Learning PHP

A GENTLE INTRODUCTION TO THE WEB'S MOST POPULAR LANGUAGE

Intro to PHP (Hypertext Preprocessor)

- Static Websites: the content does not change and is fixed. The content is the same for all visitors. **E.g. personal websites**
- Dynamic Websites: pictures and contents are different for different visitors.**E.g. Amazon.com**

- PHP is a programming language for building **dynamic websites**
- PHP is a **server-side** language
  - Example: JavaScript is a **client-side** language
  - Example: PHP and ASP.NET are **server-side** languages
- PHP is **free** and **general purpose** language
- **OS X** and most **Linux** distributions come with PHP **already installed**.

Static Webpages

- PHP **runs on the server** not on the client

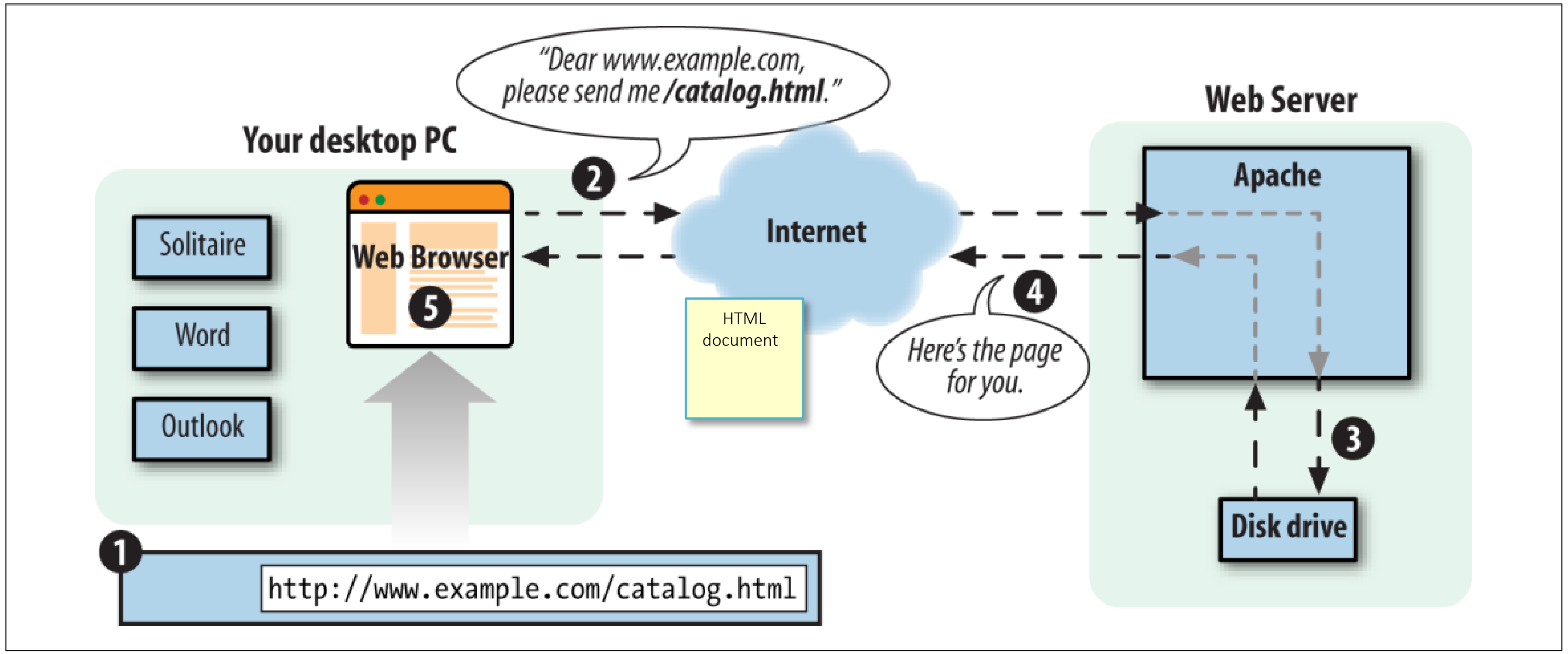


Figure 1-1. Client and server communication without PHP

Dynamic Webpages

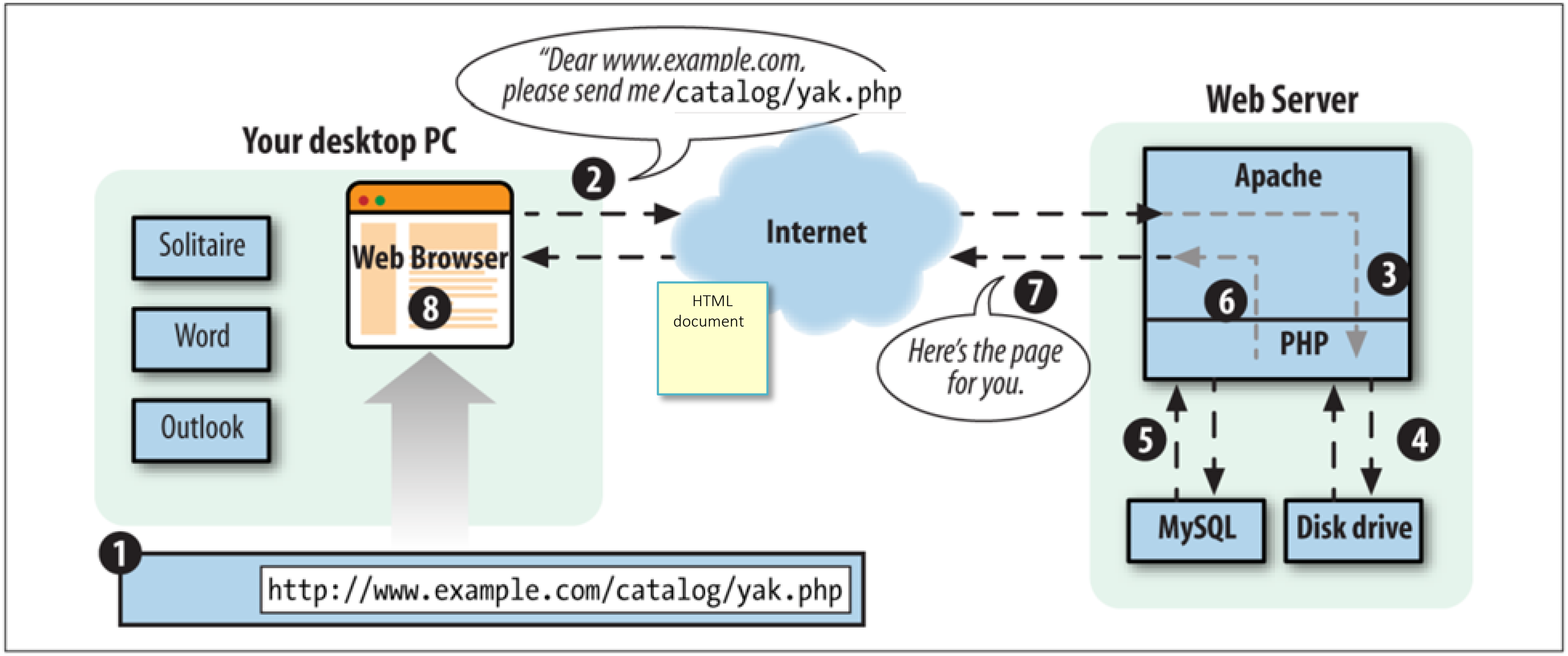


Figure 1-2. Client and server communication with PHP

PHP and PHP Engine

- PHP is a language
- PHP Engine is a software
  - It runs on a Web Server
  - It understands PHP language and executes the commands
  - For example, talking to DBMS, retrieves data and generates pages
- PHP works with a web servers running on **Windows**, **Mac OS X**, **Linux**, and many **other versions of Unix**.
- PHP works on **Web Servers** such as **Apache**, **nginx**, **MS IIS**, or any web server that supports CGI standard.
- PHP works on many **DBMSs**: **MySQL**, **PostgreSQL**, **Orcle**, **MS SQL Server**, **SQLite**, **Redis**, and **MongoDB**
- PHP is used on more than **200 million different websites**, including giants like **Facebook**, **Wikipedia**, and **Yahoo.**

Basics of PHP - <?php ?>

- It can be part of a HTML file
- It starts with <?php and ends with ?>
- **PHP engine** executes only code between <?php **and** ?>, text out of them is ignored
  - It can be <?php or <?PHP but no space in <?php
- If there is no code at the end of the file, ?> end tag is optional
- There can be multiple blocks of HP code in an HTML file
- NOTE: The extension of file must be .php not .html
- PHP can go anywhere, including inside HTML tag

attributes, and inside quotes ("):

```
<div class="<?php echo 'big-element'; ?>">Hello</div>
```

```
<span>Five plus five is:<span>
<?php echo 5 + 5; ?>
<p>
Four plus four is:
<?php
echo 4 + 4;
?>
</p>
<img src="vacation.jpg" alt="My
Vacation" />
```

Basics of PHP – Whitespaces

- Line breaks do not effect the output
- In the example on right, echos are on two lines but the

Output is on one line!

```
<p>before PHP</p>
    <?php

    echo '<p>I am PHP.</p>';

    echo '<p>Still PHP.</p>';

    ?>
    <p>Now HTML.</p>
```

```
<p>before PHP</p>
<p>I am PHP.</p><p>Still PHP.</p>        <p>Now HTML.</p>
```

Basics of PHP – Whitespaces

- Every program is composed of **statements**
- Statments end with **semi-colon** (;)
- You can put as many **blank lines between statements** as you want.

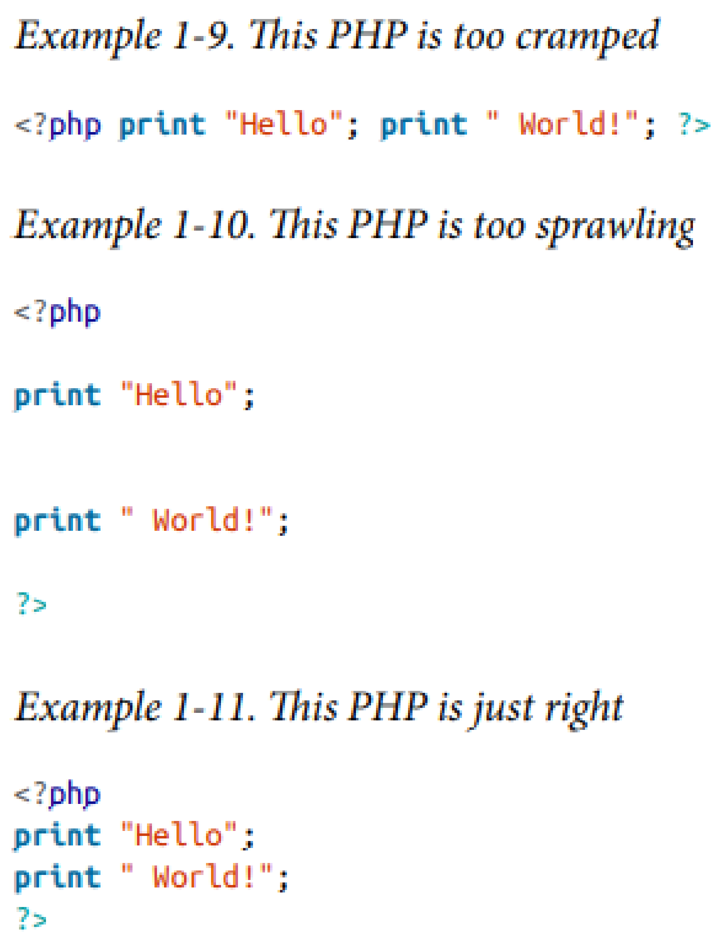
The PHP engine ignores them.

- It is recommended to put **one statement on a line**

 and blank lines between statements only when it

improves the readability.

- You can write **multiple PHP statements** on the same line of a program as long as they are separated with a **semicolon**.



Example 1-9. This PHP is too cramped

```
<?php print "Hello"; print " World!"; ?>
```

Example 1-10. This PHP is too sprawling

```
<?php

print "Hello";


print " World!";

?>
```

Example 1-11. This PHP is just right

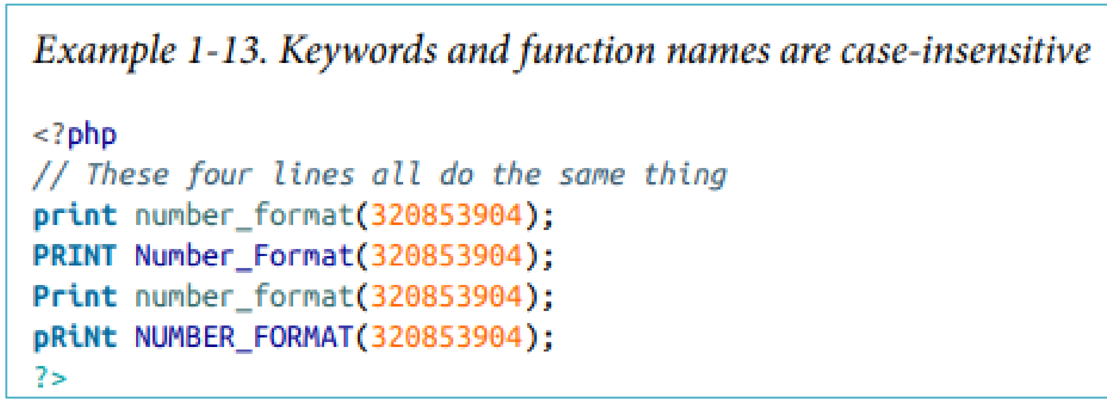
```
<?php
print "Hello";
print " World!";
?>
```

Basics of PHP – Case-sensivity

- PHP is a mixed case-sensitive and insensitive language
  - function names, class names, keywords are insensitive
    - *print, Print, PrInT are the same. or echo, Echo, ECHO, eCHo are the same*
  - Variables are case-sensitive
    - *$_POST and $_post are different*
    - *$name_of_car and $NAME_OF_CAR are different*

```
<?php
    $name_of_car = "Audi R8";

    echo $NAME_OF_CAR;
?>
```



Example 1-13. Keywords and function names are case-insensitive

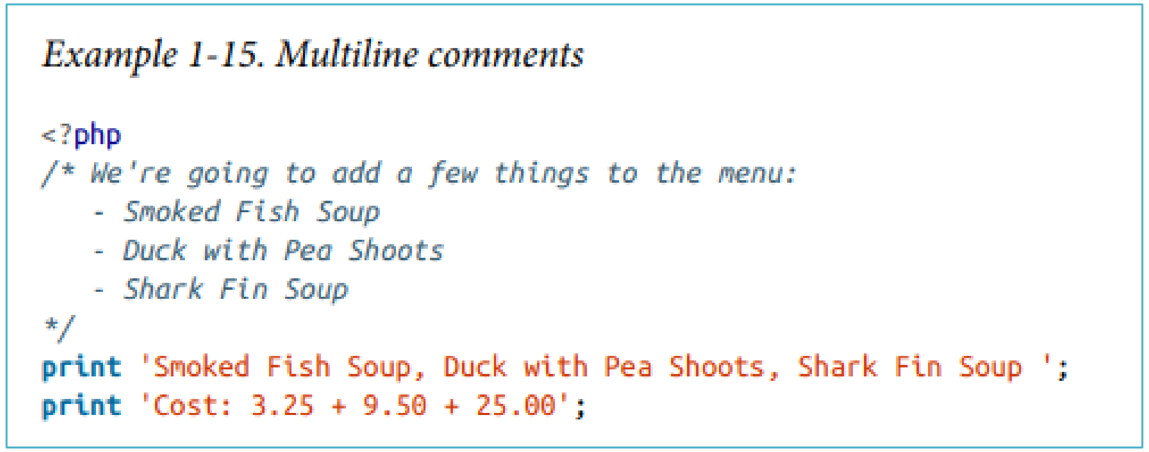
```
<?php
// These four lines all do the same thing
print number_format(320853904);
PRINT Number_Format(320853904);
Print number_format(320853904);
pRiNt NUMBER_FORMAT(320853904);
?>
```

Basics of PHP - Comments

Comments

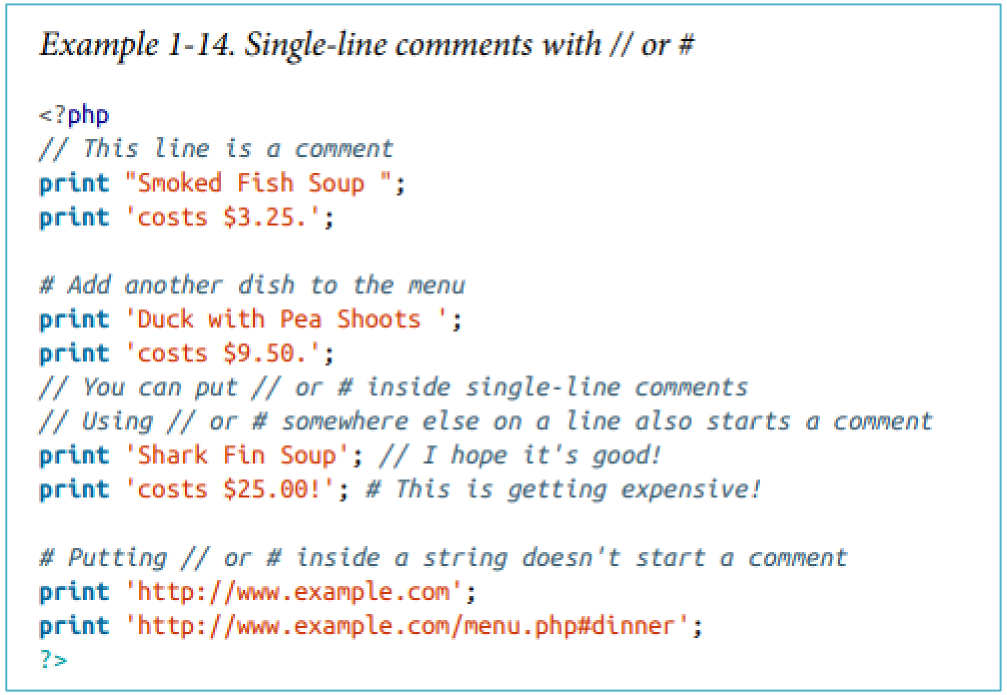
1. are an essential part of any program. By **explaining in plain language** how the programs work, comments make programs much more understandable.
2. You can also disable a part of code for testing your program

- inline comments //
- Inline comments #
- multiline commensts /* */



Example 1-15. Multiline comments

```php
<?php
/* We're going to add a few things to the menu:
    - Smoked Fish Soup
    - Duck with Pea Shoots
    - Shark Fin Soup
*/
print 'Smoked Fish Soup, Duck with Pea Shoots, Shark Fin Soup ';
print 'Cost: 3.25 + 9.50 + 25.00';
```



Example 1-14. Single-line comments with // or #

```php
<?php
// This line is a comment
print "Smoked Fish Soup ";
print 'costs $3.25.';

# Add another dish to the menu
print 'Duck with Pea Shoots ';
print 'costs $9.50.';
// You can put // or # inside single-line comments
// Using // or # somewhere else on a line also starts a comment
print 'Shark Fin Soup'; // I hope it's good!
print 'costs $25.00!'; # This is getting expensive!

# Putting // or # inside a string doesn't start a comment
print 'http://www.example.com';
print 'http://www.example.com/menu.php#dinner';
?>
```

echo , print Statements

- In PHP, there are two ways for printing anything like string, variable etc.
  - echo
  - print
- Important Point About print
  - Print statement delivers only one string or variable to the output.
  - You cannot pass the multiple arguments in case of the print.
  - Print always return one.
  - It's slower than an echo.
- **Brackets** are **optional, space** after print is **optional**
- **Multiple arguments is not allowed for print**

```
print "Hello PHP!", "How are you?";    ERROR
```

```
print "Hello PHP!";

print"<p>Welcome to the PHP</p>";

print("Hello PHP!");

print ("<p>Welcome to the PHP</p>");
```

```
# print returns 1, if it is successful
    $x = print("Print a string<br/>");

    print "return value is " . $x;
```

echo , print Statements

- With echo you can pass multiple arguments to output.
- It's faster than print statement.
- **Brackets** are optional, **space** after echo is optional
- Multiple arguments is allowed for echo

```
echo "Hello PHP!";
echo ("Hello PHP!");

echo "<p>Welcome to the PHP</p>";

echo "Hello!", "<p>Welcome to PHP!</p>";
```

- With brackets, multiple arguments is NOT allowed

```
echo ("Hello PHP!", "How are you?");    ERROR
```

Short echo

- PHP includes a short echo tag <?= which is a short-hand of <?php echo.

```
<?php echo "Hello PHP!"; ?>

// is equal to

<?= "Hello PHP!"; ?>
```

First Program: HELLO WORLD!

The .PHP file

```
<!DOCTYPE html>
<html>
<head>
    <title>PHP says hello</title>
</head>
<body>
    <b>
        <?php
        print "Hello, World!";
        ?>
    </b>
</body>
</html>
```

OUTPUT:

```
<!DOCTYPE html>
<html>
<head>
    <title>PHP says hello</title>
</head>
<body>
      <b>
           Hello, World!
      </b>
</body>
</html>
```

Strings – parameters of echo and print

- A string is series of characters
- You can use single-quoted or double-quoted strings for echo and print
- There are differences between **single-quoted** or **double-quoted** will discuss later

```
echo 'Hello PHP!';

echo "Hello PHP!";
```

- We can concatenate strings by a dot (.)
  - **Note** that the '+' (addition) operator will not work for this

```
echo "Hello PHP!" . "How are you?";

# or

echo 'Hello PHP!' . 'How are you?';
```

Escape Characters in strings

- Some of common escape characters

\n	new line
\r	carriage return
\t	horizontal tab
\\	backslash
\$	dollar sign
\"	double-quote
\'	single-quote

- \n, \r, variables do not work with single-quote strings
- You must use double-quote strings

```
echo 'You can also have embedded newlines in
strings this way as it is
okay to do';

// Output: Arnold once said: "I'll be back"
echo 'Arnold once said: "I\'ll be back"';

// Output: You deleted C:\*.*?
echo 'You deleted C:\\*.*?';

// Output: You deleted C:\*.*?
echo 'You deleted C:\*.*?';

// Output: This will not expand: \n a newline
echo 'This will not expand: \n a newline';

// Output: Variables do not $expand $either
echo 'Variables do not $expand $either';
```

Another Example

```
index.html

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Example</title>
</head>
<body>
    <form method="POST" action="sayhello.php">
        Your Name: <input type="text" name="user" />
        <br />
        <button type="submit">Say Hello</button>
    </form>
</body>
</html>
```

```
sayhello.php

<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8" />
    <title>Example</title>
</head>
<body>
<?php
echo "Hello, ";
// Print what was submitted in the form parameter called 'user'
echo $_POST['user'];
echo "!";
?>
</body>
</html>
```