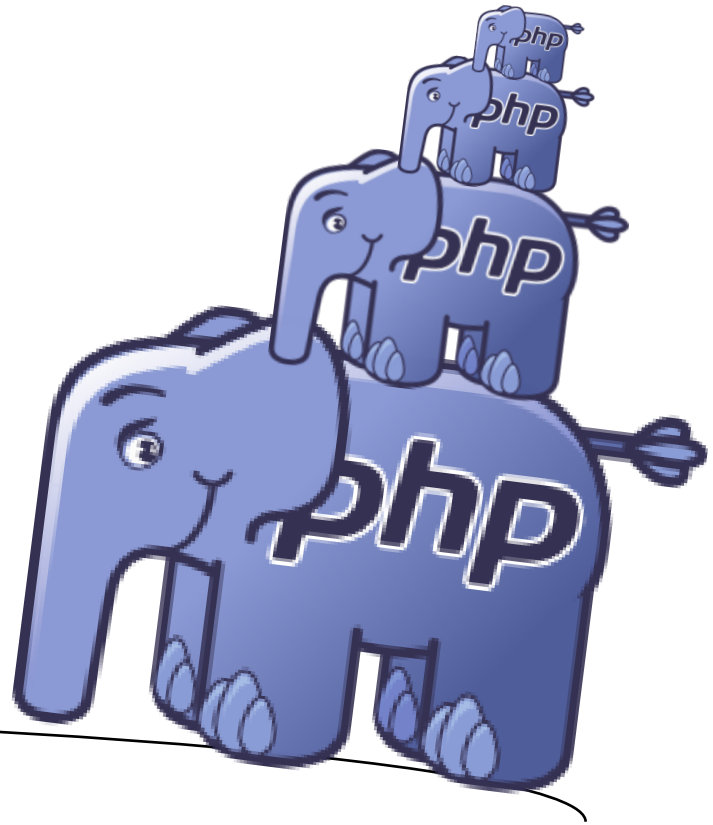
PHP Basics



STRINGS

ARRAYS

EASTERN WASHINGTON UNIVERSITY

Printing Options

`echo`	Language Construct Can use with or without `()` Multiple items separated by commas
`print()`	Function to send one thing to browser
`printf()`	Formatted print Similar to C equivalent First argument is format string • Use `%x` syntax for each type specifier • Can specify length, #digits, zero-padding, etc.
`print_r()`	Intelligently display parameter • Does not do well with `false`
`var_dump()`	Displays any PHP value in human readable format

Strings

`strlen()` – function to return length of string

Use string offset syntax to address individual character

```
$string = “Howdy”;
$last = strlen($string) – 1;
printf(“The last character is: %s\n”, $string{$last});
```

Cleaning Strings

Remove whitespace characters from string ends

`trim(<string> [, charlist])` – both ends

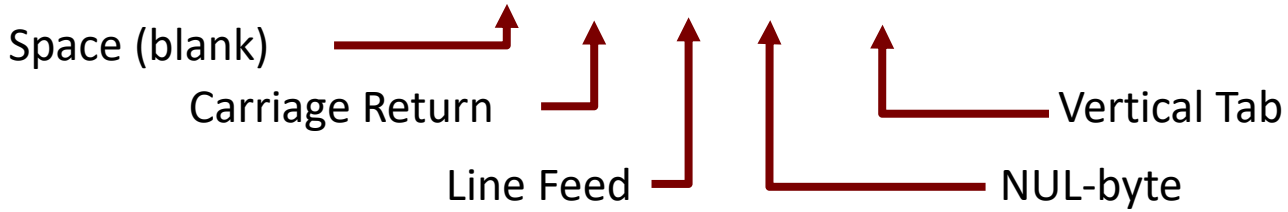
`ltrim (<string> [, charlist])` – left end

`rtrim (<string> [, charlist])` – right end

trim() Example

charlist is (optional) string of whitespace charaters to remove

```
$record = “  Fred\tFlintstone\t35\tWilma\t\n”;
$record = trim($record, “ \r\n\0\x0B”);
```




```
// $record is "Fred\tFlintstone\t35\tWilma\t"
```

Case Conversion Functions

`strtolower()` and `strtoupper()`

- operate on entire string

`ucfirst()`

- operates only on the first character of the string

`ucwords()`

- operates on the first character of each word in the string.

String Comparison

`==` Compares values; converts string to numbers if comparing against numeric

`===` Must be same type, too

```
$string = “PHP Rocks”;
$number = 5;
if ($string < $number) {
    echo(“{$string} < {$number}\n”);
}
```

Will print
`$string` is cast to number `0`

String Comparison

`strcmp($s1, $s2)`	Compare two strings as strings
	`-1` if `$s1` sorts BEFORE `$s2`
	`0` if EQUAL (same value)
	`+1` if `$s1` sorts AFTER `$s2`

`strcasecmp($s1, $s2)`	Convert `$s1`, `$s2` to all lower case for compare

Substring Functions

`substr(<string>, <start> [,<length>])`

- Copy out substring from `start` for `length` characters

`substr_count(<big string>, <small string>)`

- Return the number of times `<small string>` appears in `<big string>`

`strpos(<string>, <small string>)`

- Find position of first occurrence of `<small string>` in `<string>`

URL Parse

parse_url(*<URL string>*)

- Returns an array of components from URL

```
$bits = parse_url(“http://me:secret@exmaple.com/cgi-bin/board?user=fred”);

print_r($bits);
```

Regular Expressions

PHP has almost complete support for Perl regular expression features

- If interested, look up on your own

Arrays

Some Basics

Use `array()` construct to start array

```
$fibonacci = array(1, 1, 2, 3, 5, 8, 13);
$prez = array('first' => "Washington",
              'second' => "Adams",
              'third' => "Jefferson");
```

Add new elements

```
$fibonacci[] = 21;
$prez['fourth'] = "Madison";
```

EASTERN WASHINGTON UNIVERSITY

Assigning a Range of Values

`range()` function creates array of consecutive integer or character values

```
$numbers = range(2, 5)
$letters = range('a', 'z');
$reverse_numbers = range(5, 2);

$by_tens = range(0, 100, 10);
```

Size of an Array

`count()` and `sizeof()` return the number of elements in the array

```
$by_tens = range(0, 100, 10);
echo count($by_tens), “\n”;

$letters = range(‘a’, ‘z’);
echo sizeof($letters), “\n”;
```

Multidimensional Array

Values in arrays can be arrays themselves

Can use additional [ ] to index elements in array of arrays

```
$multi = array(array(1, 2, 3),
               array(4, 5, 6),
               array(7, 8, 9));

echo $multi[2][0], “\n”
```

Extracting Multiple Values

Use the `list()` construct to copy multiple values out of an array

```
$person = array("Peter", 26, "Programmer");

list($name, $age, $job) = $person;
```

If fewer variables in list than array values, ignore extras

If more variables that values, extra values set to NULL

Useful when dealing with one row of database query

Keys and Values

`array_keys(`*`$array`*`)`	Returns array of keys from *`$array`*
`array_values(`*`$array`*`)`	Returns array of values from *`$array`*
`array_key_exists(`*`key, $array`*`)`	Returns Boolean: if *`$key`* is in *`$array`*

NOT sufficient: `if ($person["name"]). . .`

- If exists, might be `false` value

Converting Between Arrays and Variables

`extract($array)`	Automatically create local variables, using keys, with values assigned
`compact(<variables>)`	Combines named variables (as strings) into array with names as keys and assigned values as values

```
$color = "indigo";
$shape = "curvy";
$floppy = "none";
$a = compact("color", "shape", "floppy");
// or
$names = array("color", "shape", "floppy");
$a = compact($names);
```

Iterator Functions

Arrays have iterators

`current()` – return current element

`reset()` – sets iterator to FIRST and return value

`next()` – advance iterator & return value

`prev()` – back up one element & return value

`end()` – move to LAST and return value

`key()` – return key of current

`each()` – return key and value of current as array & advance

Iterator Example

```
$name["Tinker"] = "Percy Alleline";
$name["Tailor"] = "Bill Haydon";
$name["Soldier"] = "Tony Bland";
$name["Poor Man"] = "Toby Esterhase";
$name["Beggarman"] = "George Smiley";

reset($name);

while (list($key, $value) = each($name)) {
    echo “{$key} is {$value}<br />\n”;
}
```

Traversing Arrays

`array_walk($array, <callable function>)`

- Executes user-defined function once per element

Function parameters:

1. Element's value
2. Element's key
3. (Optional) Value supplied to `array_walk` when it is called

Table from Array Example

```
$printRow = function ($value, $key)
{
 print("<tr><td>{$key}</td><td>{$value}</td></tr>\n");
};

$new_ages = array('Fred' => 35,'Barney' => 30,
                  'Tigger' => 8,'Pooh' => 40);

echo "<table border=1>\n<tr><th>Person</th><th>Age</th></tr>\n";

array_walk($new_ages, $printRow);

echo "</table>\n";
```

Array Reduction

```
array_reduce ($array, <callable function>
[, <seedvalue>])
```

- Applies function to each element to build a single value

Function parameters:

1. Running total
2. Current value

Array Reduce Example

```
$numbers = range(1, 10);
shuffle($numbers);      // randomly order the array elements
echo "\nContents of array: ";
array_walk($numbers, function ($value, $key){
    echo "$value ";
});
echo "\n";

$max_value = array_reduce($numbers, function ($r, $c){
    return ($r > $c) ? $r : $c;
});

echo "Largest value = ", $max_value, "\n";
```

Searching for Values

`in_array(<to find>, $array)`

- Returns Boolean on whether `<to find>` is stored in `$array`

`array_search(<to find>, $array)`

- Returns key of element if `<to find>` is stored in `$array`

Sorting Arrays

General in-place sorting

	Sort Ascending	Sort Descending	User-Defined Order
Sort array by values, then reassign indices starting with 0	`sort($array)`	`rsort($array)`	`usort($array, <callable>)`
Sort array by values	`asort($array)`	`arsort($array)`	`ausort($array, <callable>)`
Sort array by keys	`ksort($array)`	`krsort($array)`	`kusort($array, <callable>)`

Sort arrays of string containing numbers

`natsort($array)` `natcasesort($array)`

Miscellaneous Functions

array_reverse()	Reverse the order of elements
array_flip()	Flip the order of key-value to value-key
array_sum()	Sum up values of indexed or associative arrays
array_merge()	“Concatenate” two or more arrays
array_diff()	Find elements of first not in second (or more) arrays