The Basics

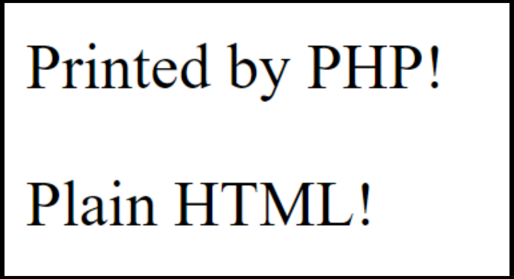
Intro PHP

Download the code:
No Code Samples for the Slides

Open and Close Tags

- PHP FIles (generally) use .php file extension:
  - index.php
- PHP can be embedded in HTML templates
- Embedded PHP is placed between <?php ?> tags
- All statements end with a semi-colon;

```
<?php
echo "<p>Printed by PHP!</p>";
?>

<p>Plain HTML!</p>
```


Printed by PHP!

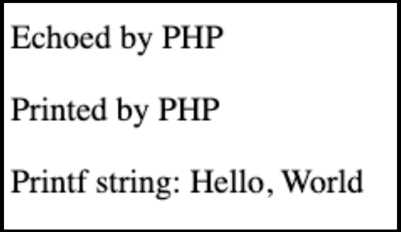
Plain HTML!

01_tags.php

Output

- Output intended for the user can be accomplished in one of three ways.
    - echo – a language structure
    - print() - a function
    - printf() – a function for printing formatted strings

```
echo "<p>Echoed by PHP</p>";

print("<p>Printed by PHP</p>");

printf('Formatted string: Hello, %s', 'World');
```



```
Echoed by PHP

Printed by PHP

Printf string: Hello, World
```

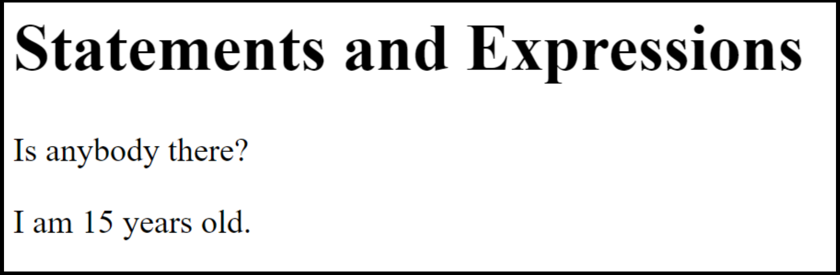
sprintf() is also available... but it does not print out directly. Instead, it returns a formatted string that you can print later.

See all files for output examples

Statements and Expressions

- Expressions resolve to a value
- Statements do not resolve to a value

```php
echo '<p>Is anybody there?</p>';

$num1 = 12;
$num2 = 3;
$age = $num1 + $num2;

echo '<p>I am ' . $age .
' years old.</p>';
```



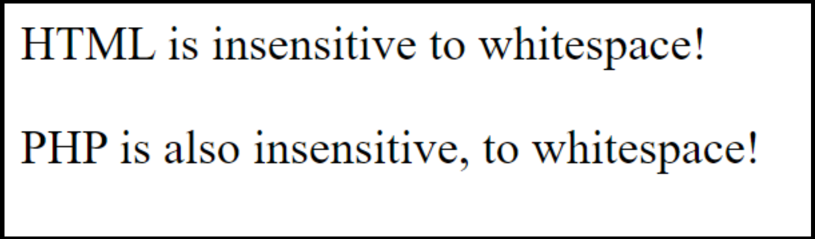
```
Statements and Expressions

Is anybody there?

I am 15 years old.
```

02_statements.php

Whitespace means nothing

- As in HTML, whitespace has no meaning

```
<p>HTML is insensitive

to whitespace!</p>

<?php

echo "<p>PHP is also insensitive, ";




echo "to whitespace!</p>";

?>
```

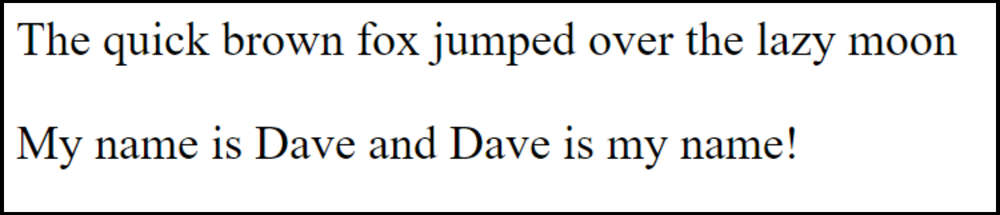


HTML is insensitive to whitespace!

PHP is also insensitive, to whitespace!

03_whitespace.php

Comments



The quick brown fox jumped over the lazy moon

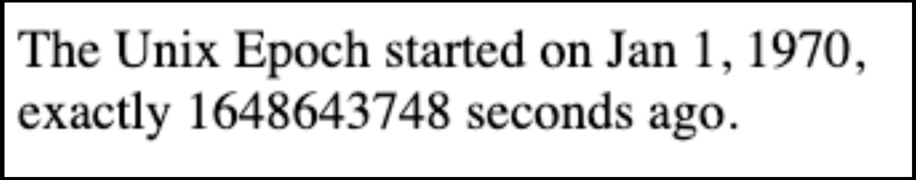
My name is Dave and Dave is my name!

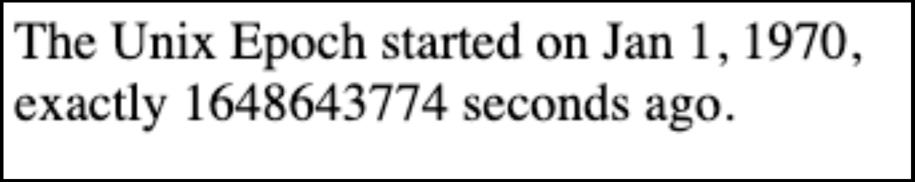
```
/* This is a multiline
comment with a slash asterisk
at the beginning and an
asterisk slash at the end. */

// This is a single line comment
echo '<p>The quick brown fox jumped over the lazy moon</p>';

$name = 'Dave'; // single line comment appended to end of line

echo "<p>My name is " . $name /* inline comment */.
" and " . $name /* another inline comment */. " is my name!</p>"

# This is a bash/c style single line comment
```

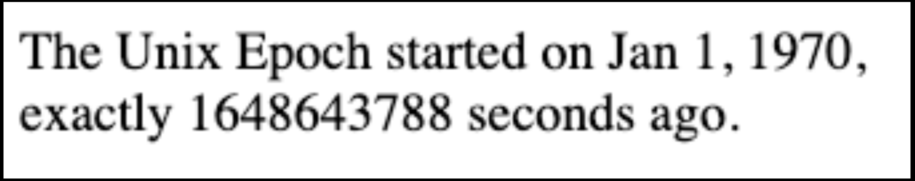
04_comments.php

Dynamic Content

```
<p>The Unix Epoch started on
Jan 1, 1970,<br />
exactly <?php echo time(); ?>
seconds ago.</p>
```


The Unix Epoch started on Jan 1, 1970,
exactly 1648643748 seconds ago.


The Unix Epoch started on Jan 1, 1970,
exactly 1648643774 seconds ago.


The Unix Epoch started on Jan 1, 1970,
exactly 1648643788 seconds ago.

05_dynamic_content.php

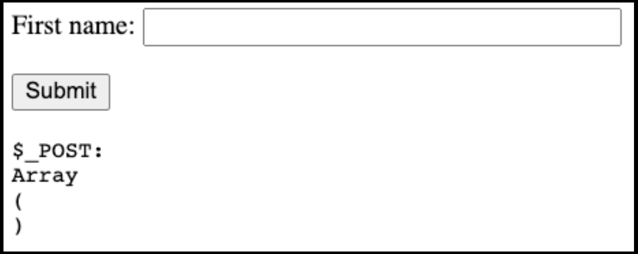
Variables and Data Types

```
$a = 12;
$b = 22.334;
$c = 'Hello';
$d = [1,2,3];
$e = new stdClass();
$f = fopen('test.txt', 'w');
$g = null;
$h = false;
$i = true;
```

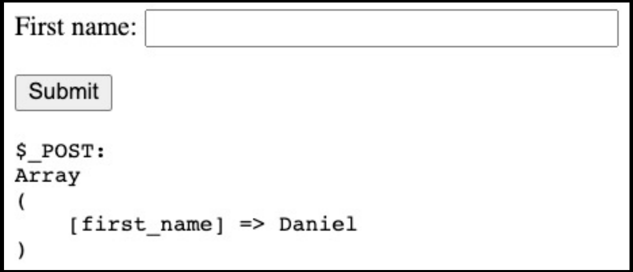
06_variables.php

Form Variables

```
<form method="post">
<p><label for="first_name">First
name</label>:
<input type="text" name="first_name"
size="40" /></p>
<p><input type="submit"></p>
</form>

<pre>
$_REQUEST:
<?php print_r($_REQUEST); ?>
$_POST:
<?php print_r($_POST); ?>
$_GET:
<?php print_r($_GET); ?>
</pre>
```



```
$_POST:
Array
(
)
```



```
$_POST:
Array
(
    [first_name] => Daniel
)
```

Submitted form data will be available in a SuperGlobal array named after the HTTP verb used as the form method: $_GET or $_POST

07_form_variables.php

Beware Type Coercion

If you inadvertently attempt to perform an operation on two types of data, PHP will attempt to coerce one of the values into the correct type.

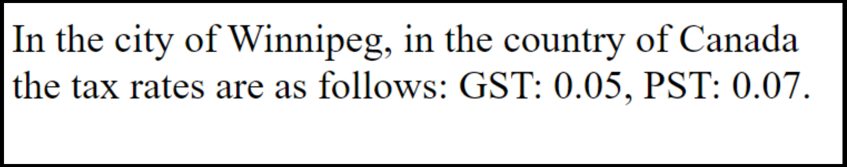
```
$a = '25' + 25;  // 50
$b = '25' . 25;  // 2525
$c = 25 . 25;  // 2525
$d = 25.25;  // 25.25
$g = true + 3;  // 4
$h = 5 - false;  // 5
$i = (3 == 4 - true);  // true
$j = intval(25 . "hello");  // 25
$k = intval("hello" . 25);  // 0
$l = floatval(25 . '25hello’);  // 2525
$m = floatval(25 . '.25hello’);  // 25.25
```

08_type_coercion.php

Constants

As well as mutable variables, PHP supports immutable constants… once set, their value cannot be changed.

```php
<?php
define('CITY', 'Winnipeg');
define('COUNTRY', 'Canada');
define('GST', .05);
define('PST', .07);
?>

<p>In the city of <?php echo CITY; ?>, in the
country of <?php echo COUNTRY; ?> <br />
the tax rates are as follows: GST: <?php echo GST;
?>,
PST: <?php echo PST; ?>.</p>
```



```
In the city of Winnipeg, in the country of Canada
the tax rates are as follows: GST: 0.05, PST: 0.07.
```

09_constants.php

Variable Status – To Be Or Not to Be

PHP has two ways to determine a variable's status:

isset and empty

```
$name = 'Davey';
$friends = [];
$city = '';
```

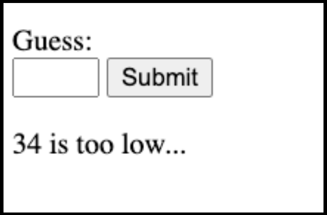
- $name is set, and it is not empty
- $friends is set, but it is empty
- $city is set, but it is empty
- $country is not set, and it is empty

10_variable_status.php

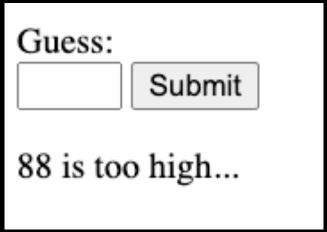
Conditionals – if/else

PHP's if/else conditional structure works exactly the same way as in Javascript.

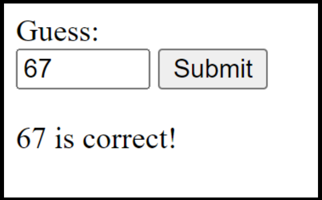
```
$num = 67; // guess this number

if(isset($_POST["guess"])) {

$guess = intval($_POST['guess']);

if($guess < $num) {
echo "<p>" . $guess . " is too low...</p>";
} elseif($guess > $num) {
echo "<p>" . $guess . " is too high...</p>";
} else {
echo "<p>" . $guess . " is correct!</p>";
}

} // endif
```



34 is too low...



88 is too high...



67 is correct!

11_if_else.php

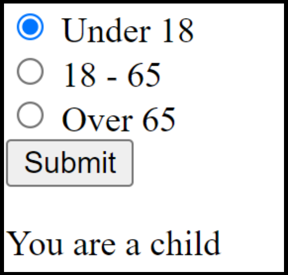
one difference. PHP has an elseif while Javascript uses two keywords: else if

Conditionals – switch/case

PHP’s switch/case conditional structure works exactly the same way as in Javascript.

```
if(isset($_POST['age'])) {

    switch($_POST['age']) {
        case 1:
            echo '<p>You are a child</p>';
            break;
        case 2:
            echo '<p>You are an adult</p>';
            break;
        case 3:
            echo '<p>You are a senior</p>';
            break;
        default:
            echo '<p>You are unborn or
        deceased.</p>';
    }

}
```

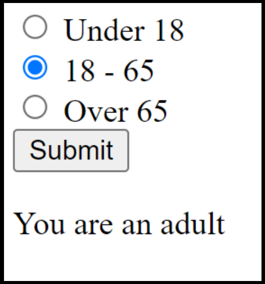
12_switch_case.php



[✓ Under 18] 18 - 65 Over 65

Submit

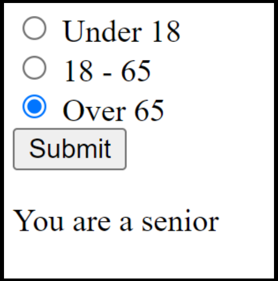
You are a child



Under 18 [✓ 18 - 65] Over 65

Submit

You are an adult



Under 18 18 - 65 [✓ Over 65]

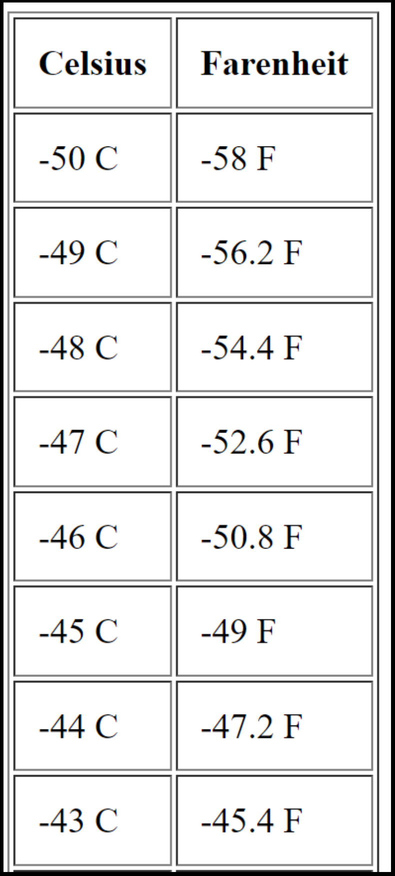
Submit

You are a senior

Loops – For Loop

PHP's for loop is exactly the same as in Javascript

```
for($i=-50;$i<=50;$i++) {
    echo "<tr>";
    echo "<td>" . $i . " C</td>";
    echo "<td>" . ($i * 1.8 + 32) . " F</td>";
    echo "</tr>\n";
}
```



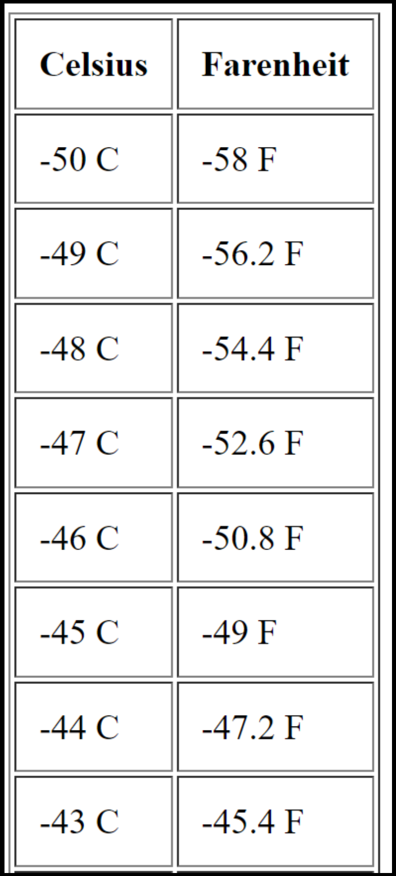
Celsius	Farenheit
-50 C	-58 F
-49 C	-56.2 F
-48 C	-54.4 F
-47 C	-52.6 F
-46 C	-50.8 F
-45 C	-49 F
-44 C	-47.2 F
-43 C	-45.4 F

13_loops1.php

Loops – While Loop

PHP’s while loop is exactly the same as in Javascript

```
$i = -50;
while($i<=50) {
     echo "<tr>";
     echo "<td>" . $i . " C</td>";
     echo "<td>" . ($i * 1.8 + 32) . " F</td>";
     echo "</tr>\n";
     $i++;
}
```



Celsius	Farenheit
-50 C	-58 F
-49 C	-56.2 F
-48 C	-54.4 F
-47 C	-52.6 F
-46 C	-50.8 F
-45 C	-49 F
-44 C	-47.2 F
-43 C	-45.4 F

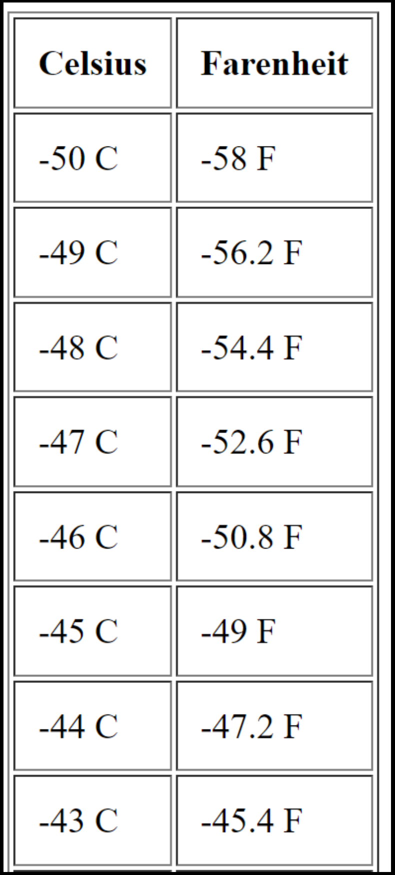
14_loops2.php

Loops – Do While Loop

PHP's do while loop is exactly the same as in Javascript

```
$i = -50;
do {
     echo "<tr>";
     echo "<td>" . $i . " C</td>";
     echo "<td>" . ($i * 1.8 + 32) . " F</td>";
     echo "</tr>\n";
     $i++;
} while($i<=50)
```

15_loops3.php



Celsius	Farenheit
-50 C	-58 F
-49 C	-56.2 F
-48 C	-54.4 F
-47 C	-52.6 F
-46 C	-50.8 F
-45 C	-49 F
-44 C	-47.2 F
-43 C	-45.4 F

Loops – For Each Loop

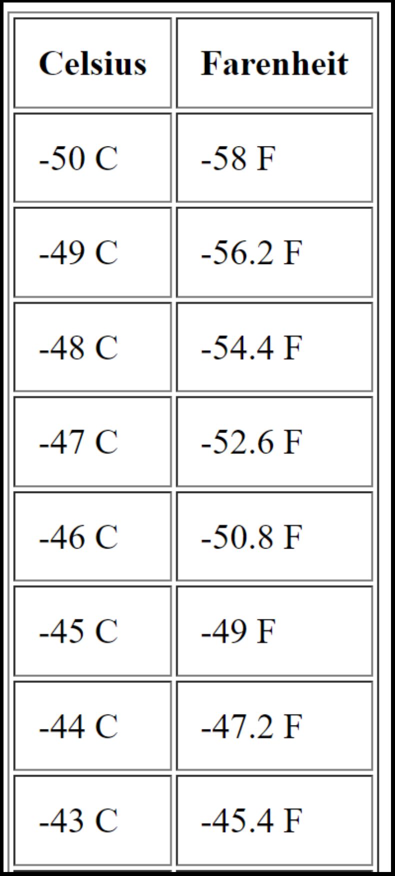
PHP's foreach loop is available in Javascript (for in) but does not work quite the same way. The foreach loop is perfectly suited for working with arrays.

```
$temp = range(-50, 50); // creates an array
foreach($temp as $c) {
    echo "<tr>";
    echo "<td>" . $c . " C</td>";
    echo "<td>" . ($c * 1.8 + 32) . " F</td>";
    echo "</tr>\n";
}
```

The foreach loop also allows to acces the key of each element inside the loop, when used like this:

```
foreach($array as $key => $value) { ... }
```

16_loops4.php



Celsius	Farenheit
-50 C	-58 F
-49 C	-56.2 F
-48 C	-54.4 F
-47 C	-52.6 F
-46 C	-50.8 F
-45 C	-49 F
-44 C	-47.2 F
-43 C	-45.4 F

Next:

Arrays and Array Functions