INLS 672
Web Development 2

PHP Introduction
Variables, operators, forms

UNC
SCHOOL OF INFORMATION
AND LIBRARY SCIENCE

Joan Boone
jpboone@email.unc.edu

First PHP Script

```
<html>
    <head><title>Hello World</title></head>
    <body>
        <h2>Greetings...</h2>
        <?php
            $greeting = "Hello World";
            echo "<p>$greeting</p>";
        ?>
    </body>
</html>
```

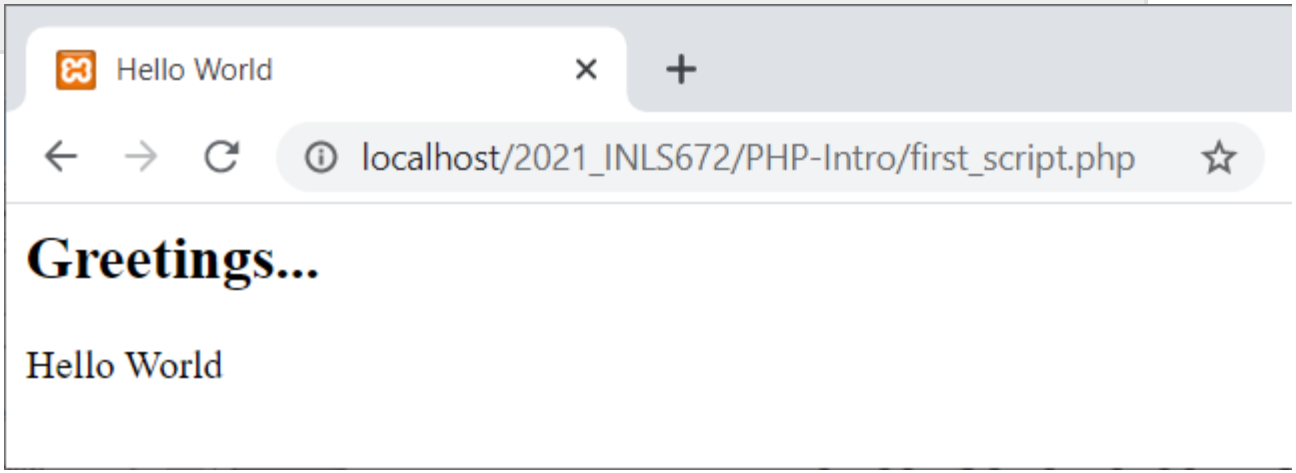
Start tag indicates PHP statements follow

PHP variable is assigned a string value, delimited by quotes

End PHP tag

Echo is a PHP function that outputs a string

Output

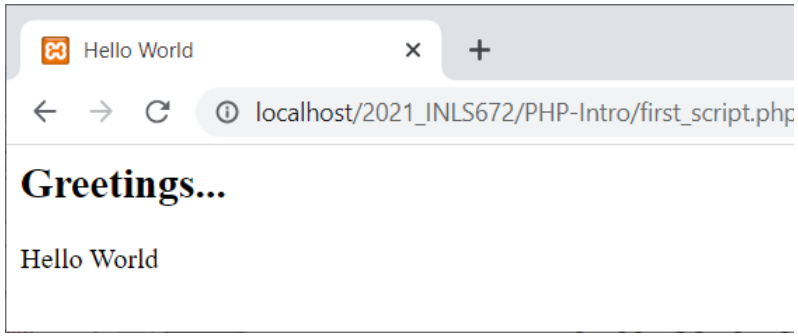


First Script Example

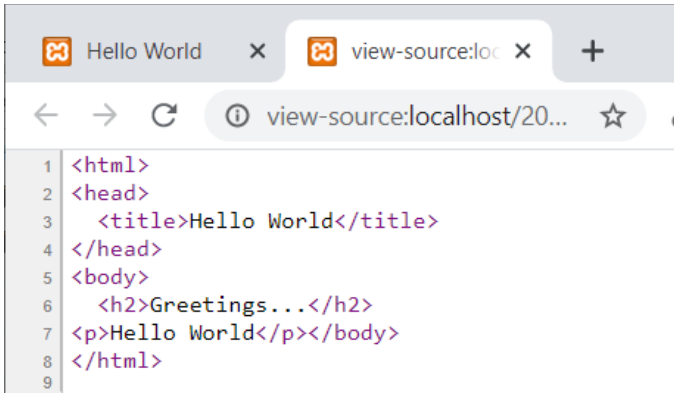
first_script.php

```
<html>
   <head>
     <title>Hello World</title>
   </head>
   <body>
      <h2>Greetings...</h2>
      <?php
         $greeting = "Hello World";
         echo "<p>$greeting</p>";
      ?>
   </body>
</html>
```

What the browser renders...



"View page source"



How this works

- Browser issues a GET request (to the Apache server) for **`first_script.php`**
- Because the file has a PHP extension, Apache gives the file to the PHP interpreter for processing
- PHP creates an output file that contains the HTML and the output from executing the PHP script.
- In this example, the script outputs a paragraph element containing the value of the PHP variable, **`$greeting`**

PHP Variables

Different types

- Number: integers, floating point numbers (25, 25.0)
- String (“Hello World”)
- Boolean (true, false)
- Array (array('Hello', 'World'))

Loosely (or weakly) typed (similar to Python, JavaScript)

- This means you do not have to specify a variable's type
- PHP converts to the type required by context

Variable names start with **$**

- Followed by a letter or underscore
- Contain only a-z, A-Z, 0-9, and underscore
- No spaces
- Case sensitive: `$Password` is not the same as `$password`

Operators

Arithmetic

Addition, subtraction, multiplication, division

Comparison

> < >= <= == !=

Logical

and (&&), or (||), ! (not)

String

Concatenation (.)

w3schools: PHP Operators

Purchase Order Example

simple_purchase_order.php

```
<html>
<head>
<style>...</style>
</head>
<body>
<?php
$quantity = 200;
$price = 35.50;
$total = $quantity * $price;
?>
<table>
  <thead>
    <tr>
      <td>Quantity</td>
      <td>Price</td>
      <td>Total</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td><?php echo $quantity ?></td>
      <td><?php echo $price ?></td>
      <td><?php echo $total ?></td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

simple_purchase_order.php

What the browser renders...

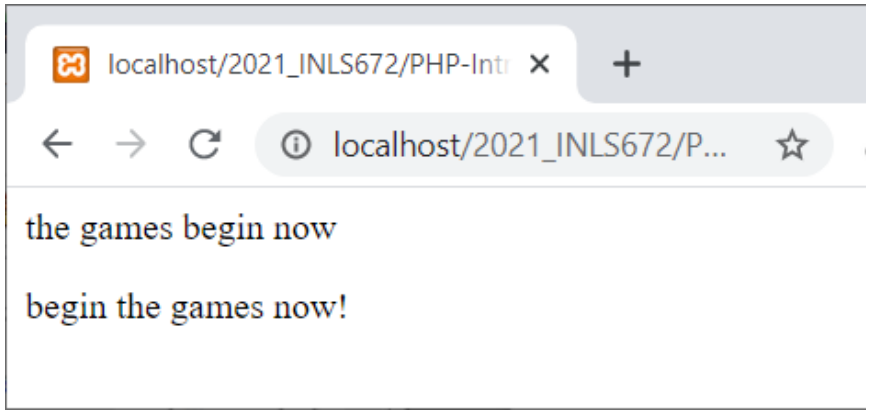
"View page source"

```
<html>
<head>
<style>...</style>
</head>
<body>
<table>
  <thead>
    <tr>
      <td>Quantity</td>
      <td>Price</td>
      <td>Total</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>200</td>
      <td>35.5</td>
      <td>7100</td>
    </tr>
  </tbody>
</table>
</body>
</html>
```

String Concatenation Example

Unlike most programming languages, PHP uses '.' for concatenation

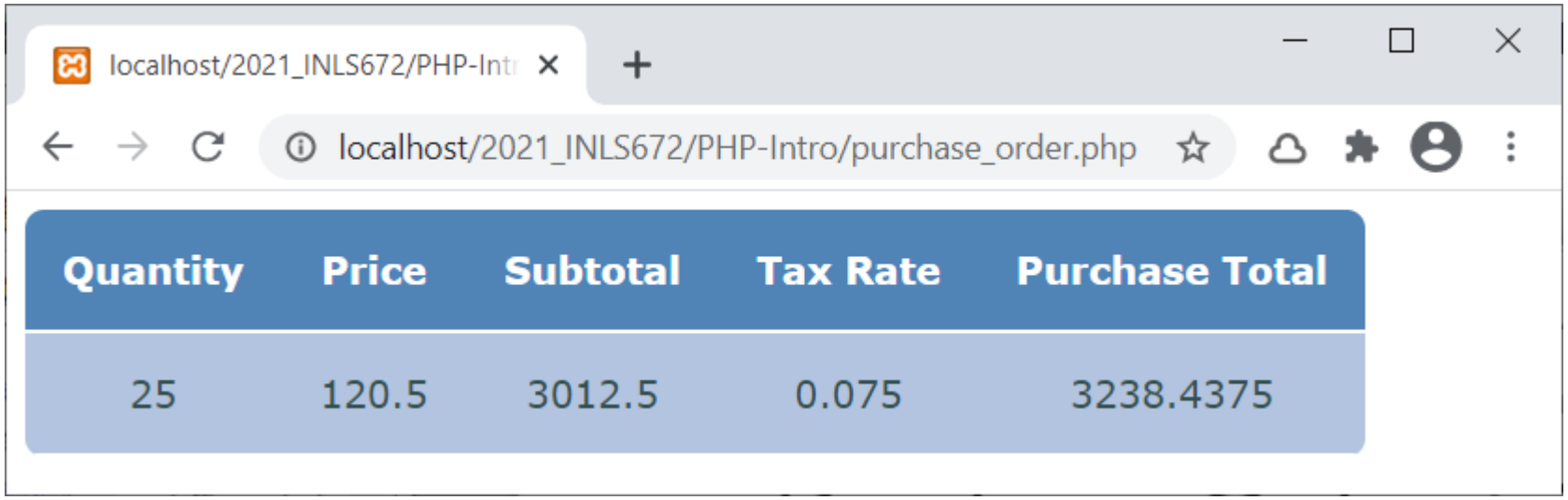
```
<html>
<head>
</head>
<body>
<?php
// initialize string variables
$a = 'the';
$b = 'games';
$c = 'begin';
$d = 'now';

// combine them using the concatenation operator
$statement = $a.' '.$b.' '.$c.' '.$d.'<br>';
$command   = $c.' '.$a.' '.$b.' '.$d.'!';
?>
<p><?php echo $statement ?></p>
<p><?php echo $command ?></p>
</body>
</html>
```



string_concatenation.php

Purchase Order + Taxes

- Calculates subtotal from quantity and price
- Applies tax rate
- Calculates purchase total



Quantity	Price	Subtotal	Tax Rate	Purchase Total
25	120.5	3012.5	0.075	3238.4375

purchase_order.php

Practice: Purchase Order

- Include Tax rate in the heading
- Replace tax rate with the tax amount
- Use PHP function, `number_format`, for decimal formatting

localhost/2021_INLS672/PHP-Intr

localhost/2021_INLS672/PHP-Intro/purcha...

Quantity	Price	Subtotal	Tax (7.5%)	Purchase Total
25	120.50	3012.50	225.94	3238.44

Original version:

localhost/2021_INLS672/PHP-Intro/purchase_order.php

Quantity	Price	Subtotal	Tax Rate	Purchase Total
25	120.5	3012.5	0.075	3238.4375

Forms

Why forms?

- Many PHP applications work with form data that was received from an HTTP request
- PHP takes the user input from a web form and converts it to PHP variables

How to build an HTML form

- Use the opening `<form>` and closing `</form>` tags
- Specify a submission type, either a GET or POST method
- Include one or more `input` fields
- Specify the destination URL to which the form is submitted

References

- w3schools: PHP Form Handling
- w3schools: HTML Forms
- w3schools: HTML Input Types and Input Attributes

How Client-side form data is sent to a Server-side PHP script

- When a **GET** or **POST** method is used, the form input values are passed as parameters with the HTTP request

```
<form action="Message.php" method="post">
```

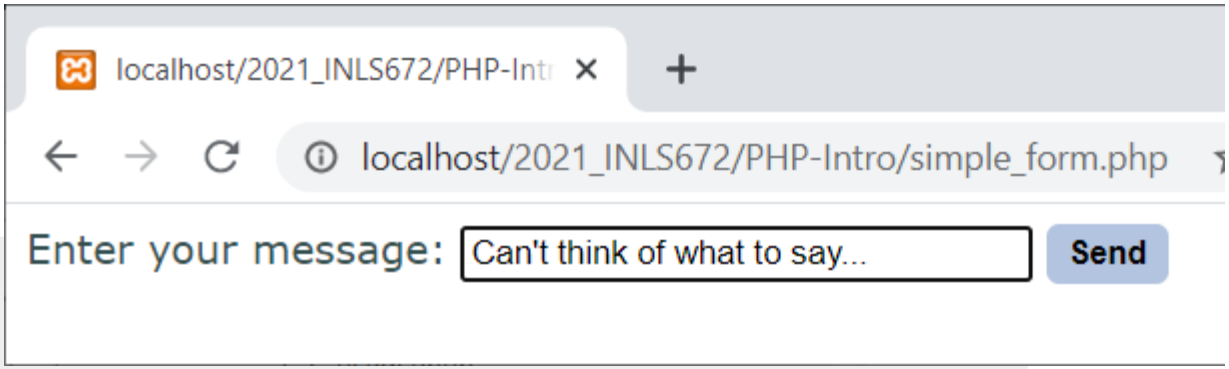
- PHP has <u>built-in arrays</u> that contain the values passed: **$_GET** and **$_POST** arrays that store key-value pairs:

Key	Value
first_name	John
last_name	Smith

When to use GET vs. POST

- Generally, use GET when reading data from a database
- Use POST when writing data to the database

Simple Form Example

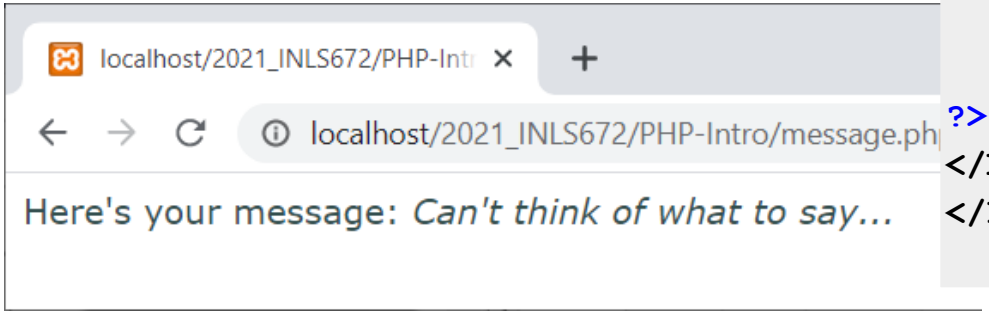



```
<html>
<head></head>
<body>
   <form action="message.php" method="post">
      Enter your message: <input type="text" name="msg" size="30">
      <input type="submit" name="submit" value="Send">
   </form>
</body>
</html>
```

simple_form.php

When the Send button is clicked...

- The action associated with the form is processed: **message.php**
- **message.php** gets the value entered in the **msg** text field from the _POST array and writes (echos) to HTML page

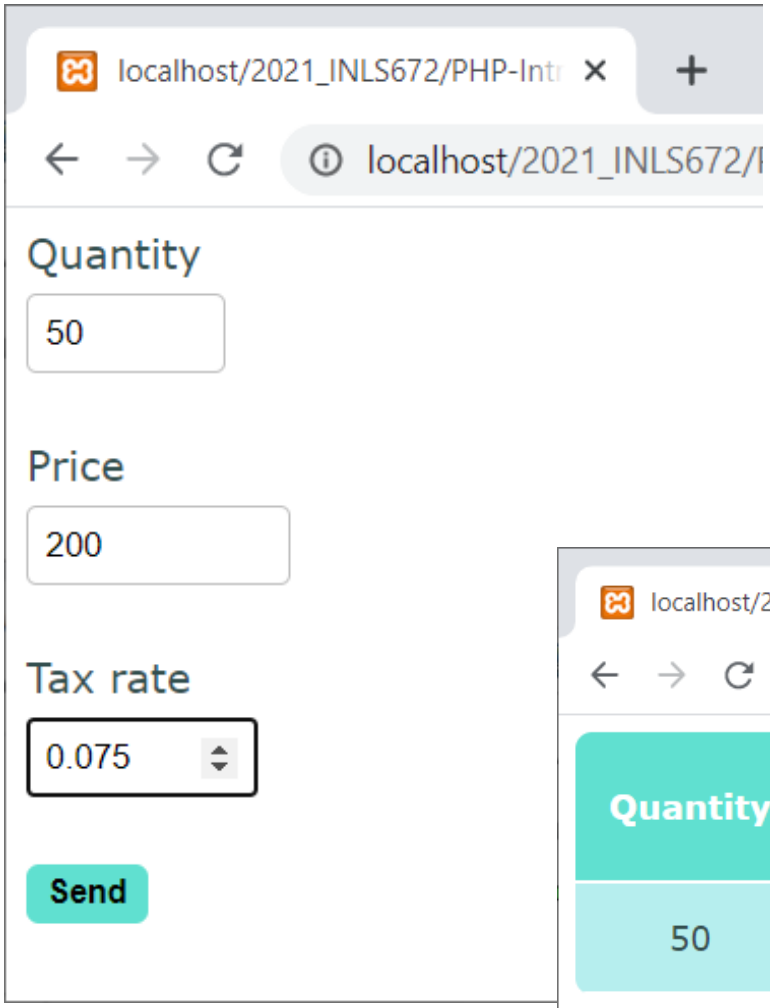


message.php

```
<html>
<head></head>
<body>
<?php
   // retrieve form data
   $input = $_POST['msg'];
   echo "Here's your message: <i>$input</i>";
?>
</body>
</html>
```

Purchase Order with Form Input

purchase_order_form.php
(provided in Examples)



- Create p**urchase_order_message.php** to process the form inputs
- Very similar to the last version of purchase_order.php, but the quantity, price, and tax rate are not hard-coded. The values are obtained from the $POST array.

purchase_order_message.php