PHP Basics

- Syntax
- Operators
- Variables
- Constants
- Control Structures
- Language Constructs and Functions
- Namespaces
- Extensions
- Config
- Performance/bytecode caching

PHP Tags

- Standard Tags - best solution for portability and backwards compatibility, because they are guaranteed to be available and cannot be disabled by changing PHP's configuration file:
    - <?php
    - ... code
    - ?>
- Short Tags - can be disabled (often for XML compliance) with short_open_tag directive in php.ini:
    - <?
    - ... code
    - ?>
    - <?=$variable ?>
- Script Tags:
    - <script language="php">
    - ... code
    - </script>
- ASP Tags - must be enabled specifically in the php.ini file via the asp_tags directive:
    - <%
    - ... code
    - %>
    - <%=$variable; %>

Comments

- Single line comments are ended by either the newline character or a closing PHP tag. Latter can cause unintended output.
    - // This is a single line comment

    - # This is also a single line comment
- /*
- This is a multi-line
- comment
- */
- /**
- * API Documentation Example
- *
- * @param string $bar
- */
- function foo($bar) { }

Operators

- >> (Shift Right) – All of the bits in the binary number shift N places to the right in the number, the right most digit(s) falls out, and the resulting number is padded on the left with 0s. A right shift of one is equivalent to dividing a number by two, and tossing the remainder.
- << (Shift Left) – All of the digits shift N places to the left, padding the right with 0s. A left shift of one is equivalent to multiplying a number by two.
- `` (backticks) – Execute the contents as a shell command (shortcut for shell_exec()) with the same permissions as the webserver.
- Instanceof – Returns true if the indicated variable is an instance of the designated class, one of it’s subclasses or a designated interface.

Variables

Naming:

- Can only contain letters, numbers or underscores
- Must start with a letter or an underscore

Types

- Scalar types: boolean, string, integer, float
- Compound types: array, object
- Special Types: resource, null

Strings

- Each character is represented by a single byte. PHP has no native support for multi-byte character sets (like Unicode)
- Ordered collections of binary data
- Can be quoted in one of three ways:
    - 'some text' - characters within single quotes will be recorded as is, without variable or escape sequences interpreted and replaced
    - "some text" – variables and escape sequences will be interpreted and replaced

    - <<< – Heredoc functions in a similar manner to double quotes, but is designed to span multiple lines, and allow for the use of quotes without escaping.
    - $greeting = <<<GREETING
    - She said "That is $name's" dog!
    - While running towards the thief
    - GREETING;

Integer

- can be specified in decimal (base 10), hexadecimal (base 16, precede with a 0x), or octal (base 8, precede with a 0) notation and optionally preceded by a sign (+, -)
- The maximum size of an integer is platform dependent, a maximum of ~2Billion is common

Float

- 1.234, 1.2e3, 7E-10
- The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value

Boolean

- Any integer other than 0 is cast to TRUE
- TRUE & FALSE are case-insensitive, though the all caps representation is common

Arrays

- Arrays can contain any combination of other variable types, even arrays or objects

Objects

- Objects allow data and methods to be combined into one cohesive structure

Resource

- special variable that represents some sort of operating system resource, generally an open file or database connection
- While variables of the type resource can be printed out, their only sensible use is with the functions designed to work with them

null

- it has no value and no type
- is not the same as the integer zero or an zero length string (which would have types)

Variable Variables

```
$a = 'name';
$$a = "Paul";
echo $name; //Paul
```

Constants

- can not be changed once set
- define('ERROR', 'Something went wrong.');
- const FOO = ‘bar’;
- only scalar

Control Structures

- If
  - Alternate (if (): ... else: ... elseif: ... endif;
  - Short ($expr ? true : false)
- Switch
- While
- do
- For
  - Continue
  - break
- Foreach
- Functions
  - Paramaters
    - Optional
    - Byref
    - Byval
    - Objects
    - Variable functions
- Objects

Language Constructs

- Elements that are built-into the language;
- Do not have return value;
- echo, die(), exit()

End a script in PHP5:

- __halt_compiler()
- die();
- exit();

Namespaces

http://www.php.net/manual/en/language.namespaces.php

Solves two problems:

1. Name collisions between code you create, and internal PHP classes/functions/constants or third-party classes/functions/constants.
2. Ability to alias (or shorten) Extra_Long_Names designed to alleviate the first problem, improving readability of source code.

Only three type of code are affected by namespaces: classes, functions and constants.
A file containing a namespace must declare the namespace at the top of the file before any other code - with one exception: the declare keyword.

Namespace name can be defined with sub-levels

A class name can be referred to in three ways:

1. Unqualified name, or an unprefixed class name like *$a = new foo();* or *foo::staticmethod();*. If the current namespace is *currentnamespace*, this resolves to *currentnamespace\foo*. If the code is global, non-namespaced code, this resolves to *foo*. One caveat: unqualified names for functions and constants will resolve to global functions and constants if the namespaced function or constant is not defined. See Using namespaces: fallback to global function/constant for details.
2. Qualified name, or a prefixed class name like *$a = new subnamespace\foo();* or *subnamespace\foo::staticmethod();*. If the current namespace is *currentnamespace*, this resolves to *currentnamespace\subnamespace\foo*. If the code is global, non-namespaced code, this resolves to *subnamespace\foo*.
3. Fully qualified name, or a prefixed name with global prefix operator like *$a = new \currentnamespace\foo();* or *\currentnamespace\foo::staticmethod();*. This always resolves to the literal name specified in the code, *currentnamespace\foo*.

Two kinds of aliasing or importing: aliasing a class name, and aliasing a namespace name

```
<?php
namespace my\name; // see "Defining Namespaces" section

class MyClass {}
function myfunction() {}
const MYCONST = 1;

$a = new MyClass;
$c = new \my\name\MyClass; // see "Global Space" section

$a = strlen('hi'); // see "Using namespaces: fallback to global
                   // function/constant" section

$d = namespace\MYCONST; // see "namespace operator and __NAMESPACE__
                        // constant" section
```

```
$d = __NAMESPACE__ . '\MYCONST';
echo constant($d); // see "Namespaces and dynamic language features" section
?>

<?php
namespace foo;
use My\Full\Classname as Another;

// this is the same as use My\Full\NSname as NSname
use My\Full\NSname;

// importing a global class
use \ArrayObject;

$obj = new namespace\Another; // instantiates object of class foo\Another
$obj = new Another; // instantiates object of class My\Full\Classname
NSname\subns\func(); // calls function My\Full\NSname\subns\func
$a = new ArrayObject(array(1)); // instantiates object of class ArrayObject
// without the "use \ArrayObject" we would instantiate an object of class foo\ArrayObject
?>
```

Extensions

http://devzone.zend.com/article/1021

The PHP source bundle comes with around 86 extensions, having an average of about 30 functions each. Do the math, that's about 2500 functions. As if this weren't enough, the PECL repository offers over 100 additional extensions, and even more can be found elsewhere on the Internet.

Config

http://www.php.net/manual/en/configure.about.php
http://php.net/manual/en/configuration.php

Performance/bytecode caching

Functions

- Syntax
- Arguments
- Variables
- References
- Returns
- Variable Scope
- Anonymous Functions, closures

Reasons

- Readability
- Maintainability
- Code separation
- Modularity
- Our Comp-Sci teacher told us to
- To do otherwise would subject us to mockery

Features

- function name() { }
- All functions in PHP return a value (null if return is not used).
- function names are not case-sensitive
- global $id; // GLOBALS['id'];
- function printList($string, $count = 5)
- function newTo5(&$number = 2)
- $a = func_get_args();
- int func_num_args(void)
- mixed func_get_arg (int $arg_num)
- PHP 5 allows default values to be specified for parameters even when they are declared as by-reference (if a variable is not passed in, a new one will be created)
- In PHP4 Objects were thrown around ByVal.
- In PHP5 Objects are always passed ByRef unless you explicitly clone it.
- Is "_" a valid function name?

By reference

- &function name() { }
    - allows you to return a variable as the result of the function, instead of a copy
    - is used for things like resources and when implementing the Factory pattern
    - you must return a variable

- you cannot return an expression by reference
- you cannot return an empty return statement to force a NULL return value

Anonymous Functions, closures

http://php.net/manual/en/functions.anonymous.php
Allow the creation of functions which have no specified name. They are most useful as the value of callback parameters, but they have many other uses.

Closures can also be used as the values of variables; PHP automatically converts such expressions into instances of the **Closure** internal class.

```
<?php
$greet = function($name)
{
   printf("Hello %s\r\n", $name);
};

$greet('World');
$greet('PHP');
?>
```

It is possible to use func_num_args(), func_get_arg(), and func_get_args() from within a closure.

The predefined final class **Closure** was introduced in PHP 5.3.0. It is used for internal implementation of anonymous functions.
The class has a constructor forbidding the manual creation of the object (issues **E_RECOVERABLE_ERROR**) and the __*invoke* method with the calling magic.

May also inherit variables from the parent scope.

```
$callback =
         function ($quantity, $product) use ($tax, &$total)
         {
            $pricePerItem = constant(__CLASS__ . "::PRICE_" .
               strtoupper($product));
            $total += ($pricePerItem * $quantity) * ($tax + 1.0);
         };

      array_walk($this->products, $callback);
```

Strings and Patterns

- Quoting
- Matching
- Extracting
- Searching
- Replacing
- Formatting
- PCRE
- HEREDOC and NOWDOC
- Encodings

Functions

- int strlen (string $string)
- int strpos (string $haystack , mixed $needle [, int $offset = 0]) - return the position of the search string within the target
- int stripos (string $haystack, string $needle [, int $offset])
- int strrpos (string $haystack, string $needle [, int $offset])
- string strstr (string $haystack , mixed $needle [, bool $before_needle = false]) - return the portion of target from the beginning of the first match until the end
- string stristr (string $haystack, string $needle)
- int strspn (string $subject , string $mask [, int $start [, int $length]]) - returns the number of characters matching the mask
- int strcspn (string $str1 , string $str2 [, int $start [, int $length]]) - uses a blacklist and returns the number of non-matching characters
- string setlocale (int $category, string $locale [, string $...])
- string str_pad (string $input , int $pad_length [, string $pad_string = " " [, int $pad_type = STR_PAD_RIGHT]])
- string strrev (string $string) - Reverse a string

Escape Sequences

- \n linefeed (LF or 0x0A (10) in ASCII)
- \r carriage return (CR or 0x0D (13) in ASCII)
- \t horizontal tab (HT or 0x09 (9) in ASCII)
- \\ backslash
- \$ dollar sign
- \" double-quote
- \[0-7]{1,3} the sequence of characters matching the regular expression is a character in octal notation

Matching Strings

- int strcmp (string $str1 , string $str2) - Returns < 0 if str1 is less than str2 ; > 0 if str1 is greater than str2 , and 0 if they are equal.
- int strcasecmp (string $str1 , string $str2) – case insensitive version of strcmp()
- int strncasecmp (string $str1 , string $str2 , int $len)
- 0 == FALSE and 0 !== FALSE

Extracting

- string substr (string $string , int $start [, int $length]) - to retrieve a portion of a string
- Array Syntax
- The {} syntax is depreciated in PHP 5.1 and will produce a warning under E_STRICT

Formatting

- sometimes governed by locale considerations
- string number_format (float $number , int $decimals , string $dec_point , string $thousands_sep) – by default formats a number with the comma as the thousands separator, and no decimal
- string money_format (string $format , float $number) – only available when the underlying c library strfmon() is available (not windows)

Print Family

- int print (string $arg) - Outputs *arg*. (language construct)
- int printf (string $format [, mixed $args [, mixed $...]]) - Output a formatted string
- string sprintf (string $format [, mixed $args [, mixed $...]]) - Return a formatted string
- int vprintf (string $format , array $args) - Display array values as a formatted string according to *format*
- mixed sscanf (string $str , string $format [, mixed &$...]) - http://ru2.php.net/sscanf
- mixed fscanf (resource $handle , string $format [, mixed &$...]) - Parses input from a file according to a format

Printf formatting specifiers:

- % - a literal percent character. No argument is required.
- b - the argument is treated as an integer, and presented as a binary number.
- c - the argument is treated as an integer, and presented as the character with that ASCII value.
- d - the argument is treated as an integer, and presented as a (signed) decimal number.
- e - the argument is treated as scientific notation (e.g. 1.2e+2).
- u - the argument is treated as an integer, and presented as an unsigned decimal number.
- f - the argument is treated as a float, and presented as a floating-point number (locale

aware).

- F - the argument is treated as a float, and presented as a floating-point number (non-locale aware). Available since PHP 4.3.10 and PHP 5.0.3.
- o - the argument is treated as an integer, and presented as an octal number.
- s - the argument is treated as and presented as a string.
- x - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
- X - the argument is treated as an integer and presented as a hexadecimal number (with
- uppercase letters).

Replacement

- mixed str_replace (mixed $search , mixed $replace , mixed $subject [, int &$count])
- mixed str_ireplace (mixed $search, mixed $replace, mixed $subject [, int &$count])
- mixed substr_replace (mixed $string, string $replacement, int $start [, int $length])
- string strtr (string $str, string $from, string $to)

PCRE(Perl Compatible Regular Expressions) – Meta Characters

- \d Digits 0-9
- \D Anything not a digit
- \w Any alphanumeric character or an underscore (_)
- \W Anything not an alphanumeric character or an underscore
- \s Any whitespace (spaces, tabs, newlines)
- \S Any non-whitespace character
- . Any character except for a newline
- ? Occurs 0 or 1 time
- * Occurs 0 or more times
- + Occurs 1 or more times
- {n} Occurs exactly n times
- {,n} Occurs at most n times
- {m,} Occurs m or more times
- {m,n} Occurs between m and n times

PCRE – Pattern Modifiers

- i – Case insensitive search
- m – Multiline, $ and ^ will match at newlines
- s – Makes the dot metacharacter match newlines
- x – Allows for commenting
- U – Makes the engine un-greedy
- u – Turns on UTF8 support
- e – Matched with preg_replace() allows you to call

- int preg_match (string $pattern , string $subject [, array &$matches [, int $flags [, int $offset]]]) – Returns the number of matches found by a given search string

- int preg_match_all (string $pattern, string $subject, array &$matches [, int $flags [, int $offset]])
- mixed preg_replace (mixed $pattern , mixed $replacement , mixed $subject [, int $limit = -1 [, int &$count]])
- array preg_split (string $pattern , string $subject [, int $limit = -1 [, int $flags = 0]])

HEREDOC and NOWDOC

http://www.php.net/manual/en/language.types.string.php#language.types.string.syntax.heredoc
As of PHP 5.3.0, it's possible to initialize static variables and class properties/constants using the Heredoc syntax.
PHP 5.3.0 also introduces the possibility for Heredocs to use double quotes in declarings.
Heredocs can not be used for initializing class properties. Since PHP 5.3, this limitation is valid only for heredocs containing variables.

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
   var $foo;
   var $bar;

   function foo()
   {
      $this->foo = 'Foo';
      $this->bar = array('Bar1', 'Bar2', 'Bar3');
   }
}

$foo = new foo();
$name = 'MyName';

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital 'A': \x41
EOT;
?>
```

http://www.php.net/manual/en/language.types.string.php#language.types.string.syntax.nowdoc

Unlike heredocs, nowdocs can be used in any static data context. The typical example is initializing class properties or constants

A nowdoc is specified similarly to a heredoc, but *no parsing is done* inside a nowdoc.

A nowdoc is identified with the same <<< sequence used for heredocs, but the identifier which follows is enclosed in single quotes, e.g. *<<<'EOT'*. All the rules for heredoc identifiers also apply to nowdoc identifiers, especially those regarding the appearance of the closing identifier.

Arrays

- Enumerated Arrays
- Associative Arrays
- Array Iteration
- Multi-Dimensional Arrays
- Array Functions
- SPL, Objects as arrays

Facts

- Numeric keys, either auto assigned or assigned by you, great when you don't care how the data is indexed
- Associative arrays frequently used with databases, or when there should be some key=>value association
- Multi-Dimensional arrays, or arrays of arrays can contain a lot of data, but can also be complicated to access
- keys containing only digits are cast to integers
- array keys are case-sensitive, but type insensitive

Iteration

- foreach() operates on a copy of the array
- foreach($array AS &$value) { … }
- end() – move pointer to last element
- key() - retreives key from current position
- next() – advances array pointer one spot then returns current value
- prev() – rewinds the array pointer one spot, then returns the current value
- reset() – resets the array pointer to the beginning of the array
- each() – returns the current key and value from the array, then iterates
- current()

Sorting

- bool sort (array &$array [, int $sort_flags = SORT_REGULAR])
  - SORT_REGULAR
  - SORT_NUMERIC
  - SORT_STRING
- rsort()
- ksort()
- asort()
- usort()

- bool natsort (array &$array)
- bool shuffle (array &$array)

Stacks & Queues

- Stack, implementing LIFO
  - int array_push (array &$array , mixed $var [, mixed $...])
  - mixed array_pop (array &$array)
- FIFO
  - int array_unshift (array &$array , mixed $var [, mixed $...])
  - mixed array_shift (array &$array)

Functions

- http://ru.php.net/array/
- bool array_walk (array &$array , callback $funcname [, mixed $userdata])
- bool array_walk_recursive (array &$input , callback $funcname [, mixed $userdata])
- array array_keys (array $input [, mixed $search_value [, bool $strict = false]])
- array array_values (array $input)
- array array_change_key_case (array $input [, int $case = CASE_LOWER])
- array array_merge (array $array1 [, array $array2 [, array $...]])
- array array_merge_recursive (array $array1 [, array $...])
- array array_splice (array &$input , int $offset [, int $length = 0 [, mixed $replacement]]) - cut out a chunk of an array
- bool array_key_exists (mixed $key , array $search)
- array array_flip (array $trans)
- array array_reverse (array $array [, bool $preserve_keys = false])
- array array_combine (array $keys , array $values)
- mixed array_rand (array $input [, int $num_req = 1])
- array array_diff (array $array1 , array $array2 [, array $ ...])
- array array_intersect (array $array1 , array $array2 [, array $ ...]
- bool in_array (mixed $needle , array $haystack [, bool $strict])
- array list (mixed $varname [, mixed $...])
- int count (mixed $var [, int $mode = COUNT_NORMAL])
- int extract (array $var_array [, int $extract_type = EXTR_OVERWRITE [, string $prefix]])

SPL, Objects as arrays

http://www.php.net/manual/en/class.arrayobject.php
http://www.php.net/manual/en/class.arrayiterator.php
http://www.php.net/manual/en/class.recursivearrayiterator.php

I/O

- Files
- Reading
- Writing
- File System Functions
- Streams
- Contexts

Files

- provide a simple temporary to permanent data store
- string file_get_contents (string $filename [, bool $use_include_path = false [, resource $context [, int $offset = -1 [, int $maxlen = -1]]]])
- int file_put_contents (string $filename , mixed $data [, int $flags = 0 [, resource $context]])
    - FILE_USE_INCLUDE_PATH
    - FILE_APPEND
    - LOCK_EX
- $fp = fopen(__FILE__, 'r')
- string fread (resource $handle , int $length)
- string fgets (resource $handle [, int $length])
- int fwrite (resource $handle , string $string [, int $length])
- array fstat (resource $handle)
- array stat (string $filename)
- bool fclose (resource $handle)
- bool feof (resource $handle)

Constants

- __LINE__ - The current line number of the file.
- __FILE__ The full path and filename of the file.
- __FUNCTION__ The function name. (Added in PHP 4.3.0) (case-sensitive)
- __CLASS__ The class name. (Added in PHP 4.3.0) (case-sensitive)
- __METHOD__ The class method name. (Added in PHP 5.0.0) (case-sensitive)

File Modes

- 'r' Open for reading only; place the file pointer at the beginning of the file.
- 'r+' Open for reading and writing; place the file pointer at the beginning of the file.
- 'w' Open for writing only; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.

- 'w+' Open for reading and writing; place the file pointer at the beginning of the file and truncate the file to zero length. If the file does not exist, attempt to create it.
- 'a' Open for writing only; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'a+' Open for reading and writing; place the file pointer at the end of the file. If the file does not exist, attempt to create it.
- 'x' Create and open for writing only; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning FALSE and generating an error of level E_WARNING. If the file does not exist, attempt to create it.
- 'x+' Create and open for reading and writing; place the file pointer at the beginning of the file. If the file already exists, the fopen() call will fail by returning FALSE and generating an error of level E_WARNING.

Locks

- bool flock (resource $handle , int $operation [, int &$wouldblock])
    - LOCK_SH - place Shared lock
    - LOCK_EX - place Exclusive (write) Lock
    - LOCK_UN - release lock
- Placing and removing locks can inhibit performance as traffic increases

Functions

- Access controll (Note that the results of a call to any of these functions will be cached, so that two calls to a given function on the same stream resource and during the same script will return the same value, regardless of whether the underlying resource has changed in the meantime)
    - is_dir($path); //Returns if path is a directory
    - is_executable($path); //Returns if path is an executable
    - is_file($path); //Returns if path exists and is a regular file
    - is_link($path); //Returns if path exists and is a symlink
    - is_readable($path); //Returns if path exists and is readable
    - is_writable($path); //Returns if path exists and is writable
    - bool is_uploaded_file (string $filename)
    - bool file_exists (string $filename) - Checks whether a file or directory exists.
    - void clearstatcache ([bool $clear_realpath_cache = false [, string $filename]])
- Links
    - bool link (string $target, string $link) - Create a hard link.
    - bool symlink (string $target, string $link) - creates a symbolic link to the existing *target* with the specified name *link*.
    - string readlink (string $path) - Returns the target of a symbolic link.
- float disk_free_space (string $directory)
- bool move_uploaded_file (string $filename, string $destination)
- int fseek (resource $handle, int $offset [, int $whence])
    - SEEK_SET (start from the beginning of the file)

    - SEEK_CUR (start from the current position)
    - SEEK_END (start from the end of the file)
- int ftell (resource $handle) - gives undefined results for append-only streams (opened with “a” flag).
- bool ftruncate (resource $handle, int $size)
- array fgetcsv (resource $handle [, int $length [, string $delimiter [, string $enclosure]]])
- int fputcsv (resource $handle, array $fields [, string $delimiter [, string $enclosure]])
- int readfile (string $filename [, bool $use_include_path [, resource $context]]) - Outputs a file.
- int fpassthru (resource $handle) - Output all remaining data on a file pointer
- array file (string $filename [, int $flags [, resource $context]])
- string implode (string $glue, array $pieces)
- string basename (string $path [, string $suffix])
- string realpath (string $path) - Returns canonicalized absolute pathname.
- bool chdir (string $directory)
- string getcwd (void)
- bool unlink (string $filename [, resource $context])
- bool rename (string $oldname, string $newname [, resource $context])- Attempts to rename *oldname* to *newname*.
- bool rmdir (string $dirname [, resource $context]) - Attempts to remove the directory named by *dirname*. The directory must be empty, and the relevant permissions must permit this.
- bool mkdir (string $pathname [, int $mode [, bool $recursive [, resource $context]]])
- string readdir (resource $dir_handle)- Read entry from directory handle.
- array scandir (string $directory [, int $sorting_order [, resource $context]]) - Returns an array of files and directories from the *directory*.
- int filectime (string filename) / int filemtime (string filename) - The "last changed time" differs from the "last modified time" in that the last changed time refers to any change in the file’s inode data, including changes to permissions, owner, group, or other inode-specific information, whereas the last modified time refers to changes to the file’s content (specifically, byte size).
- int fileperms (string filename)
- string tempnam (string $dir , string $prefix)

Streams

- Are the way that PHP handles working with network resources
- Whenever you open up a file PHP creates a stream in the background

Types

- provides access to a certain type of stream resource (built-in)
    - php.* — standard PHP input/output
    - file — standard file access

    - http — access to remote resources via HTTP
    - ftp — access to remote resources via FTP
    - compress.zlib — access to compressed data stream using the zlib compression library.
- stream filters - extensions that can be “installed” on top of the existing one to form chains of filters that act cumulatively on a data stream
    - string.rot13—encodes the data stream using the ROT-13 algorithm
    - string.toupper—converts strings to uppercase
    - string.tolower—converts strings to lowercase
    - string.strip_tags—removes XML tags from a stream
    - convert.*—a family of filters that converts to and from the base64 encoding.
    - mcrypt.*—a family of filters that encrypts and decrypts data according to multiple algorithms
    - zlib.*—a family of filters that compressed and decompresses data using the zlib compression library

Components

- file wrapper
- one or two pipe-lines
- an optional context
- metadata about the stream
- Functions
    - array stream_get_meta_data (resource $stream)
    - resource stream_context_create ([array $options [, array $params]])

Functions

- resource stream_socket_server (string $local_socket [, int &$errno [, string &$errstr [, int $flags [, resource $context]]]])
- array stream_get_transports (void)
- resource stream_socket_accept (resource $server_socket [, float $timeout [, string &$peername]])
- array stream_get_wrappers (void)
- bool stream_wrapper_register (string $protocol, string $classname)
- resource stream_socket_client (string $remote_socket [, int &$errno [, string &$errstr [, float $timeout [, int $flags [, resource $context]]]]])
- string stream_socket_get_name (resource $handle, bool $want_peer)
- bool stream_filter_register (string $filtername, string $classname)
- resource stream_filter_prepend (resource $stream, string $filtername [, int $read_write [, mixed $params]])
- resource stream_filter_append (resource $stream, string $filtername [, int $read_write [, mixed $params]])
- int stream_select (array &$read , array &$write , array &$except , int $tv_sec [, int $tv_usec = 0])

Sockets

- $fp = fsockopen("example.preinheimer.com", 80, $errno, $errstr, 30)

Contexts

http://php.net/manual/en/context.php

- resource **stream_context_create** ([array $options [, array $params]])
- bool **stream_context_set_option** (resource $stream_or_context , string $wrapper , string $option , mixed $value)
- bool **stream_context_set_option** (resource $stream_or_context , array $options)
- bool **stream_context_set_params** (resource $stream_or_context , array $params)

```php
<?php
$opts = array(
  'http'=>array(
    'method'=>"GET",
    'header'=>"Accept-language: en\r\n" .
              "Cookie: foo=bar\r\n"
  )
);

$context = stream_context_create($opts);

/* Sends an http request to www.example.com
   with additional headers shown above */
$fp = fopen('http://www.example.com', 'r', false, $context);
fpassthru($fp);
fclose($fp);
?>
```

Security

- Configuration
- Session Security
- Cross-Site Scripting
- Cross-Site Request Forgeries
- SQL Injection
- Remote Code Injection
- Email Injection
- Filter Input
- Escape Output
- Encryption, Hashing algorithms
- File uploads
- Data storage
- SSL

Levels

- Web Server level
- Data base level
- PHP itself

Concepts

- "with great power comes great responsibility" (yeah)
- Defense in Depth - plan for failure; applications remain more resistant over time;
- Principle of Least Privilege
- Terms: Validation vs Escaping
    - Validation or Filtering is the process by which you subject data to a series of rules, either it passes (validates) or does not
    - Escaping is the process by which you prepare data for a specific resource by “escaping” certain portions of the data to avoid confusion of instruction and data

Validate Input

- One of three things can be said about received data:
    - It is valid, the user entered the data you want, in the format you desire
    - It is invalid because the user either did not comply or did not understand the rules on the data you requested (eg. Poorly formatted phone number)
    - It is invalid because the user is attempting to compromise your system
- Data from the end user can not be trusted
- Validate data first don’t save it for last

- Fail early, tell the user what went wrong
- Whitelist vs Blacklist

Functions

- bool ctype_alnum (string $text) -- Check for alphanumeric character(s)
- ctype_alpha -- Check for alphabetic character(s)
- ctype_cntrl -- Check for control character(s)
- ctype_digit -- Check for numeric character(s)
- ctype_graph -- Check for any printable character(s) except space
- ctype_lower -- Check for lowercase character(s)
- ctype_print -- Check for printable character(s)
- ctype_punct -- Check for any printable character which is not whitespace or an alphanumeric character
- ctype_space -- Check for whitespace character(s)
- ctype_upper -- Check for uppercase character(s)
- ctype_xdigit -- Check for character(s) representing a hexadecimal digit

Register Globals

- Register globals was great, when you were learning, and no one else saw your applications
- on it’s own is not a security risk; combined with sloppy coding is the problem
- pre-initilize all your variables
- Code with E_STRICT enabled

Magic Quotes

- automatically escapes quote characters, backslash and null
- Gone completely in PHP6 (oO)
- Disabled by default in PHP5
- A pain in the neck, you end up with code like this:
- if (get_magic_quotes_gpc())
- {
- $string = stripslashes($string);
- }

Escaping Output

- string strip_tags (string $str [, string $allowable_tags]) – Remove anything that looks like an HTML tag from a string
- string htmlentities (string $string [, int $quote_style = ENT_COMPAT [, string $charset [, bool $double_encode = true]]]) – Convert any character that has a specific HTML entity into it
    - ENT_COMPAT - convert double-quotes and leave single-quotes alone
    - ENT_QUOTES - convert both double and single quotes

    - ENT_NOQUOTES - leave both double and single quotes unconverted
- string htmlspecialchars (string $string [, int $quote_style = ENT_COMPAT [, string $charset [, bool $double_encode = true]]]) – Convert &, “, ‘, <, > into their entities
- escape your data or use prepared statements

XSS

- attacker injects code into your page that contains code that re-writes the page to do something nefarious
- can be prevented through proper escaping and data validation

CSRF

- site exploits another sites persistent user trust relationship to make something happen
- iFrames

Sessions

- Basically, combine cookies containing a session ID with a local(ish) data store corresponding to that session id
- If the session id is compromised, or the data store is not secure (/tmp on a shared machine) sessions are still vulnerable to attack

Session Fixation

- either an attacker tricks another user into clicking on a link providing a session id, or an innocent user pastes a link containing their session id to someone they shouldn’t have
- don’t allow session IDs to come in over GET, and regenerate session ids when a user authenticates themself

Session Hijacking

- Session IDs are very random
- To defend, implement some browser fingerprinting

Command Injection

- PHP provides great power with the exec(), system() and passthru() functions, aswell as the ‘ (backtick) operator
- string escapeshellcmd (string $command)
- string escapeshellarg (string $arg)

Security Settings

- open_basedir – Restricts PHP’s file acces to one or more specified directores
- safe_mode – limits file access based on uid/gid of running script and file to be accessed. Slower, doesn’t work well with uploaded files.

- disable_functions
- disable_classes

Encryption, Hashing algorithms

http://php.net/manual/en/function.crypt.php
http://php.net/manual/en/function.md5.php
http://php.net/mcrypt
http://php.net/mhash

File uploads

1. Assign 775 permission to upload folder
2. Check the file using PHP functions (if its photo upload, for example)
3. Disable directory indexes and script exection (using .htaccess or server settings)
4. Place the upload folder outside WWW root.

Data storage

SSL

http://php.net/manual/en/book.openssl.php

Databases

- SQL
- Joins
- Analyzing Queries
- Prepared Statements
- Transactions
- PDO

Facts

- Recent versions of PHP no longer ship with the MySQL extension built in, you must enable it at configure time.
- We now have two versions of the MySQL extension available, MySQL and MySQLi
- The MySQLi extension makes some of MySQL's more recent functionality available, things like prepared statements

Analyzing Queries

- EXPLAIN EVERYTHING!
- EXPLAIN SELECT * FROM 07_amazon_monitor_rank_results WHERE asin = '0672324547';

Prepared Statements

- allow you to increase speed of repeated queries, and isolate data from command
- First you Prepare the statement, then you bind parameters to it, then you execute it
- http://php.net/manual/en/function.mysqliprepare.php
- Bound parameters
- The bound-parameter variant allows you to store a query on the MySQL server, with only the iterative data being repeatedly sent to the server, and integrated into the query for execution.
    - boolean mysqli_stmt_bind_param (mysqli_stmt stmt, string types, mixed &var1 [, mixed &varN)
    - class mysqli_stmt {
    - boolean bind_param (string types, mixed &var1 [, mixed &varN])
    - }
- Bound results
- The bound-result variant allows you to use sometimes-unwieldy indexed or associative arrays to pull values from result sets by binding PHP variables to corresponding retrieved fields, and then use those variables as necessary
    - boolean mysqli_stmt_bind_result (mysqli_stmt stmt, mixed &var1 [, mixed

&varN…])
    - class mysqli_stmt {
    - boolean bind_result (mixed &var1 [, mixed &varN])
    - }
- After a query has been prepared and executed, you can bind variables to the retrieved fields by using $stmt->bind_result.

Mysqli

- $link = mysqli_connect(“localhost”, “u”, “p”, “ex”);
- $city = “Montreal”;
- $stmt = mysqli_stmt_init($link);
- if ($stmt = mysqli_stmt_prepare ($stmt, “SELECT Province FROM City WHERE Name=?”))
- {
- mysqli_stmt_bind_param($stmt, “s”, $city);
- mysqli_stmt_execute($stmt);
- mysqli_stmt_bind_result($stmt, $province);
- mysqli_stmt_fetch($stmt);
- printf(“%s is in district %s\n”, $city, $province);
- mysqli_stmt_close($stmt);
- }
- mysqli_close($link);

Transactions

- allow you to merge multiple queries into one atomic operation, either they ALL execute successfully, or none do
    - BEGIN TRANSACTION #name;
    - … queries here
    - COMMIT;

PDO

- PHP Data Objects
- consistent object oriented method to interact with various databases
- database specific PDO driver must also be installed
- $pdoConnection = new PDO('mysql:host=localhost;dbname=example', USERNAME, PASSWORD);
- foreach ($pdoConnection->query("SELECT * FROM users") AS $user) { echo $user['id']; }
- prepared statement:
    - $query = "SELECT * FROM posts WHERE topicID = :tid AND poster = :userid";
    - $statement = $pdoConnection->prepare($query, array(PDO::ATTR_CURSOR, PDO::CURSOR_FWDONLY));
    - $statement->execute(array(':tid' => 100, ':userid' => 12));
    - $userAPosts = $statement->fetchAll();
    - $statement->execute(array(':tid' => 100, ':userid' => 13));

    - $userBPosts = $statement->fetchAll();
- $pdoConnection = null; // closing connection
- PDOStatement->nextRowset()
- array PDO::errorInfo (void)
    - 0 - SQLSTATE error code (a five characters alphanumeric identifier defined in the ANSI SQL standard)
    - 1 - Driver-specific error code.
    - 2 - Driver-specific error message.

SQLite

- is a database, without the database
- rather than using a separate program to persistently maintain the database, SQLite on the other hand requires the C libraries that comprise the DB to be built into whatever program would like to use them
- was built into PHP by default as of PHP5, which is cool, some twits turn it off, which isn't
- It's fast, free, and has nice licensing terms
- Apart from not needing to connect to a remote server or process, SQLite is no different from other database systems
- catagorizes data into textual and numeric

Object Oriented Programming

- Instantiation
- Modifiers/Inheritance
- Interfaces
- Exceptions
- Static Methods & Properties
- Autoload
- Reflection
- Type Hinting
- Class Constants
- Late Static Binding
- Magic (_*) Methods
- Instance Methods & Properties
- Class Definition
- SPL

Facts

- Prior to PHP5 OOP was a hack on top of the array implementation
- Slower than procedural code, but allows complex tasks to be understood more readily
- Objects are now dealt with by reference rather than by value, objects must be explicitly cloned to be copied

PPP & F

- Public – Method or property can be accessed externally
- Private – Method or property is private to that class and can not be accessed externally
- Protected – Method or property is private and can also be accessed by extending classes
- Final – Method can not be overridden by extending classes

Constants, Static

- are accessible as part of a class itself
- calling static properties using object notation will result in a notice
- by default the static method or property is considered public
- Class constants are public, and accessible from all scopes
- constants can only contain scalar values
- much cleaner code
- significantly faster than those declared with the define() construct

Interfaces & Abstract Classes

- New feature added to PHP 5
- used to create a series of constraints on the base design of a group of classes
- You must declare a class as abstract so long as it has (or inherits without providing a body) at least one abstract method.
- a class can only extend one parent class, but it can implement multiple interfaces.

Instanceof

- allows you to inspect all of the ancestor classes of your object, as well as any interfaces

Autoloading

- function __autoload($class_name)
- {
-     require_once "/www/phpClasses/{$class_name}.inc.php";
- }
- $a = new friend;

Special Functions

- http://php.net/manual/en/language.oop5.magic.php
- __construct()
- __destruct() - Destruction occurs when all references to an object are gone, and this may not necessarily take place when you expect it—or even when you want it to.
- __toString()
- __sleep()
- __wakeup()
- __call()
- __get()
- __set()

Exceptions

- Unified method of error handling
- Makes proper error handling possible
- Can be accomplished with try catch blocks or by setting a unified error handler once
- Try catch blocks take precedence if the error raising code is within them
- are objects, created (or “thrown”) when an error occurs
- All unhandled exceptions are fatal.
- catch() portion of the statement requires us to hint the type of Exception
- callback set_exception_handler (callback $exception_handler)
- bool restore_exception_handler (void)

SPL

- allow to stack autoloaders on top of each other
- void spl_autoload (string $class_name [, string $file_extensions = spl_autoload_extensions()])
- string spl_autoload_extensions ([string $file_extensions])
- bool spl_autoload_register ([callback $autoload_function [, bool $throw = true [, bool $prepend = false]]]) - first call to this function replaces the __autoload() call in the engine with its own implementation

Reflection API

- http://php.net/reflection
- provides a manner to obtain detailed information about code
- $func = new ReflectionFunction($func);
- ReflectionClass
- ReflectionMethod
- array get_defined_functions (void)
- array get_object_vars (object $object)

Type Hinting

http://www.php.net/manual/en/language.oop5.typehinting.php

Late Static Binding

http://php.net/manual/en/language.oop5.late-static-bindings.php

Data Format & Types

- XML Basics
- SimpleXML
- XML Extension
- Xpath
- Webservices Basics
- SOAP
- REST
- JSON & AJAX
- DateTime
- DOMDocument

XML

- Well Formed
  - Single root level tag
  - Tags must be opened and closed properly
  - Entities must be well formed
- Valid
  - Well formed
  - Contain a DTD
  - Follow that DTD

Parsing XML

- DOM
  - Document Object Model
  - Loads entire document into ram, then creates an internal tree representation
  - http://php.net/dom
- SimpleXML
  - All objects are instances of the SimpleXMLElement class
  - Uses DOM method
  - Very easy to use
  - Memory Intensive
  - http://php.net/simplexml

SimpleXml

- $xml = simplexml_load_string($library);
- object simplexml_load_file (string $filename [, string $class_name [, int $options [, string $ns [, bool $is_prefix]]]])

- SimpleXMLElement::xpath()
- Adding elements
  - SimpleXMLElement::addChild()
  - SimpleXMLElement::addAttribute()
- asXML()
- $library->book[0] = NULL; - to remove child elements
- children() - return iterator for subnodes
- attributes()
- Namespaces
  - SimpleXMLElement::getDocNamespaces()
  - SimpleXMLElement::getNamespaces()

DOM

- $dom = new DomDocument();
- $dom->load("library.xml");
- $dom->loadXML($xml);
- DomDocument::loadHtmlFile() and DomDocument::loadHTML()
- DomDocument::save() (to a file)
- DomDocument::saveXML() (to a string)
- DomDocument::saveHTML() (also to a string, but saves an HTML document instead of an XML file)
- DomDocument:saveHTMLFile() (to a file in HTML format).
- DomNode
  - DomDocument::createElement()
  - *DomDocument::createElementNS()*
  - DomDocument::createTextNode()
  - DomNode::appendChild()
  - DomNode::insertBefore()
  - DomNode::cloneNode()
  - DomNode::setAttributeNS()
- Removing
  - DomNode::removeAttribute()
  - DomNode::removeChild()
  - DomCharacterData::deleteData()
- dom_import_simplexml($sxml)
- simplexml_import_dom($dom);

XPath

- W3C Standard supported by many languages
- Combines the mojo of Regex with some SQL like results
- $xpath = new DomXPath($dom);
- A call to DomXpath::query() will return a DomNodeList object;
- http://www.w3schools.com/xpath/xpath_syntax.asp

XPath Searches

- xpath("item") - will return an array of item objects
- xpath("/bookshelf") - will return all children of the forum node
- xpath("//book") - will return an array of title values
- xpath(".") - will return the current node, <bookshelf>
- xpath("..") - will return an empty array, root node <bookshelf> does not have a parent element.

Web Services

- Generic umbrella that describes how disparate systems can integrate with each other over the web
- Most major sites offer some form of web services:
    - Amazon
    - FedEx
    - eBay
    - PayPal
    - del.icio.us
- Someone else has data you need
- Easier than scraping pages (also more reliable)
- Automate processes
- Provide additional information to your clients

REST

- Representational State Transfer
- Requests look like filled out forms, can use either GET or POST
- Response is an XML document
- Request and Response is defined by the service provider
- Services that use the REST architecture are referred to as RESTful services; those who use or provide RESTful services are sometimes humorously referred to as RESTafarians.
- http://library.example.com/api.php?devkey=123&action=search&type=book&keyword=style

SOAP

- Simple Object Access Protocol (Doesn't actually stand for anything anymore)
- Requests are sent using POST
- Requests are encapsulated within a SOAP Envelope
- Responses are XML documents with similar Envelope & Body sections
- WSDL Documents
    - SOAP web services are described by WSDL files

    - They are designed for automated (not human) consumption
    - Describes the methods offered by the web service, the parameters required, and the return type, as well as all complex types required by the service
- PHP5 has built in SOAP support
- The functions (generally) take a WSDL file as input, and create an object that mimics the services of the webservice:
    - $client = new SoapClient("http://soap.amazon.com/schemas2/AmazonWebServices.wsdl");
- API call:
    - $result = $client->KeywordSearchRequest($params);
- Debugging
    - $client = new SoapClient('http://api.google.com/GoogleSearch.wsdl', array('trace' => 1));
    - $client->__getLastRequestHeaders();
    - $client->__getLastRequest();
- PHP5 SOAP Server
    - doesn't include creation of WSDL files (NuSOAP does, Zend Studio does)
    - You can still provide a service either by using an external WSDL file, or merely defining the access methods
    - $options = array('uri' => 'http://example.org/soap/server/');
    - $server = new SoapServer(NULL, $options);
    - $server->setClass('MySoapServer');
    - $server->handle();

DateTime

http://php.net/manual/en/class.datetime.php
http://php.net/manual/en/book.datetime.php

JSON & AJAX

http://php.net/manual/en/book.json.php

Web Features

- Sessions
- Forms
- GET and POST data
- Cookies
- HTTP Headers
- HTTP Authentication

EGPCS

- Environment, Get, Post, Cookie, Server and Built-in variables

Forms

- GET (retrieve) vs POST (send)
- Form Tokens
- Default Values
- Re-populating data
- enctype="multipart/form-data" for files
- post_max_size
- max_input_time
- upload_max_filesize
- MAX_FILE_SIZE
- $_FILES[]
    - $_FILES['name']
    - $_FILES['type']
    - $_FILES['size']
    - $_FILES['tmp_name']
    - $_FILES['error']

Cookies

- Client side data store
    - Can be deleted, manipulated, copied
    - Behavior is inconsistent
    - allow your applications to store a small amount of textual data (typically, 4-6kB) on a Web client
- Header()
    - Set cookies manually, using the RFC for the appropriate headers
- setcookie()
    - Wraps the Header function, sets default values when nothing is passed.

- $_COOKIE[]
- Cookie values must be scalar.

Sessions

- Sessions, The safer way to state
- Use a cookie that contains a Session ID
- That Session ID corresponds with a local(ish) data store that contains the user's information
- The Session ID is the only data that is transmitted to a client, and is also the only thing that identifies them
- Session Hijacking and Session Fixation
- session.use_trans_sid
- session.auto_start
- session_start()
- $_SESSION[]
- bool session_set_save_handler (callback $open , callback $close , callback $read , callback $write , callback $destroy , callback $gc)

HTTP Headers

- void header (string $string [, bool $replace = true [, int $http_response_code]])
- Redirection
    - header("Location: http://phparch.com");
- Other arbitrary headers
    - Header injection attacks
    - Caching
        - header("Cache-Control: no-cache, must-revalidate");
        - header("Expires: Thu, 31 May 1984 04:35:00 GMT");
    - Content-Type
    - Meta Information
- bool headers_sent ([string &$file [, int &$line]])

Compression

- ob_start("ob_gzhandler");
- zlib.output_compression = on
- zlib.output_compression_level = 9

PHP input/output streams

- http://php.net/manual/en/wrappers.php.php
- php://stdin - STDIN
- php://stdout - STDOUT
- php://stderr - STDERR
- php://output

- php://input
- php://filter (available since PHP 5.0.0)
- php://memory (available since PHP 5.1.0)
- php://temp (available since PHP 5.1.0)

HTTP authentication with PHP

- http://php.net/manual/en/features.http-auth.php
- PHP_AUTH_USER
- PHP_AUTH_PW
- AUTH_TYPE

Design and Theory

- Design Patterns
- Code Reuse
- OOP Theory

To look

- SPL (standard PHP Library)
- PECL (PHP Extension Community Library)
- PEAR (PHP Extension and Application Repository).

SPL

- Standard PHP Library
- Extension available and compiled by default in PHP 5
- Improved in PHP 5.1
- http://www.php.net/~helly/php/ext/spl/
- The ArrayAccess interface
    - function offsetSet($offset, $value);
    - function offsetGet($offset);
    - function offsetUnset($offset);
    - function offsetExists($offset);
- The Iterator interface
    - function current();
    - function next();
    - function rewind();
    - function key();
    - function valid();
- The SeekableIterator interface
    - Iterator
    - function seek($index);
- The RecursiveIterator interface
    - Iterator
    - public RecursiveIterator getChildren (void)
    - public bool hasChildren (void)
- The FilterIterator

Code Reuse

- We solve the same problems every day
- We solve the same problems as our neighbor every day
- We solve the same problems on every project

- Code Reuse allows us to cut development time while increasing code quality

Design Patterns

- present a well defined solution to common problems
- provide a common language for developers

Factory

- is used to provide a common interface to a series of classes with identical functionality but different internals (think data storage, varying shipping systems, payment processing)
- The “factory” provides an instance of the appropriate class

Singleton

- to ensure that you have only one instance of a given class at a time
- myriad of circumstances; resource connections (database, file, external) most notably

Registry

- An extension of the Singleton Pattern, that allows for differing functionality based on some input data

Active Record

- Encapsulates a data source, allowing external code to concentrate on using the data while the active record pattern provides a consistent interface, hiding the work that goes into iterating over records, making changes etc.

MVC

- Model-View-Controller
- Complex pattern, the user initiates an action via the controller, which interfaces with the model, finally the view is called which takes care of dealing with the user interface
- allows for modularity in code

Sources

1. Zend PHP 5.0 Certification Course by Paul Reinheimer
2. Zend PHP 5 Certification Study Guide
3. Zend PHP 5.3 Study Guide v1
4. Read The Web Blog
5. http://www.php.net/manual/en/
6. http://zend-php.appspot.com/

This document was prepared by Roman Lapin
site: http://victimofbabylon.com
twitter: @memphys