Introduction to PHP

Let's Look Back

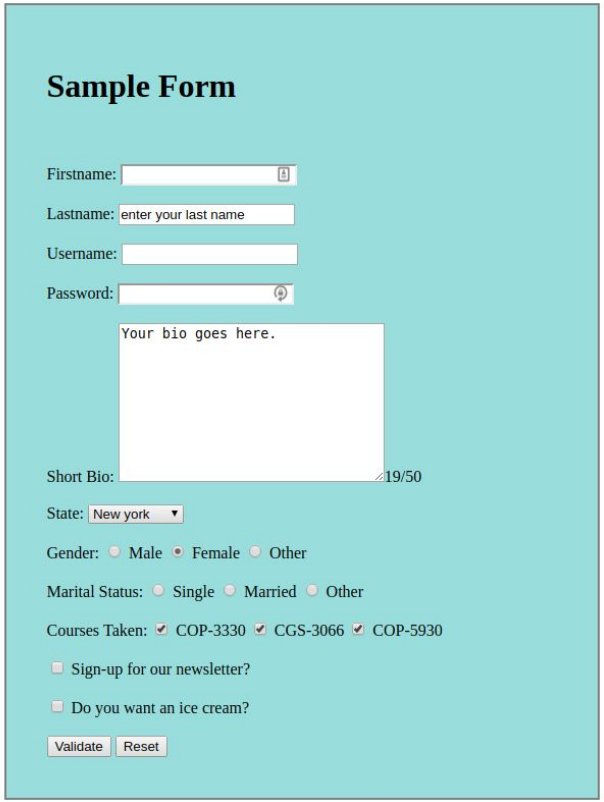
We talked about how to create a form in HTML

Forms are one way to interact with users

Users can enter information into forms which can be used by you (programmer)

We also talked about how to validate these input data by using javascript on client-side

But, what about storing and using data?



PHP

Originally, PHP stood for Personal Home Page. The acronym doesn't mean much anymore, but officially it is called PHP: Hypertext Processor

PHP is a **server-side** processing language. The server reads the PHP code and outputs browser-friendly HTML

With PHP you can **store data on the server** as **files** or in **databases** like MySQL.

With PHP, you can write code once and use it everywhere. Remember, you want DRY code (Don't Repeat Yourself).

PHP lets you build **dynamic webpages** that respond to input from users.

PHP

PHP runs on different platforms (Windows, Linux, Unix, etc.)

PHP is compatible with almost all servers used today (Apache, IIS, etc.)

PHP is FREE to download and use.

PHP is easy to learn and runs efficiently on the server side

How to use PHP

In order to use PHP or any other server-side tool, you have two options:

- Find a web host with PHP or other server-side tool support
- Install a web server on your own PC, and then install PHP and other tools

How to use PHP

In order to use PHP or any other server-side tool, you have two options:

- Find a web host with PHP or other server-side tool support
- Install a web server on your own PC, and then install PHP and other tools

CS department has PHP installed on ww2 server and you can use that. I have posted a guide on how to do that on the course website.

How to use PHP

In order to use PHP or any other server-side tool, you have two options:

- Find a web host with PHP or other server-side tool support
- Install a web server on your own PC, and then install PHP and other tools

On windows, you can download and install WAMP. With one installation and you get an Apache webserver, database server and php.

On mac, you can download and install MAMP.

PHP Basics

The PHP code is enclosed in special start and end processing instructions `<?php` and `?>` that allow you to jump into and out of "PHP mode."

PHP code is executed on the server, generating HTML which is then sent to the client. The client would receive the results of running that script, but would not know what the underlying code was.

PHP: Hello World

index.php on the server

```
<!doctype html>
<html>
<head>
<title>PHP Hello World!</title>
</head>
<body>
<?php
  echo "Hello World!";
  print "<br>";
  echo "Welcome!";
?>
</body>
</html>
```

What shows up in the browser

Hello World!
Welcome!

Client sees this as source of the page

```
<!doctype html>
<html>
<head>
<title>PHP Hello World!</title>
</head>
<body>
Hello World!<br>Welcome!
</body>
</html>
```

PHP: Hello World

This program is extremely simple and you really did not need to use PHP to create a page like this. All it does is display: Hello World and Welcome using the PHP `echo` and `print` statements.

Think of this as a normal HTML file which happens to have a set of special tags available to you that do a lot of interesting things.

PHP Comments

In PHP, we use `//` to make a **single-line comment** or `/*` and `*/` to make a large comment block. Just similar to javascript

```
<!doctype html>
<html>
<body>
<?php
  // This is a comment
  /*
    This is
    a comment block
  */
?>
</body>
</html>
```

PHP Variables

A variable is a place to store values

When you first create a variable, it does not have a value (it is null).

You can set a value for a variable.

Variables can hold different types of information, like words, numbers, and collections of data.

The value of a variable can change over time.

PHP Variables

In PHP, you write a variable with a dollar sign followed by the name of the variable.

The variable name is case-sensitive.

A new variable needs to have a unique name.

Variable names need to start with a letter or underscore.

Variable names can only be made of letters and numbers.

PHP Variables

To create a variable, just type a dollar sign and the variable name. PHP will create the variable for you.

```
$age;
```

It is a good idea to give your variable a starting value. This is called initializing the variable.

```
$age = 5;
```

Once you have created a variable, you can use it in your code. Just type a dollar sign and the name of the variable

```
$age = 5;
echo $age;
```

PHP Variable Values: Numbers

Variables can be numbers, either integers or floats (decimals).

```
$numberOfCoffees = 4;
$coffeePrice = 2.3;
```

PHP will automatically convert integers to floats if needed.

Once you have numbers, you can do math with them!

```
$totalPrice = $numberOfCoffees * $coffeePrice;
```

PHP Arithmetic Operators

Example	Name	Result
-$a	Negation	Opposite of $a
$a + $b	Addition	Sum of $a and $b
$a - $b	Subtraction	Difference of $a and $b
$a * $b	Multiplication	Product of $a and $b
$a / $b	Division	Quotient of $a and $b
$a % $b	Modulus	Remainder of $a divided by $b

PHP Variable Values: Strings

Variables can be strings(groups of characters). You put your string in quotes.

```
$name = 'Fluffy';
```

If you want to use a quote in your string, you'll need to "escape" it with a backslash.

```
echo 'I\'d like to use an apostrophe';
```

PHP String Operators

You can put strings together with a period, the concatenation operator.

```
$firstName = 'Fluffy';
$fullName = $firstName . ' McDougle';
echo $fullName; //Outputs 'Fluffy McDougle'
```

You can also use .= to add things to the end of a string.

```
$name = 'Fluffy';
$name .= ' McDougle';
echo $name; //Outputs 'Fluffy McDougle'
```

PHP Arrays

Let's assume we have a list of items (car names). If we wanted to store these values in single variables we could do something like this:

```
$car1 = 'BMW';
$car2 = 'Nissan';
$car3 = 'Honda';
```

But what if you had 300 values to be stored? Is there a better way to do this?

PHP Arrays

The best solution here is to use an array.

An array can hold all your variable values under a single name. And you can access the values by referring to the array name.

Each element in the array has its own index so that it can be easily accessed.

You can define the array of cars in two ways

PHP Numeric Arrays

In the following example, the index is automatically assigned(starting from 0)

```
$cars = array("BMW","Nissan","Honda","Toyota");
```

In the following example we assign the index manually:

```
$cars[0] = "BMW";
$cars[1] = "Nissan";
$cars[2] = "Honda";
$cars[3] = "Toyota";
```

PHP Numeric Arrays

Regardless of the way we create the array, we can access individual elements by using their index:

```
<!doctype html>
<html>
<body>
<?php
  $cars = array("BMW","Nissan","Honda","Toyota");
  echo $cars[1] . " and " . $cars[2] . " are Japanese cars";
?>
</body>
</html>
```

Output: Nissan and Honda are Japanese cars.

PHP Associative Arrays

With 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 keys and assign values to them.

PHP Associative Arrays

In the following example, we use an array to assign ages to different people:

```
$ages = array("Peter"=>32,"Nancy"=>30,"Joe"=>44);
```

We could also do the same thing as above, in this way:

```
$ages['Peter'] = 32;
$ages['Nancy'] = 30;
$ages['Joe'] = 44;
```

PHP Associative Arrays

The ID keys can be used to access the values associated with them:

```
<!doctype html>
<html>
<body>
<?php
  $ages['Peter'] = 32;
  $ages['Nancy'] = 30;
  $ages['Joe'] = 44;
  echo "Peter is " . $ages['Peter'] . " years old.";
?>
</body>
</html>
```

Output: Peter is 32 years old.

PHP Operators

Arithmetic Operators

Operator	Description	Example	Result
+	Addition	x=2 x+2	4
-	Subtraction	x=2 5-x	3
*	Multiplication	x=4 x*5	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

PHP Operators

Assignment Operators

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y
%=	x%=y	x=x%y

PHP Operators

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
!=	is not equal	5!=8 returns true
<>	is not equal	5<>8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

PHP Operators

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different decisions.

You can use conditional statements in your code to do this.

In PHP we have the following conditional statements...

PHP Conditional Statements

if statement - use this statement to execute some code only if a specified condition is true

```
<!doctype html>
<html>
<body>
<?php
  $d = date("D");
  if ($d == "Fri")
    echo "Have a nice weekend!";
?>
</body>
</html>
```

PHP Conditional Statements

if...else statement - use this statement to execute some code if a condition is true and another code if the condition is false

```
<!doctype html>
<html>
<body>
<?php
  $d = date("D");
  if ($d == "Fri")
    echo "Have a nice weekend!";
  else
    echo "Have a nice day!";
?>
</body>
</html>
```

PHP Conditional Statements

if...elseif....else statement - use this statement to select one of several blocks of code to be executed

```
<!doctype html>
<html>
<body>
<?php
  $d = date("D");
  if ($d == "Fri")
    echo "Have a nice weekend!";
  elseif($d == "Sun")
    echo "Have a nice Sunday!";
  else
    echo "Have a nice day!";
?>
</body>
</html>
```

PHP Conditional Statements

If more than one line should be executed if a condition is true/false, the lines should be enclosed within curly braces { }

```
<!doctype html>
<html>
<body>
<?php
  $d = date("D");
  if ($d == "Fri") {
    echo "Have a nice weekend!";
    echo "See you on Monday!";
  }
?>
</body>
</html>
```

PHP Loops

Often when you write code, you want the same block of code to run over and over again in a row.

Instead of adding several almost equal lines in a script we can use loops to perform a task like this.

In PHP and other languages, we have several looping statements

PHP Loops

while - loops through a block of code while a specified condition is true

```
<!doctype html>
<html>
<body>
<?php
  $i = 1;
  while ($i <= 5) {
    echo "The number is " . $i . "<br>";
    $i++;
  }
?>
</body>
</html>
```

The loop starts with i=1. It will continue to run as long as i is less than, or equal to 5. It will increase by 1 each time the loop runs.

Output:
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

PHP Loops

for - loops through a block of code a specified number of times

```
for (init; condition; increment)
{
   code to be executed;
}
```

init: Mostly used to set a counter (but can be any code to be executed once at the beginning of the loop)

condition: Evaluated for each loop iteration. If it evaluates to TRUE, the loop continues. If it evaluates to FALSE, the loop ends.

increment: Mostly used to increment a counter (but can be any code to be executed at the end of the loop)

PHP Loops

for - loops through a block of code a specified number of times

```
<!doctype html>
<html>
<body>
<?php
  for ($i=1; $i<=5; $i++) {
    echo "The number is " . $i . "<br>";
  }
?>
</body>
</html>
```

The loop starts with i=1. It will continue to run as long as i is less than, or equal to 5. It will increase by 1 each time the loop runs.

Output:
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5

PHP Loops

foreach - loops through a block of code for each element in an array

```
<!doctype html>
<html>
<body>
<?php
  $cars = array("BMW","Nissan","Honda","Toyota");
  foreach ($cars as $c) {
    echo $c . "<br>";
  }
?>
</body>
</html>
```

This code will loop through values in the array and print them out.

Output:
BMW
Nissan
Honda
Toyota

PHP Loops

foreach - loops through a block of code for each element in an array

```
<!doctype html>
<html>
<body>
<?php
  $ages = array("Peter"=>32,"Nancy"=>30,"Joe"=>44);

  foreach ($ages as $person => $age) {
    echo $person . " is " . $age . " years old."
    echo "<br>";
  }
?>
</body>
</html>
```

This code will loop through keys and values in an associative array and print them out.

Output:
Peter is 32 years old.
Nancy is 30 years old.
Joe is 44 years old.

PHP Functions

Functions are separable, reusable pieces of code.

To use a function:

- first declare the function

```
function callMeByMyName()
{
   echo 'Hey Peyman!';
}
```

- Then call it as many times as you want

```
callMeByMyName();
```

PHP Function Parameters

To add more functionality to a function, we can add parameters. A parameter is just like a variable.

Parameters are specified after the function name, inside the parentheses.

```
<!doctype html>
<html>
<body>
<?php
  function callMeByMyName($name)
  {
     echo 'Hey ' . $name;
  }

  callMeByMyName('Peter');

  $someName = 'Donald';
  callMeByMyName($someName);
?>
</body>
</html>
```

Output:
Hey Peter
Hey Donald

PHP Function Return Values

You can have a function give you back a value, to use later.

Return will immediately end a function.

```
<!doctype html>
<html>
<body>
<?php
  function square($num) {
    return $num * $num;
  }

  echo square(4);

  $squareOfFive = square(5);
  echo square($squareOfFive);
?>
</body>
</html>
```

Output:
16
625