Web Scripting using PHP

Server side scripting

No Scripting example - how it works...

User on a machine somewhere

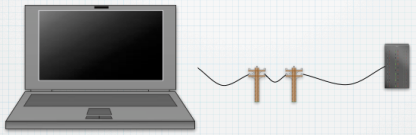
Server machine

So what is a Server Side Scripting Language?

- Programming language code embedded into a web page

PERL
PHP
PYTHON
ASP

Being more specific...

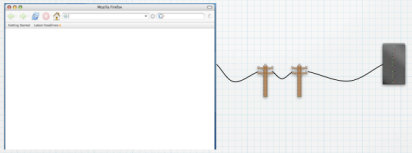


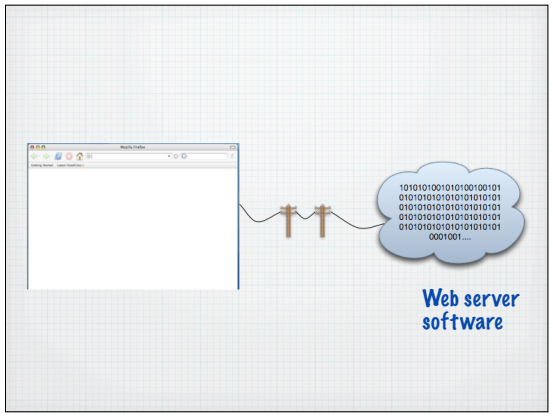
Different ways of scripting the Web

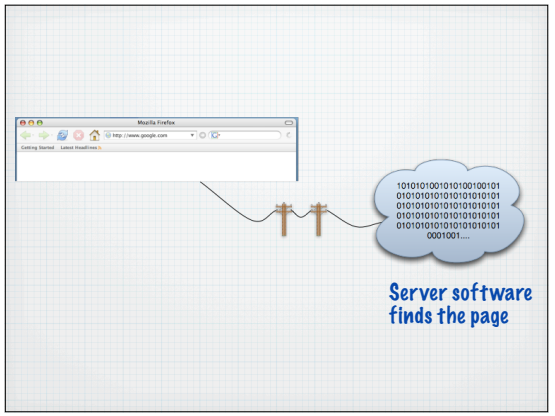
- Programming language code embedded into a web page

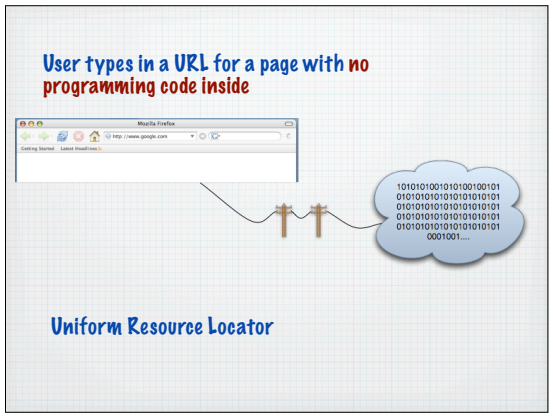
No scripting (plain markup)
Client Side scripting
Server Side scripting
Combination of the above (AJAX)

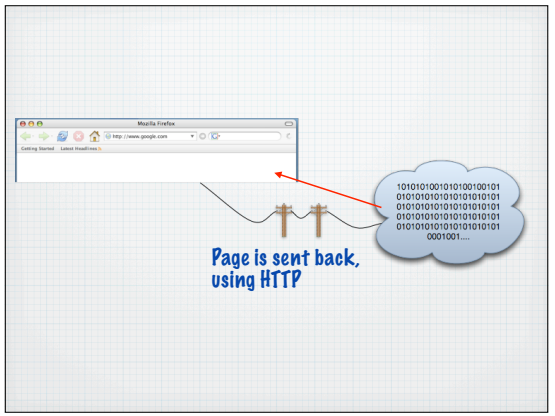
Web Browser software

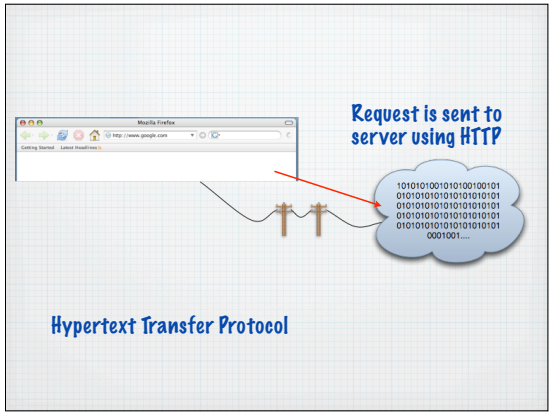


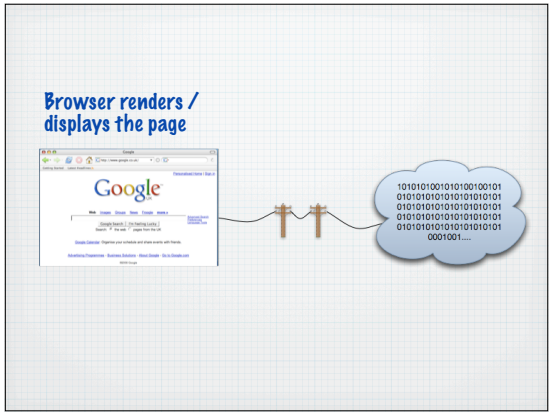

Web server
software


Server software
finds the page

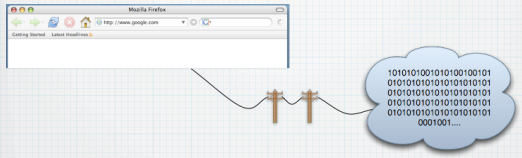

User types in a URL for a page with no
programming code inside
Uniform Resource Locator

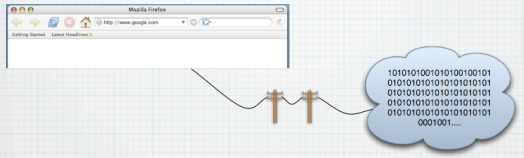

Page is sent back,
using HTTP


Request is sent to
server using HTTP
Hypertext Transfer Protocol


Browser renders /
displays the page

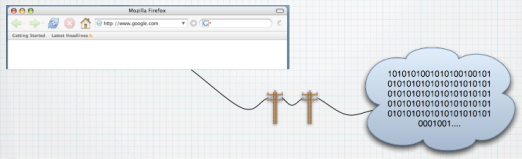
Server Side scripting

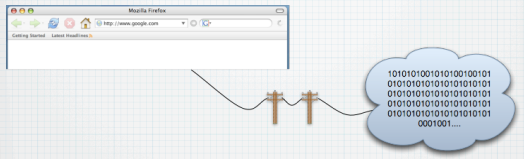




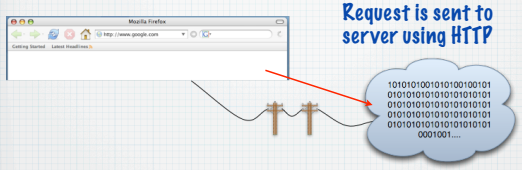
Server software finds the page

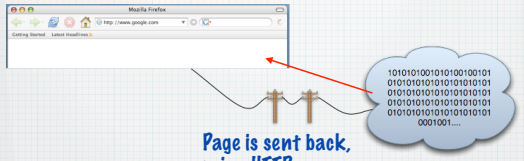
User types in a URL for a page with PHP code inside



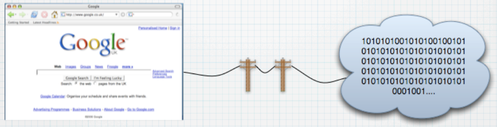


Server side code is executed



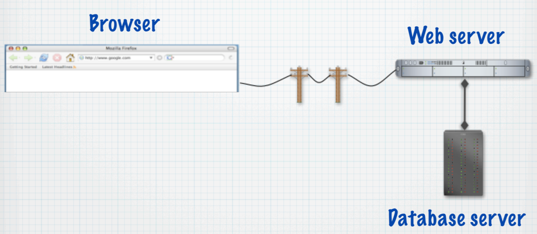


Browser renders / displays the page

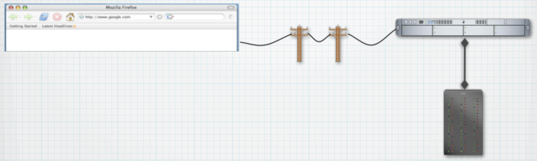


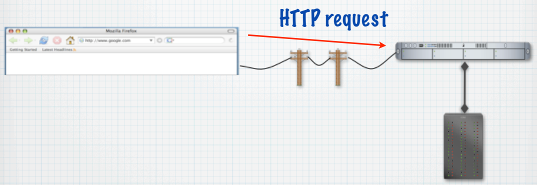
Server side scripting languages

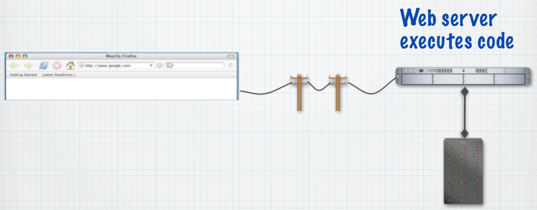
- Executes in the server
- **Before** the page is sent from server to browser
- Server side code is **not** visible in the client
- Server side code can access resources on the server side

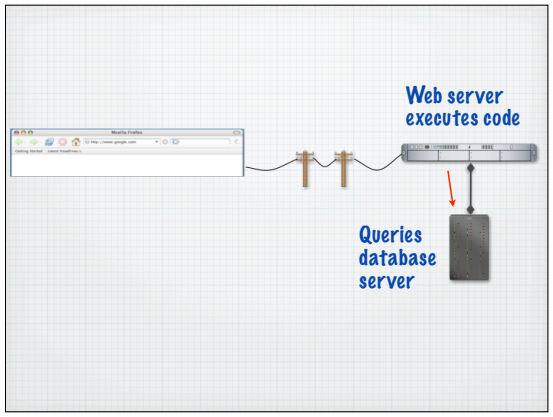


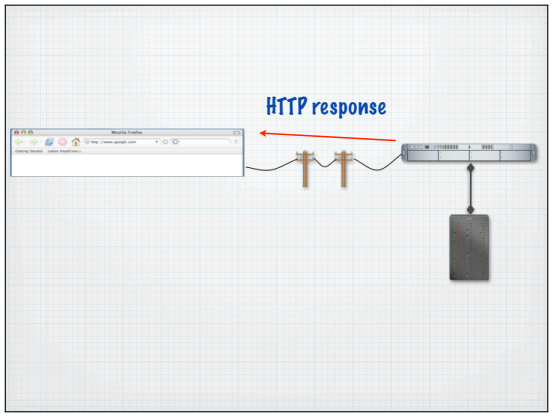
How many items in stock?

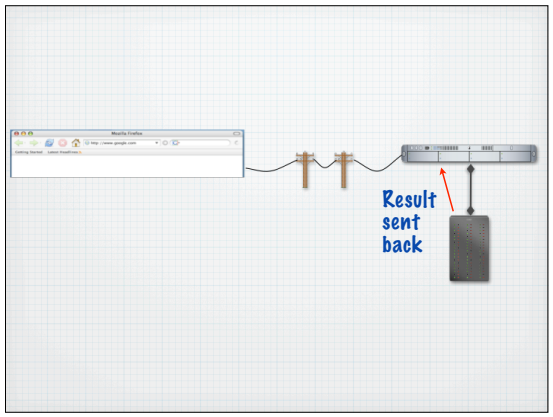


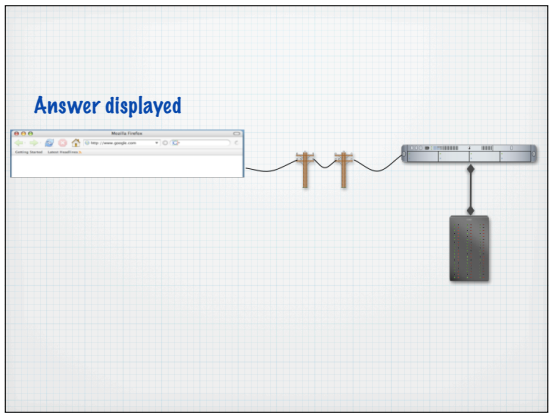


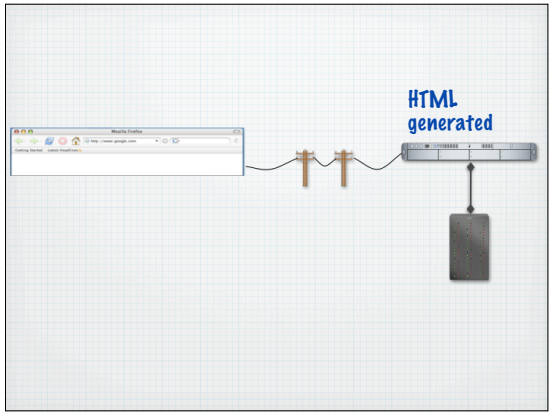








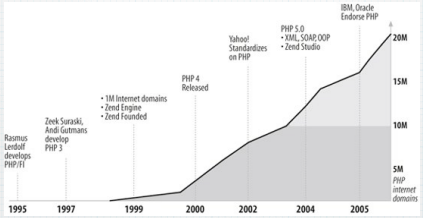




So why PHP?

PERL
PHP
PYTHON
ASP

PHP usage ...



- Source: PHP programming 2nd Ed.

PHP compared to others ...

Module	December 2006 Count	December 2006 %	November 2006 Count	November 2006 %	Growth %
PHP	6,331,188	40.11	6,364,279	40.68	-1.40
mod_ssl	4,423,054	28.02	4,418,404	28.24	-0.78
OpenSSL	4,422,425	28.02	4,417,699	28.24	-0.78
FrontPage	2,884,261	18.27	2,913,876	18.62	-1.90
perl	1,481,946	9.39	1,486,968	9.50	-1.22
mod_log_bytes	1,337,230	8.47	1,342,150	8.58	-1.25
mod_auth_passthrough	1,328,871	8.42	1,334,202	8.53	-1.28
mod_bwlimited	1,327,687	8.41	1,333,757	8.52	-1.34
DAV	692,410	4.39	682,175	4.36	0.60
mod_fastcgi	623,182	3.95	617,590	3.95	0.01
mod_gzip	584,106	3.70	579,732	3.71	-0.14
mod_throttle	521,597	3.30	523,859	3.35	-1.32
mod_jk	445,871	2.82	433,601	2.77	1.92
PHP-CGI	418,725	2.65	419,243	2.68	-1.01
Python	304,372	1.93	304,352	1.95	-0.88
mod_python	304,371	1.93	304,350	1.95	-0.88
Perl	291,867	1.85	279,658	1.79	3.44

- Source: www.securityspace.com

Books - core syntax

Programming PHP, Second Edition

By Kevin Tatroe, Rasmus Lerdorf, Peter MacIntyre
Second Edition April 2006

** Recommended

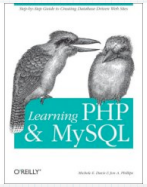
PHP in a Nutshell

By Paul Hudson
First Edition October 2005

Books - learning / tutorial based

Learning PHP 5

By David Sklar
First Edition June 2004



Learning PHP and MySQL

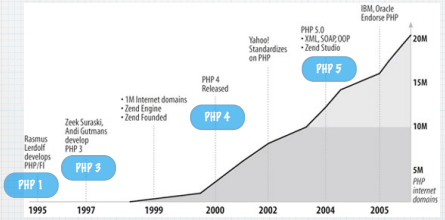
By Michele Davis, Jon Phillips
First Edition June 2006

Other texts..

- There are other publishers / texts (trade books)
- Look for books that cover PHP 5

- Open source, server side languages can rapidly develop
- Features added or deprecated rapidly

PHP development



- 5 versions in 10 years

Language basics

- Embedding PHP in Web pages
- Whitespace and Line breaks
- Statements and semicolons
- Comments
- Literals / names
- Identifiers
- Keywords
- Data types

Much of this material is explained in PHP programming 2nd Ed. Chap 1 & 2

Embedding PHP in web pages

```
<?php
statement;
statement;
statement
?>
```

Use <?php and ?> to surround the php code

Embedding PHP in web pages

```
<?php
statement;statement;   statement;
             statement;
   statement;statement;
?>
```

In general whitespace doesn't matter

Use indenting and separate lines to create readable code

The legendary Hello World program

```
<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.01
   Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>This is the first PHP program</title>
</head>
<body>
<p>
<?php
print "Hello World!";
?>
</p>
</body>
</html>
```

The legendary Hello World program

```
<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.01
   Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>This is the first PHP program</title>
</head>
<body>
<p>
<?php
print "Hello World!";
?>
</p>
</body>
</html>
```

The legendary Hello World program

```
<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML 4.01
   Transitional//EN"
 "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>This is the first PHP program</title>
</head>
<body>
<p>
<?php
print "Hello World!";
?>
</p>
</body>
</html>
```

print sends a sequence of characters to the output

The sequence here is indicated by start and end quotes

PHP can be put anywhere'..

All the php blocks are processed before the page is sent

```
<html>
<?php ... ?>
<head>
<?php ... ?>
<title>... <?php ... ?> ...</title>
</head>
<body>
<p>
<?php ... ?>
</p>
</body>
</html>
```

All of these would work the same way...

```
<?php statement; statement;statement ?>
```

```
<?php
statement; statement;statement;
?>
```

```
<?php
statement;
statement;
statement;
?>
```

This is the best way of laying the code out

PHP can be put anywhere'.. but works in sequence

```
<html>
<?php ... ?>
<head>
<?php ... ?>
<title>... <?php ... ?> ...</title>
</head>
<body>
<p>
<?php ... ?>
</p>
</body>
</html>
```

Starting at the top

Working down to the bottom

Comments

Many different ways to add comments

Comment	Source	Action
//	C++	Comments to EOL
#	Unix shell scripting	Comments to EOL
/* and */	C	Comments out a block

Statements and semicolons

```
<?php
statement;
statement;
statement;
?>
```

Use ; to separate statements

Comments

```
<?php
php statement; // A comment here
php statement; # Another comment here

/* A series of lines
with comments ignored by the PHP processor
*/
php statement;
?>
```

Comments

```
<?php
php statement; // A comment here
php statement; # Another comment here

/* A series of lines
with comments ignored by the PHP processor
*/
php statement;
?>
```

Everything in red is ignored by the PHP interpreter

Language basics

- Embedding PHP in Web pages ✔
- Whitespace and Line breaks ✔
- Statements and semicolons ✔
- Comments ✔
- Literals / names
- Identifiers
- Keywords
- Data types

Literals

A data value that appears directly in the program

2001	An integer
0xFE	Hexadecimal number
1.4142	Float
"Hello World"	String
'Hi'	String
true	Bool
null	built in 'no value' symbol

Identifiers

Identifiers (or names) in PHP must -

Begin with an ASCII letter (uppercase or lowercase)

or begin with the underscore character _

or any character between ASCII 0x7F to 0xFF

followed by any of these characters and the digits 0-9

Variables

Variables in PHP are identifiers prefixed by $

```
$bill
$value_count
$anothervalue3
$THIS_IS_NOT_A_GOOD_IDEA
$_underscore
```

Valid

Invalid

```
$not  valid
$[
$3wa
```

Case sensitivity - names we define are case sensitive

```
$value
$VALUE
$vaLUE
```

Three different names

Variables

We use variables for items of data that will change as the program runs

Choose a sensible name and have as many as you like

$total_income

$bill

$salary $month

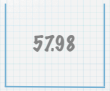
$total

$percentage_increase

Variables

To give it a value, use the equals sign

$bill = 57.98



Variables

When we declare a variable, a space is reserved and labelled for that item (in memory)

$bill

Variables

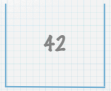
To give it a value, use the equals sign

$bill = "No payment"

Variables

To give it a value, use the equals sign

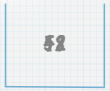
$bill = 42



Variables

If a value is changed, the old value is overwritten

$bill = 42;
$bill = 58;



Variables

Sometimes we use the old value to recalculate the new value

```
$bill = 42;
$bill = $bill*2 ;
```

Variables

Some languages are very strict about what kinds of data are stored in variables - PHP doesn't care

```
$bill=42;
```

Stores an integer

```
$bill=42;
$bill="Now its a string";
```

Overwrites with a string

```
print $bull;
```

Whoops - made a mistake but it still works

Variables

Some languages are very strict about what kinds of data are stored in variables - PHP doesn't care

```
$bill=42;
```

Stores an integer

```
$bill=42;
$bill="Now its a string";
```

Overwrites with a string

```
print $bull;
```

Whoops - made a mistake but it still works

Keywords

Reserved by the language for core functionality

Also - can't use a built in function name as a variable

__CLASS__	Declare	extends	print( )
__FILE__	Default	final	private
__FUNCTION__	die( )	for	protected
__LINE__	Do	foreach	public
__METHOD__	echo( )	function	require( )
Abstract	Else	global	require_once( )
And	elseif	if	return( )
array( )	empty( )	implements	static
As	enddeclare	include( )	switch
Break	endfor	include_once( )	tHRow
Case	endforeach	interface	TRy
catch	endif	isset( )	unset( )
cfunction	endswitch	list( )	use
Class	endwhile	new	var
clone	eval( )	old_function	while
Const	exception	Or	xor
Continue	exit( )	php_user_filter	

Data types

PHP provides 8 types

scalar (single-value)	compound
integers	arrays
floating-point	objects
string	
booleans	

Two are special - resource and NULL

Integers

Whole numbers - range depends on the C compiler that PHP was made in (compiled in)

Typically +2,147,483,647 to -2,147,483,647

Octal 0755

Hexadecimal 0xFF

Larger integers get converted to floats automatically

Floating-Point Numbers

Real numbers - again range is implementation specific

Typically	1.7E-308 to 1.7E+308 with 15 digits of accuracy
Examples	3.14, 0.017, -7.1, 0.314E1, 17.0E-3

Strings

Delimited by either single or double quotes

```
'here is a string'
"here is another string"
```

Strings - single quotes

You can use single quotes to enclose double quotes

```
$outputstring='He then said "Goodbye" and left';
```

Useful for easily printing HTML attributes

```
$outputstring='<a href="http:/www.bbc.co.uk">BBC</a>';
```

Strings - double quotes

You can use double quotes to enclose single quotes

```
$outputstring="He then said 'Goodbye' and left";
```

Operators

Standard arithmetic operators: +, -, *, /, % ..

Concatenation operator: .

```
$outputstring="He then said ".$quote;
```

Any non-string value is converted to a string before the concatenation.

Operators

```
$aBool=true;
$anInt=156;
$aFloat=12.56;
$anotherFloat=12.2E6;
$massiveFloat=12.2E-78;
print "The bool printed looks like this: ".$aBool."<br />";
print "The int printed looks like this: ".$anInt."<br />";
print "The (smaller) float printed looks like this: ".$aFloat."<br />";
print "The larger float printed looks like this: ".$anotherFloat."<br />";
print "The even larger float printed looks like this: ".$massiveFloat."<br />";
```

Operators

```
$aBool=true;
$anInt=156;
$aFloat=12.56;
$anotherFloat=12.2E6;
$massiveFloat=12.2E-78;
print "The bool printed looks like this: ".$aBool."<br />";
print "The int printed looks like this: ".$anInt."<br />";
print "The (smaller) float printed looks like this: ".$aFloat."<br />";
print "The larger float printed looks like this: ".$anotherFloat."<br />";
print "The even larger float printed looks like this: ".$massiveFloat."<br />";
```