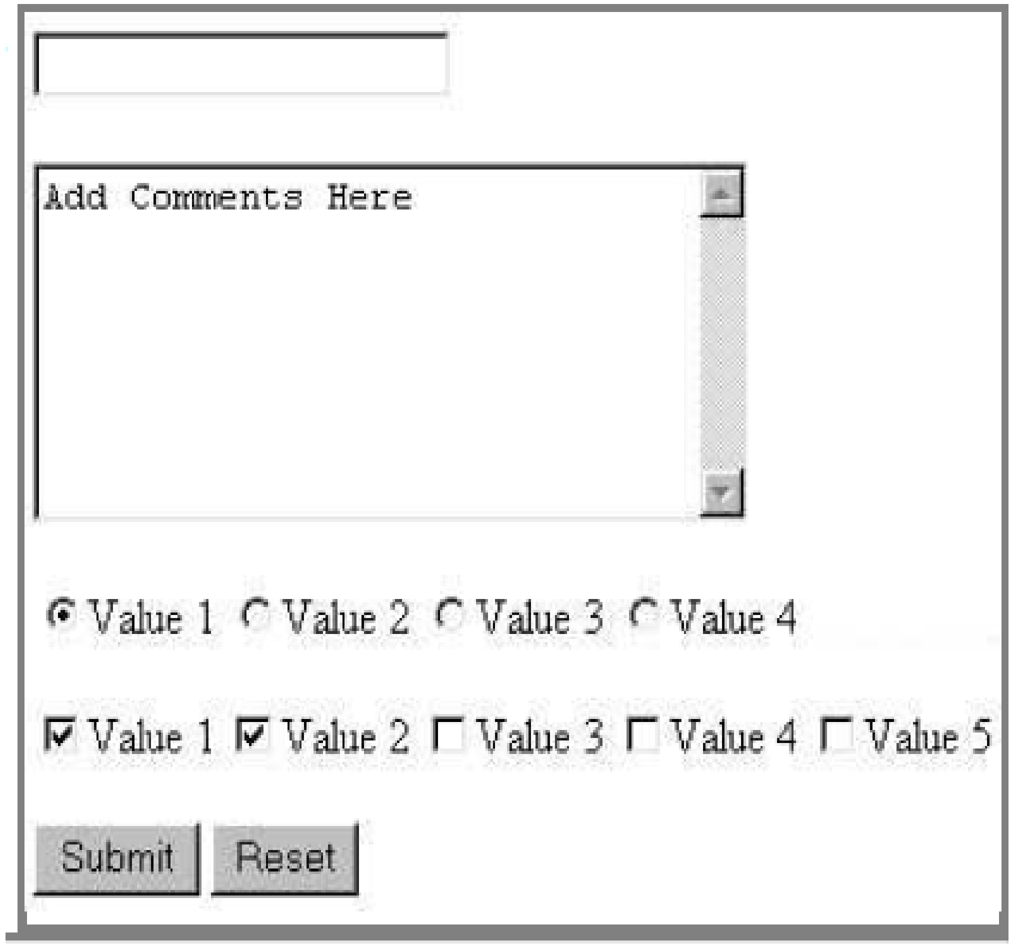
```
URL?name=value&name=value...
```

- **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
- PHP code on the server can examine and utilize the value of parameters

```
http://example.com/student_login.php?username=xenia&sid=1234567
```

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



How Forms Work

Web forms have two components:

- The **form on the page** that collects input
- An **application on the server** that processes the collected information

Web Form Transaction

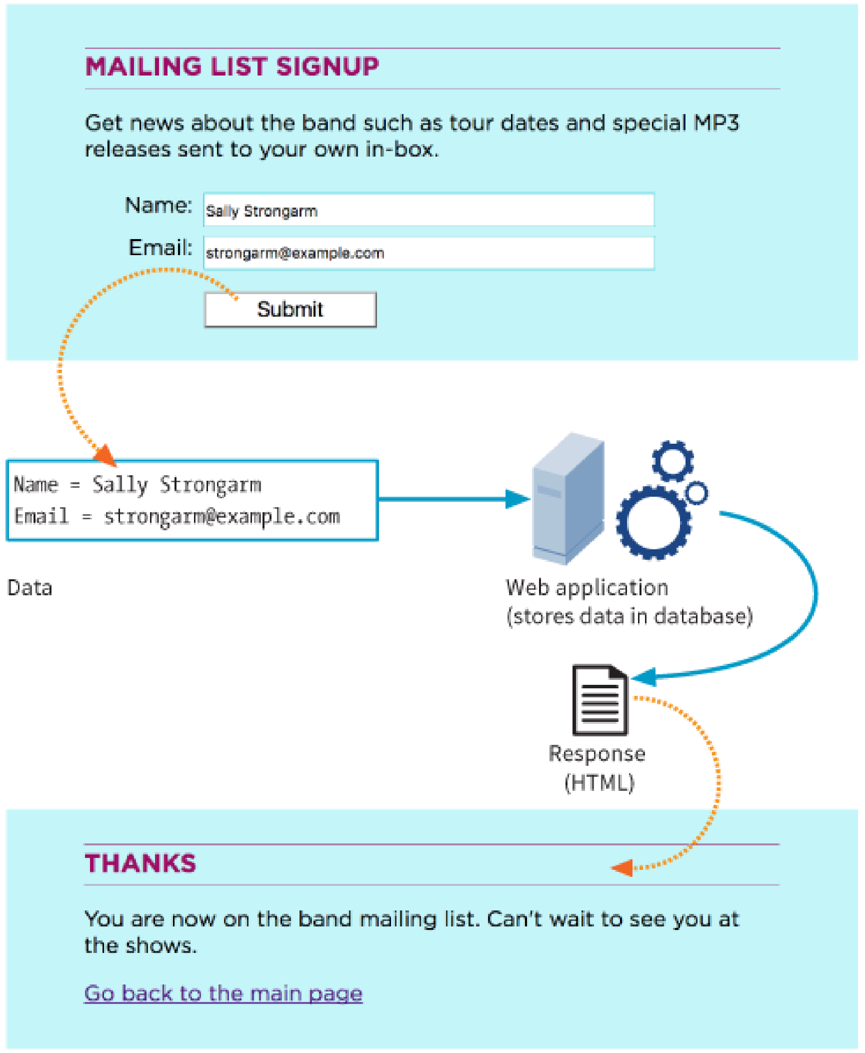
1. Browser renders the form inputs as indicated in the markup.

2. User enters information in the form and hits Submit.

3. The browser encodes the information entered and sends it to the server.

4. The application processes the information.

5. The application returns a response (for example, a thank you message or reloading the page).


MAILING LIST SIGNUP
Get news about the band such as tour dates and special MP3 releases sent to your own in-box.
Name: Sally Strongarm
Email: strongarm@example.com
Submit
Name = Sally Strongarm
Email = strongarm@example.com
Data
Web application (stores data in database)
Response (HTML)
THANKS
You are now on the band mailing list. Can't wait to see you at the shows.
Go back to the main page

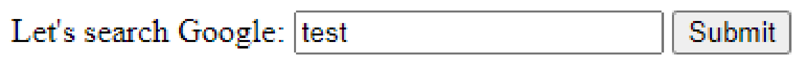
Web Processing Applications

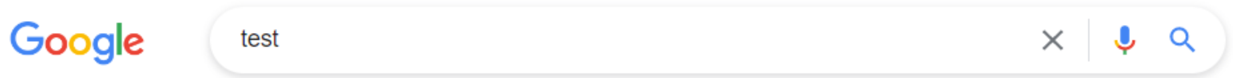
Web forms may be processed by any of the following technologies:

- PHP (*.php*)
- Microsoft ASP (*.asp*)
- Microsoft ASP.net (*.aspx*)
- Ruby on Rails
- JavaServer Pages (*.jsp*)
- Python

Form example

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





- Wrap the form's controls in a block element such as div

The form Element

- The **`form`** element is a container for all the content in the form and its input controls.
- The **`action`** and **`method`** attributes are necessary for interacting with the processing program.

```
<form action="URL" method="POST or GET">
  <!-- Form content and inputs here -->
</form>
```

The action Attribute

The **`action`** attribute provides the location of the script or application that will process the collected form data.

```
<form action="mailinglist.php" method="POST">
```

The method Attribute

```
<form action="mailinglist.php" method="POST">
```

The **`method`** attribute specifies how the encoded information should be sent to the server (GET or POST):

- **GET**: The encoded data is tacked onto the URL sent to the server:

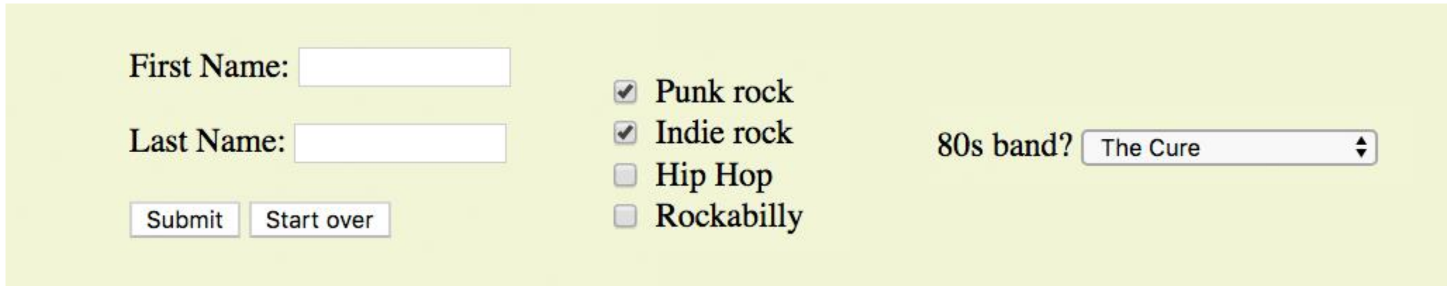
```
get http://www.bandname.com/
mailinglist.php?name=Sally%20Strongarm&email=strongarm%40example.com
```

- **POST**: Data is sent in a **separate transaction** and can be encrypted with HTTPS.

NOTE: POST is the most common method.

Form Control Elements

- **Form control elements** (also called **widgets**) collect data from the user. A few examples:



```
<input type="text">
<input type="submit">
<input type="checkbox">
<select>
```

Form Control Elements (cont’d)

Form controls collect data in **variable/value pairs**.

Examples:

variable	= "email"
value	= jen@example.com
variable	= "color"
value	= green

Variables (the name Attribute)

- A **variable** is a bit of information collected by a form control (example: the user's last name).
- The required **`name`** attribute in the control element provides the name of the variable for that control:

```
<input name="lastname">
```

NOTE: The variable name is also programmed into the web processing script or app, so the name in the markup must match the name in the processor.

Values

- The data entered or selected by the user in the form control is the **value** that gets sent to the server. It is paired with the variable for that control.
- You can provide a **default value** with the **`value`** attribute:

```
Name: <input name="lastname" value="Unknown">
```

In this example, if the text input is left blank, the value “Unknown” would be sent to the server for the variable “lastname”.

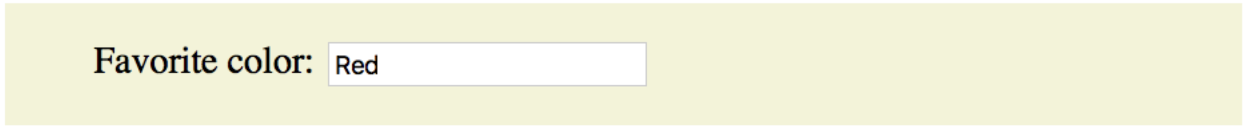
FORM CONTROL ELEMENTS

Text Entry Input

```
<input type="text">
```

- **`type`**: Type of input control, in this case a single-line text field
- **`name`**: Required variable name
- **`value`**: Default text that appears in the field and is sent to server if the field is left blank
- **`maxlength`**, **`minlength`**: Sets a character limit for the field
- **`size`**: The length of the field in number of characters (CSS is more common for sizing)

```
<input type="text" name="color" value="Red" maxlength="24">
```


Favorite color: Red

FORM CONTROL ELEMENTS

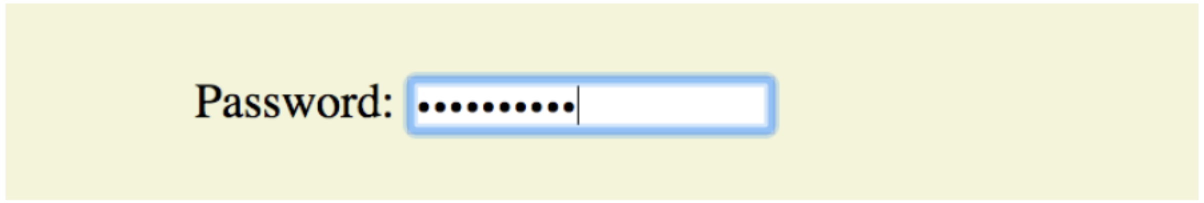
Password Field

```
<input type="password">
```

- Like a text entry field, except the characters are obscured from view

- The data entered is *not* encrypted on the way to the server (unless it uses HTTPS, a secure web protocol).

```
<input type="password" name="pswd" maxlength="10">
```



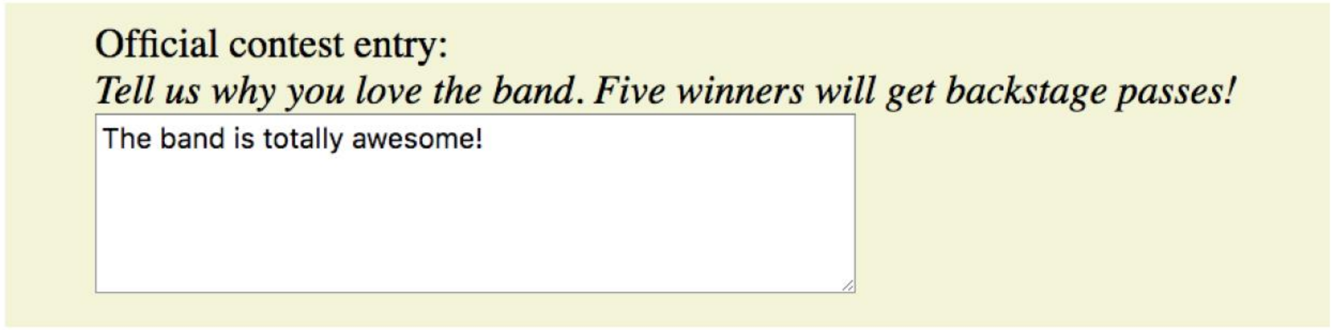
FORM CONTROL ELEMENTS

Multi-line Text Entry

`<textarea> </textarea>`

- The content of the **`textarea`** element is the default value.
- **`rows`**: The number of rows tall the field is initially drawn (users can write more)
- **`cols`**: Width of initial text field, in number of characters
- **`maxlength`**, **`minlength`**: Limits the number of characters that can be entered

```
<textarea name="entry" rows="6"
cols="64">This band is totally
awesome!</textarea>
```



Official contest entry:

Tell us why you love the band. Five winners will get backstage passes!

The band is totally awesome!

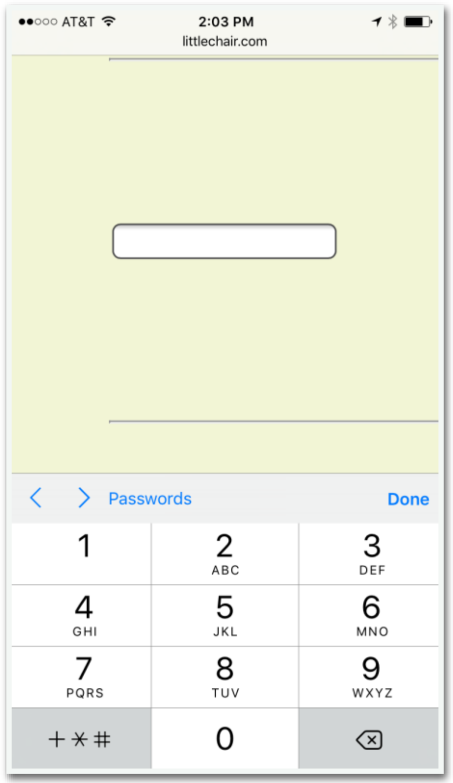
Specialized Text Entry Fields

```
<input type="search">
<input type="email">
<input type="tel">
<input type="url">
```

- These input types are more semantically rich than a default text field.
- Browsers may provide keyboards tailored to the input type.
- Browsers may validate entries on the fly without using the server application.

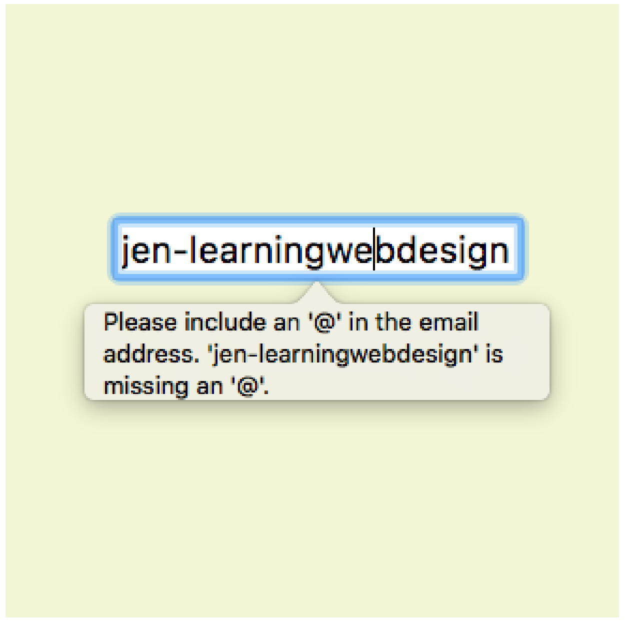
Specialized Text Entries (cont'd)

```
<input type="tel" name="">
```



Numerical keyboard provided on iOS

```
<input type="email" name="">
```



Please include an '@' in the email address. 'jen-learningwebdesign' is missing an '@'.

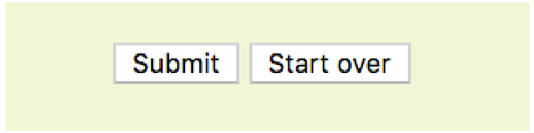
Opera looks for email address structure

FORM CONTROL ELEMENTS

Submit and Reset Buttons

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

- **Submits** the collected form data to the server. Does not require a variable name (**`name`** attribute):



```
<input type="reset" value="Start over">
```

- **Resets** the form to its defaults
- Less common with the rise of JavaScript for form handling
- Change the button text with the **`value`** attribute.

FORM CONTROL ELEMENTS

Custom Buttons

`<button> </button>`

The **`button`** element is used for creating custom buttons with JavaScript.

`<input type="button">`

Creates a custom button that has no predefined function and can be customized with JavaScript

`<input type="image" alt="">`

Allows an image to be used as a button to replace the Submit button. It requires a descriptive **`alt`** attribute value.

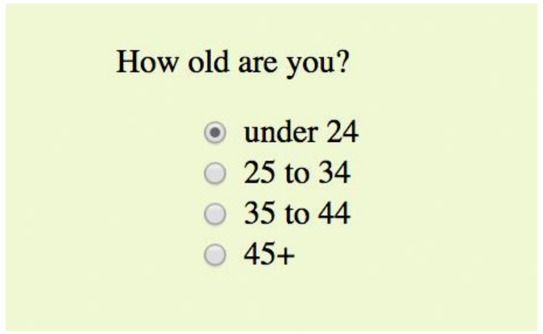
FORM CONTROL ELEMENTS

Radio Buttons

```
<input type="radio">
```

- Only one radio button may be selected at a time.

```
<p>How old are you?</p>
<ol>
 <li><input type="radio" name="age" value="under24" checked> under 24</li>
 <li><input type="radio" name="age" value="25-34"> 25 to 34</li>
 <li><input type="radio" name="age" value="35-44"> 35 to 44</li>
 <li><input type="radio" name="age" value="over45"> 45+</li>
</ol>
```


How old are you?

[✓ under 24]
25 to 34
35 to 44
45+

NOTE: You can't belong to more than one age group, so radio buttons are the right choice for this list.

Radio Buttons (cont'd.)

```
<input type="radio" value="">
```

- Applying the same variable name to input elements binds them together as a mutually exclusive set of options.
- The value for each button must be provided with the **`value`** attribute.
- The **`checked`** attribute causes an option to be selected when the page loads. Only one input may be checked.

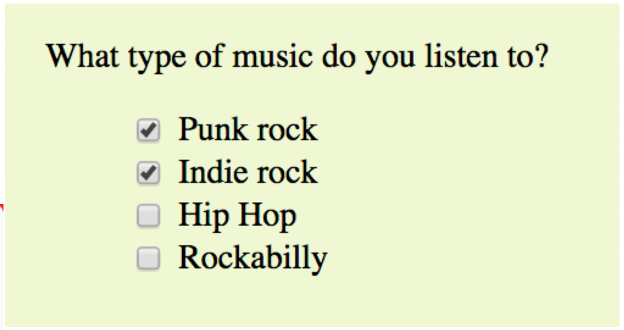
FORM CONTROL ELEMENTS

Checkbox Buttons

`<input type="checkbox">`

- Multiple checkbox buttons may be selected.

```
<p>What type of music do you listen to?</p>
<ul>
  <li><input type="checkbox" name="genre" value="punk" checked> Punk rock</li>
  <li><input type="checkbox" name="genre" value="indie" checked> Indie rock</li>
  <li><input type="checkbox" name="genre" value="hiphop"> Hip Hop</li>
  <li><input type="checkbox" name="genre" value="rockabilly"> Rockabilly</li>
</ul>
```


What type of music do you listen to?

- ✓ Punk rock
- ✓ Indie rock
- ☐ Hip Hop
- ☐ Rockabilly

NOTE: You can like more than one type of music, so checkbox buttons are the right choice for this list.

Checkbox Buttons (cont'd)

```
<input type="checkbox" value="">
```

- Applying the same variable name to input elements binds them together as a group.
- The value for each button must be provided with the **`value`** attribute.
- The **`checked`** attribute causes an option to be selected when the page loads. Multiple buttons in a group may be checked.

FORM CONTROL ELEMENTS

Drop-down Menus

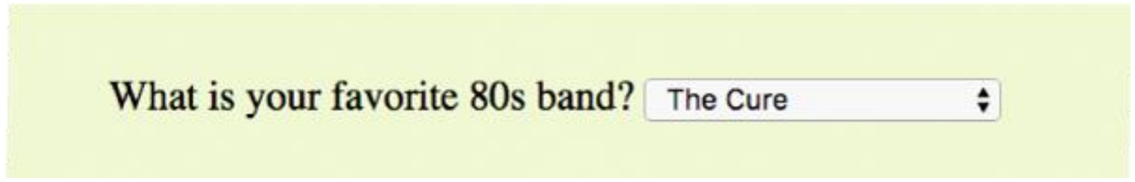
```
<select> </select>
<option> </option>
<optgroup> </optgroup>
```

- The **select** element creates a drop-down menu when there is no **size** attribute (or if `size="1"`).
- The **select** element contains some number of **option** elements.
- The content of the **option** element is the value sent to the server (or one can be provided with the **value** attribute).

Drop-down Menus (cont'd.)

- The select menu drops down to reveal options when the user clicks on it.

```
<p>What is your favorite 80s band?
<select name="EightiesFave">
   <option>The Cure</option>
   <option>Cocteau Twins</option>
   <option>Tears for Fears</option>
   <option>Thompson Twins</option>
   <option value="EBTG">Everything But the Girl</option>
   <option>Depeche Mode</option>
   <option>The Smiths</option>
   <option>New Order</option>
</select>
</p>
```

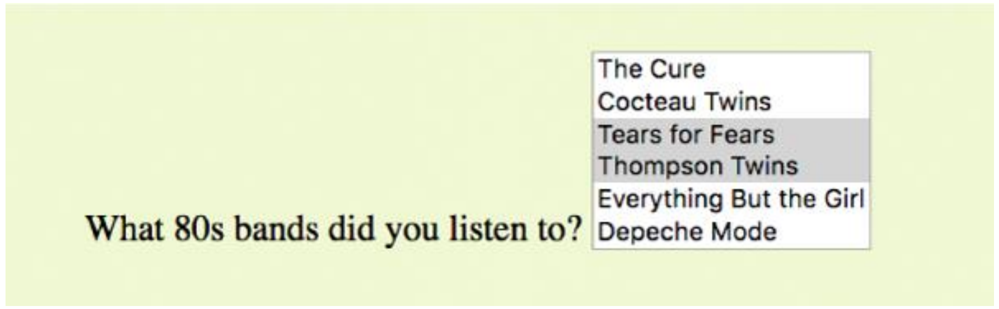


FORM CONTROL ELEMENTS

Scrolling Menus

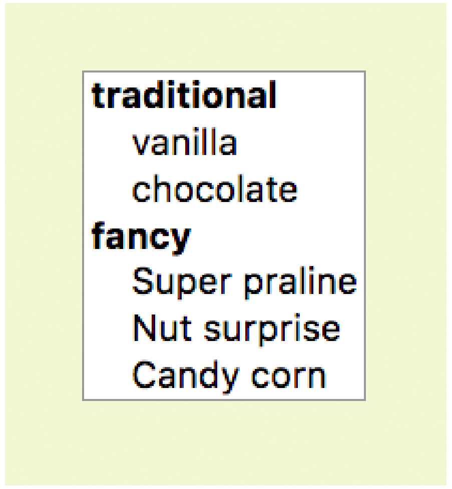
- The same markup as drop-down menus, but the **`size`** attribute specifies how many lines to display.
- The **`multiple`** attribute allows more than one option to be selected.

```
<p>What is your favorite 80s band?
<select name="EightiesFave" size="6" multiple>
   <option>The Cure</option>
   ...
</select>
</p>
```



Scrolling Menus (cont’d)

- Use the **`optgroup`** element to create a conceptual group of options.
- The **`label`** attribute provides the the heading for the group:



```
<select name="icecream" size="7" multiple>
  <optgroup label="traditional">
    <option>vanilla</option>
    <option>chocolate</option>
  </optgroup>
  <optgroup label="fancy">
    <option>Super praline</option>
    <option>Nut surprise</option>
    <option>Candy corn</option>
  </optgroup>
</select>
```

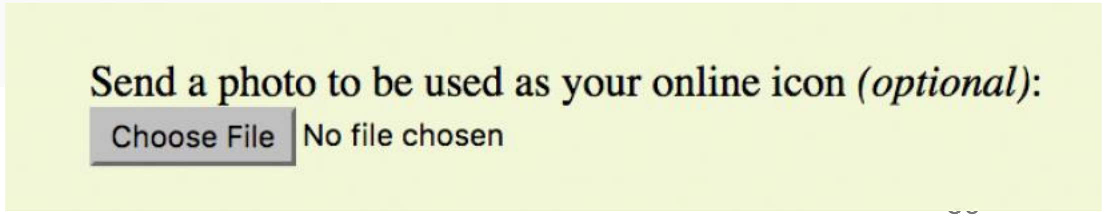
FORM CONTROL ELEMENTS

File Upload Control

```
<input type="file">
```

- The file input type allows a user to select a document from their hard drive to be submitted with the form data.
- The method must be set to POST, and the encoding type must be included.

```
<form action="/client.php" method="POST" enctype="multipart/form-data">
    <label>Send a photo to be used as your online   icon <em>(optional)</em><br>
    <input type="file" name="photo"></label>
</form>
```

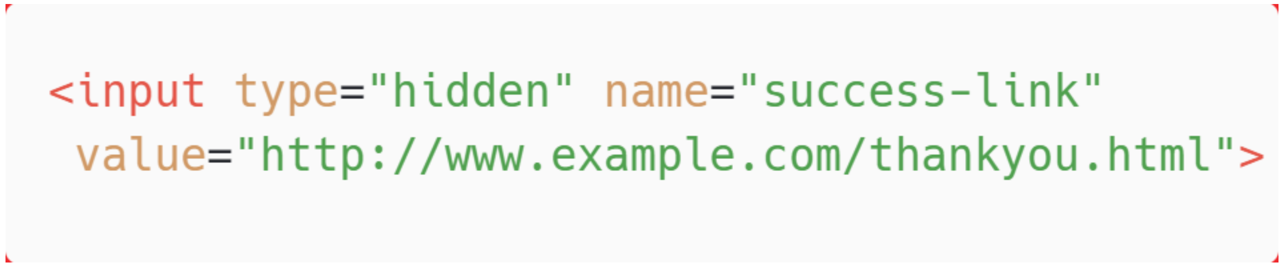

Send a photo to be used as your online icon *(optional)*:

FORM CONTROL ELEMENTS

Hidden Control

```
<input type="hidden">
```

- Sometimes it is necessary to feed values to the processing script/app that don’t come from the user.
- Hidden controls don’t display in the browser.



```
<input type="hidden" name="success-link"
 value="http://www.example.com/thankyou.html">
```

FORM CONTROL ELEMENTS

Date and Time Controls

```
<input type="date">
<input type="time">
<input type="datetime-local">
<input type="month">
<input type="week">
```

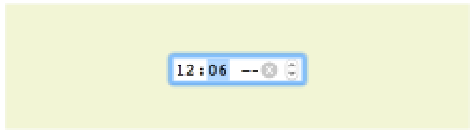
- A starting value may be provided in standard date-time format.

```
<input type="date" name="birthday" value="2017-01-14">
```

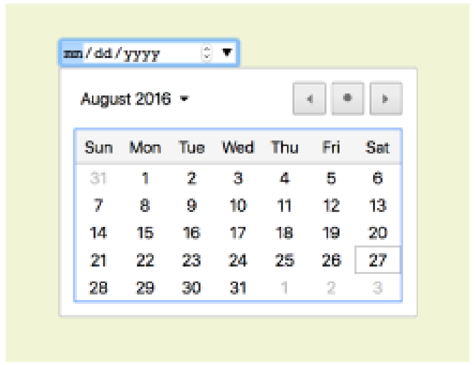
Date and Time Controls (cont’d)

- Browsers may display date and time selection widgets (not universally supported).
- On non-supporting browsers, date and time inputs display as usable text-entry fields.

input type="time"

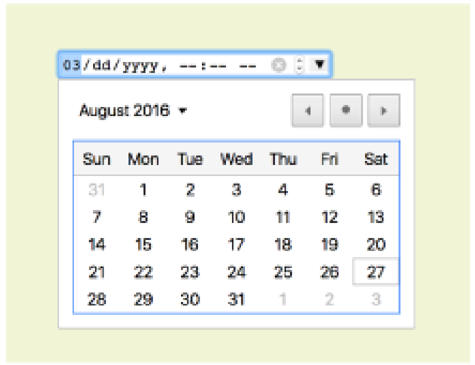


input type="date"



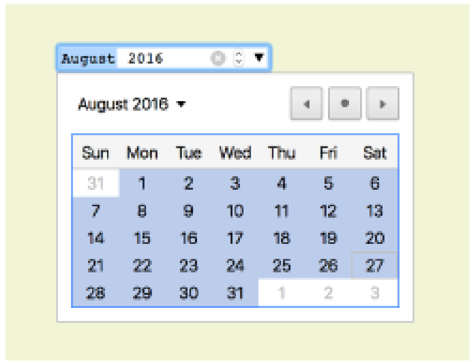
Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

input type="datetime-local"



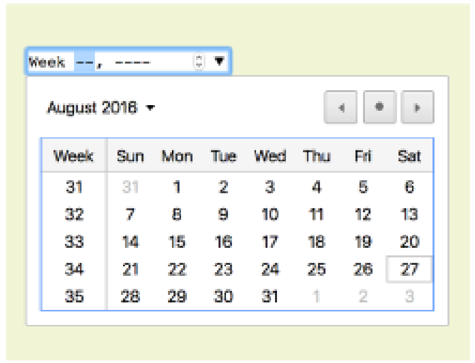
Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

input type="month"



Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	1	2	3

input type="week"



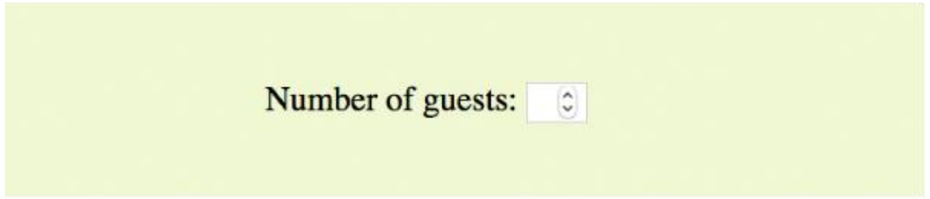
Week	Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	31	1	2	3	4	5	6
32	7	8	9	10	11	12	13
33	14	15	16	17	18	19	20
34	21	22	23	24	25	26	27
35	28	29	30	31	1	2	3

Numerical Controls

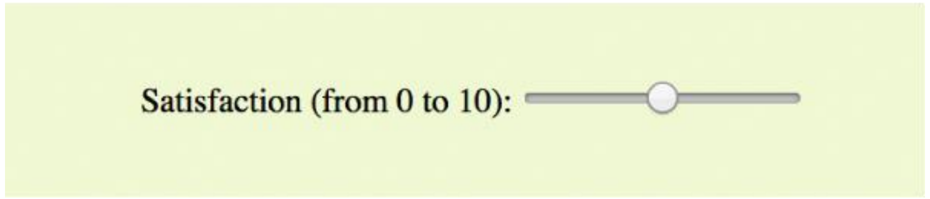
- Number and range controls collect numerical data. Browsers may render counter or slider widgets.
- Both types accept **`min`** and **`max`** attributes for setting limits on the allowed values.

```
<input type="number">
<input type="range">
```

input type="number"



input type="range"

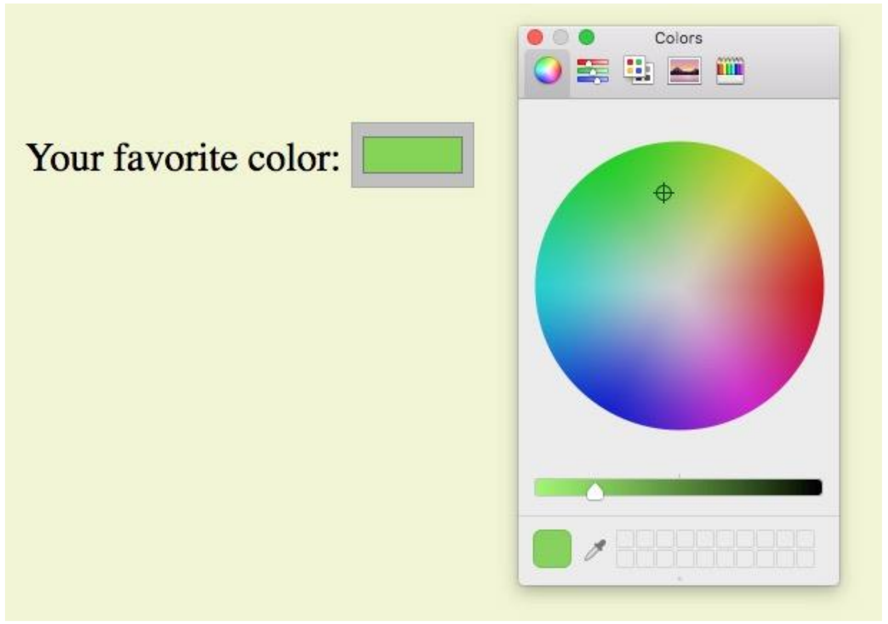


FORM CONTROL ELEMENTS

Color Selector

```
<input type="color">
```

- The color input type is intended to provide a pop-up color picker.
- It is not well supported. Non-supporting browsers display a text-entry field.



Form Accessibility

- Users may not be able to see the form. They may be listening to it with a screen reader.
- Whereas sighted users can see at a glance how elements are organized, form accessibility features create semantic connections between form components.

FORM ACCESSIBILITY

Labels

`<label> </label>`

The **`label`** element associates a descriptive label with a form field.

Implicit association
The label text and form control are both contained within the **`label`** element:

```
<li><label>Red <input type="radio" name="color" value="red"></label></li>
```

Explicit association
Matches the label with the control's ID reference using the **`for`** attribute:

```
<li><label for="form-colors-red">Red</label>
  <input type="radio" name="color" value="red id="form-colors-red"></li>
```

FORM ACCESSIBILITY

Fieldsets and Legends

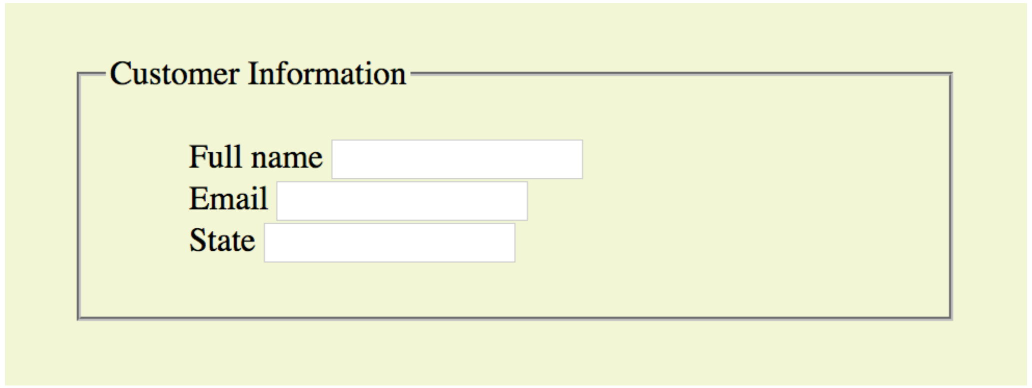
```
<fieldset> </fieldset>
<legend> </legend>
```

`fieldset`
Indicates a logical grouping of controls (examples: credit card name, number, date, etc.). By default, rendered with a box around the set of controls.

`legend`
Provides a caption for the enclosed fields. By default, it’s displayed along the top edge of the fieldset.

Fieldsets and Legends (cont'd)

```html
<fieldset>
  <legend>Customer Information</legend>
  <ul>
    <li><label>Full name: <input type="text" name="fullname"></label></li>
    <li><label>Email: <input type="text" name="email"></label></li>
    <li><label>State: <input type="text" name="state"></label></li>
  </ul>
</fieldset>
```


Customer Information

Full name
Email
State

Form Design Tips

- Avoid unnecessary questions.
- Consider the impact of label placement. Labels above fields tend to lead to faster completion.
- Choose input types carefully.
- Group related inputs.
- Primary actions (e.g., “Buy”) should be visually dominant to secondary actions (e.g., “Back”).

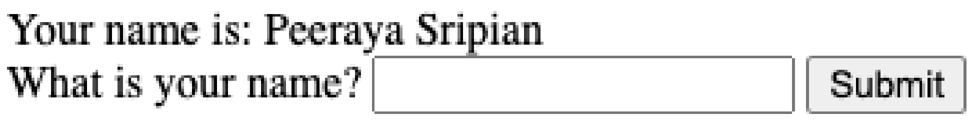
Submitting data

Data handling

```php
<?php // formtest.php
if (isset($_POST['name']))
  $name = $_POST['name'];
else
  $name = "(Not entered)";
echo <<<_END
<html>
  <head>
    <title>Form Test</title>
  </head>
  <body>
    Your name is: $name<br>
    <form method="post" action="formtest.php">
      What is your name?
      <input type="text" name="name">
      <input type="submit">
    </form>
  </body>
</html>
_END;
```

```
echo <<<_END
Some HTML
_END;
```

Can use whenever you want to output multiline of HTML



Your name is: Peeraya Sripian
What is your name?

The **$_POST** Array

- A browser sends user input through either **GET** or **POST** request
- The POST request is usually preferred
- The web server bundles up all user input and puts in array **$_POST**
- To access this array, use the element key (ex: “isbn”)
  - **$_POST['isbn']** or **$_POST[“isbn”]**
- All values in **$_POST** will be retrieved at the beginning of the program and ignored after that

Sanitizing Input

- Handling user data is a security minefield – essential to learn to treat all such data with the utmost caution
- Sanitize user input to prevent potential hacking attempts such as
  - Hacker can just use View Source jeopardize with your form to receive malicious input to your website
  - JavaScript or MySQL script injection
- Instead of just read user input with this code

```
$variable = $_POST['user_input’];
```

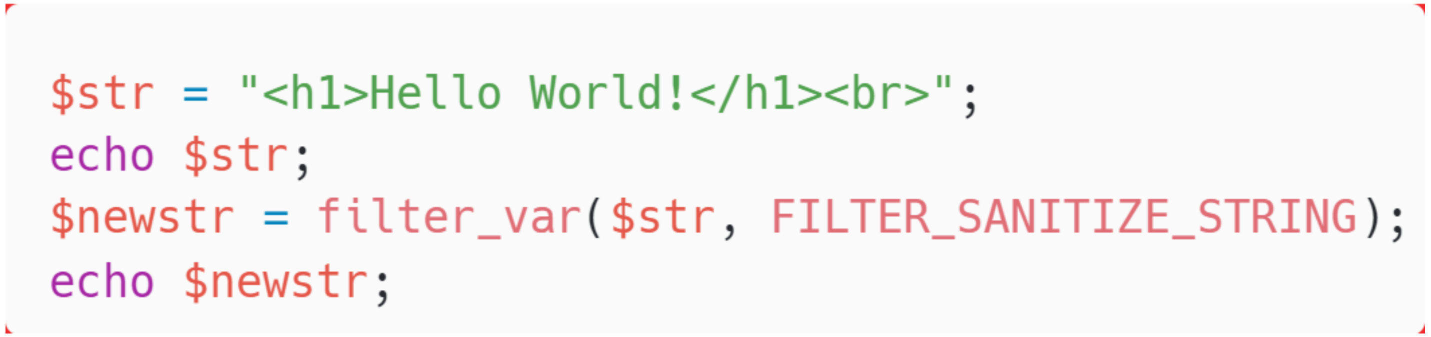
- It is better to use PHP Filter to sanitize input

PHP Filters

- Validate and sanitize external input
- PHP filter extension has many functions needed for checking user input, easy data validation
- External input/data are
  - User input from a HTML form
  - Cookies
  - Web services data
  - Server variables
  - Database query results
- You should always validate external data!

PHP filter_var() Function

- `filter_var()` filters a single variable with a specified filter
- Example: remove all HTML tags from a string



```php
$str = "<h1>Hello World!</h1><br>";
echo $str;
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
```

Output

```
Hello World!
```

DEMO

Sanitize and validate an email address

```
<!DOCTYPE html>
<html>
<body>

<?php
$email = "john.do**e@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
  echo("$email is a valid email address");
} else {
  echo("$email is not a valid email address");
}
?>

</body>
</html>
```

When `$email = john.doe@example.com`
Output:
```
john.doe@example.com is a valid email address
```

When `$email = john.do**e@example.com`
Output:
```
john.do**e@example.com is not a valid email address
```

More functions in PHP filter: https://www.w3schools.com/php/php_ref_filter.asp

Validating User Input with JavaScript

- JavaScript validation – more like an assistance to users, not to your websites
- It should use for checking that fields do not left empty, or format of email address, or values within expected bounds

- See provided code `validate.html`

DEMO

Self-processing pages

- One PHP page can be used to both generate a form and subsequently process it
- This example request with the GET method, prints a form that accept a temp (F)
- If called with POST, the page calculates and display temp (C)

```
<html>

<head>
    <title>Temperature Conversion</title>
</head>

<body>
    <?php
    if ($_SERVER['REQUEST_METHOD'] == 'GET') {
    ?>
        <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST">
            Fahrenheit temperature:
            <input type="text" name="fahrenheit" /><br />
            <input type="submit" value="Convert to Celsius!" />
        </form>
    <?php
    } else if ($_SERVER['REQUEST_METHOD'] == 'POST') {
        $fahrenheit = $_POST['fahrenheit'];
        $celsius = ($fahrenheit - 32) * 5 / 9;
        printf("%.2fF is %.2fC", $fahrenheit, $celsius);
    } else {
        die("This script only works with GET and POST requests.");
    } ?>
</body>

</html>
```

DEMO

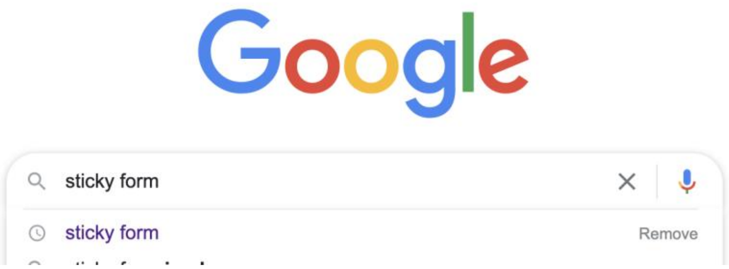
Self-processing pages

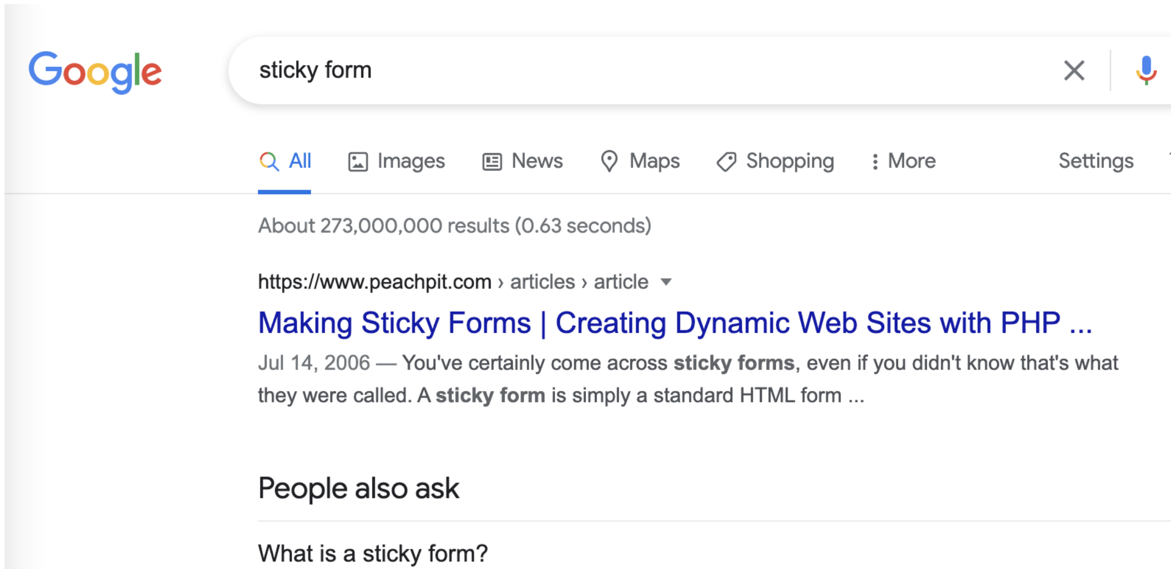
- Another way to decide whether to display a form or process is to check whether parameters are supplied or not (**`isset`**)
- This example page use GET to submit values and it uses the presence or absence of parameters to determine what to do
- The parameter values are copied to **`$farenheit`**,
- We can also use **`is_null`** to check **`$farenheit`** before display the form or process the form data

```
<html>

<head>
    <title>Temperature Conversion</title>
</head>

<body>
    <?php
    if (isset($_GET['fahrenheit'])) {
        $fahrenheit = $_GET['fahrenheit'];
    } else {
        $fahrenheit = null;
    }
    if (is_null($fahrenheit)) {
    ?>
        <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
            Fahrenheit temperature: <input type="text" name="fahrenheit" /><br />
            <input type="submit" value="Convert to Celsius!" />
        </form>
    <?php
    } else {
        $celsius = ($fahrenheit - 32) * 5 / 9;
        printf("%.2fF is %.2fC", $fahrenheit, $celsius);
    }
    ?>
</body>

</html>
```

Sticky forms

- Many website use this technique
- The results of a query are accompanied by a search form (whose default values are those of the previous query)
- Example: google search





About 273,000,000 results (0.63 seconds)

https://www.peachpit.com › articles › article

Making Sticky Forms | Creating Dynamic Web Sites with PHP ...

Jul 14, 2006 — You've certainly come across **sticky forms**, even if you didn't know that's what they were called. A **sticky form** is simply a standard HTML form ...

People also ask

What is a sticky form?

Sticky forms - implementation

```
<html>

<head>
    <title>Temperature Conversion</title>
</head>

<body>

<?php error_reporting (E_ALL ^E_NOTICE);
//this is to stop the notice generated by PHP for undefined index
?>

    <?php $fahrenheit = $_GET['fahrenheit']; ?>

    <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
        Fahrenheit temperature:
        <input type="text" name="fahrenheit" value="<?php echo $fahrenheit; ?>">
<br>
        <input type="submit" value="Convert to Celsius!">
    </form>

    <?php
    if (!is_null($fahrenheit)) {
        $celsius = ($fahrenheit - 32) * 5 / 9;
        printf("%.2fF is %.2fC", $fahrenheit, $celsius);
    } ?>

</body>

</html>
```

End of the topic

End of class 7

Next week: June, 13

PHP Fundamental 2

To do from now

- Check Github classroom for this week’s assignment explanation
- Demo of how to submit your files

  *Different from previous weeks from now on