PHP

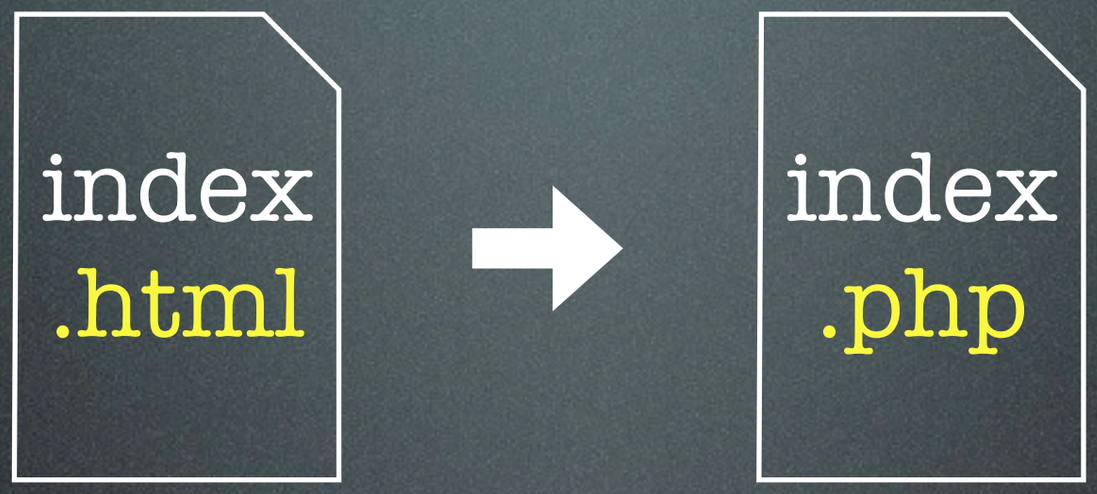
PHP

Hypertext Preprocessor

php

- designed for server-side html programming
  - but a language in its own right
- usually processes files
  - does have command-line interface

php



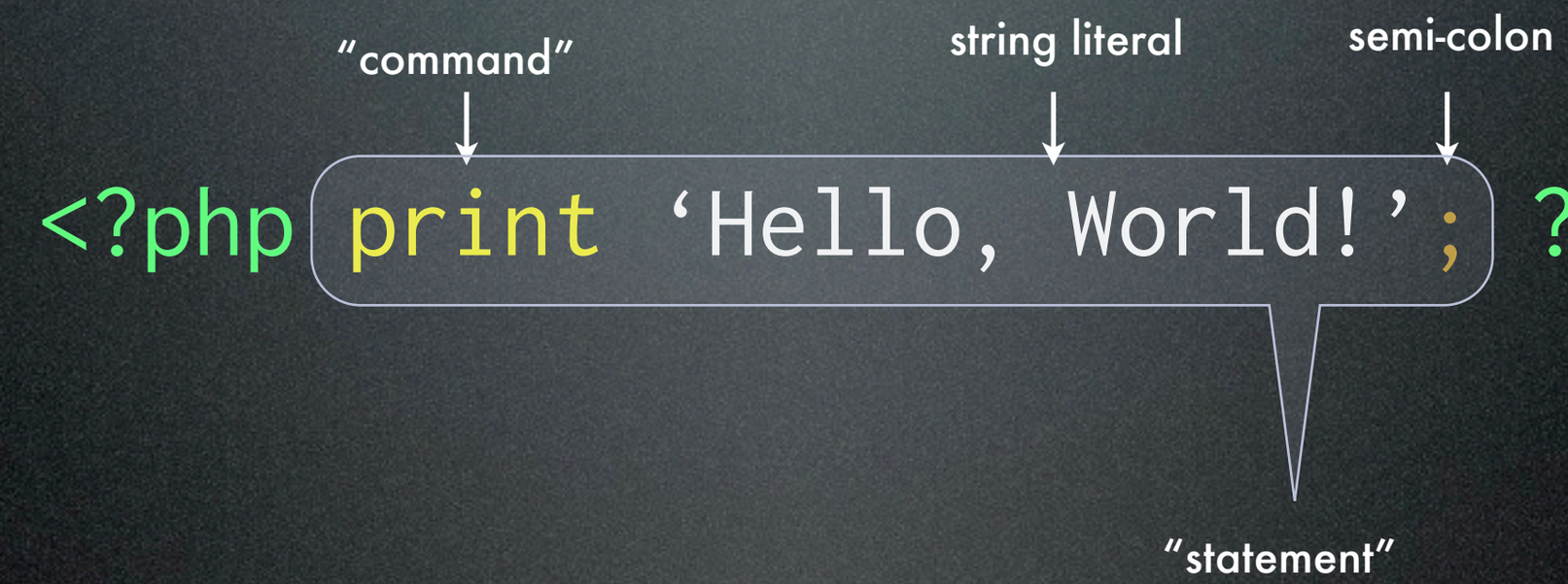

```
<?php     /* magical PHP awesomeness goes here! */    ?>
```

"open tag"

"close tag"

technically called Processing Instructions

php


"command"
string literal
semi-colon
<?php print 'Hello, World!'; ?>
"statement"

php

```
<?php

print ‘Hello, World!’;

print ‘Today\’s date is Monday, March 3 2014.’;

?>
```

“escape sequence”

php

```
<?php

print ‘Hello, World!’;
print ‘<br>’;
print ‘Today\’s date is Monday, March 3 2014.’;

?>
```

php

```php
<?php

echo ‘Hello, World!’;
echo ‘<br>’;
echo ‘Today\’s date is Monday, March 3 2014.’;

?>
```

php

```
<?php

print ‘Hello, World!’;
print ‘<br>’;
print ‘Today\’s date is ’.date('l, F j Y').’.’;

?>
```

built-in function invocation

string concatenation operator

argument to function

php

- http://php.net/date

php exercise

- Have your Hello, World page display the current date like this:

Monday, March the 3rd, 2014

using `date()`

php: conditional

```php
<?php

if (date('D') == 'Mon') {
    print '<em>I love Mondays!!!</em>';
}

?>
```

php: conditional

```
<?php if (date('D') == 'Mon') : ?>
  <p>I love Mondays!!!</p>
<?php else :  ?>
  <p>Bleh.</p>
<?php endif;  ?>
```

PHP mode

HTML mode

PHP mode

HTML mode

PHP mode

php exercise

- using the date function have your Hello, World page
- load a “light” css stylesheet during the morning
- load a “dark” css stylesheet during the night.

optional arguments

- date() takes an optional argument: a timestamp
- if you don't pass that argument, date defaults to using the current time
- how could you use this?

php exercise

- using date() and strtotime() print

  In 82 days it will be _____

  with the correct answer in any format.

php

```
<?php

print ‘Hello, World!’;
print ‘<br>’;
print ‘Today\’s date is ’.date(‘l, F j Y’).’.<br>’;
print pickles pickles pickles!

?>
```

php

- errors
  - one possibility: white screen
    - find error log output
    - may have to configure PHP
- or: error print out onscreen
  - but insecure on production

php

```
<?php

print ‘Hello, World!’;
print ‘<br>’;
print ‘Today\’s date is ’.date('l, F j Y').’.<br>’;
print ‘pickles pickles pickles!’;

?>
```

php

```
<?php

print ‘Hello, World!’;
print ‘<br>’;
print ‘Today\’s date is ’.date('l, F j Y').’.<br>’;
print ‘pickles pickles pickles!<br>’;
print ‘1 + 1 is ‘.(1 + 1);

?>
```

php

```
<?php

print ‘
  Hello, World!<br>
  Today\’s date is ’.date(’l, F j Y’).’.<br>
  pickles pickles pickles!<br>
  1 + 1 is ‘.(1 + 1)
;

?>
```

php

- http://php.net/error_log

php exercise

- use error_log
- figure out where it prints its argument

variables

variables

- like a bucket

variables

- like a bucket
- with a name tag

variables

- like a bucket
- with a name tag
- good for storing things

variable assignment

```
<?php
$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```

output

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```

output

note! doesn't exist until after 1st statement!

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```

output

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```


```
output

3
```

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```


```
output

3
9
```

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```


```
output

3
9
```

variable assignment and use

```
<?php

$sum = 1 + 2;
echo $sum.'<br>';
$sum = 4 + 5;
echo $sum.'<br>';
$sum = $sum + $sum;
```


```
output

3
9
```

```php
<?php

$nPickles = 20;
$nMorePickles = 10;
$totalPickles = $nPickles + $nMorePickles;

echo $totalPickles;
```

```php
<?php

$nPickles = 20;
$nMorePickles = 10;
$totalPickles = $nPickles + $nMorePickles;

echo $totalPickles;
```

```php
<?php

$nPickles = 20;
$nMorePickles = 10;
$totalPickles = $nPickles + $nMorePickles;

echo $totalPickles;
```

```php
<?php

$nPickles = 20;
$nMorePickles = 10;
$totalPickles = $nPickles + $nMorePickles;

echo $totalPickles;
```

variables and function arguments

```php
<?php

function add($a, $b) {
  $result = $a + $b;
  return $result;
}
```

variables

- variable names start with $ in php
- in Javacript, they do not

iteration in php

iteration

initializer

end condition

step

```
<?php
for ($i = 0; $i < 3; $i++) {
  print ‘pickle ’;
}
```

exercise in php

- write code within a web page that prints out

pickle

20 times

arrays in php

“Gherkin”	“Polish”	“Hungarian”	“Swedish”
[0]	[1]	[2]	[3]

two types of arrays

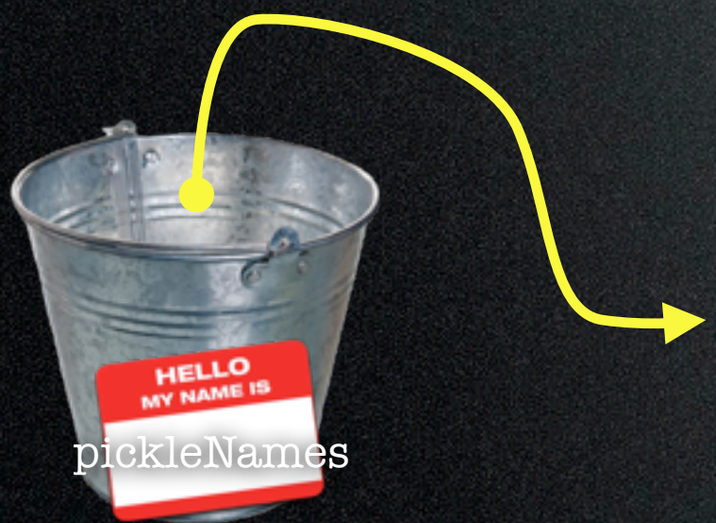
- linear (or “numeric”)
  - items referenced by number
- associative
  - items references by name

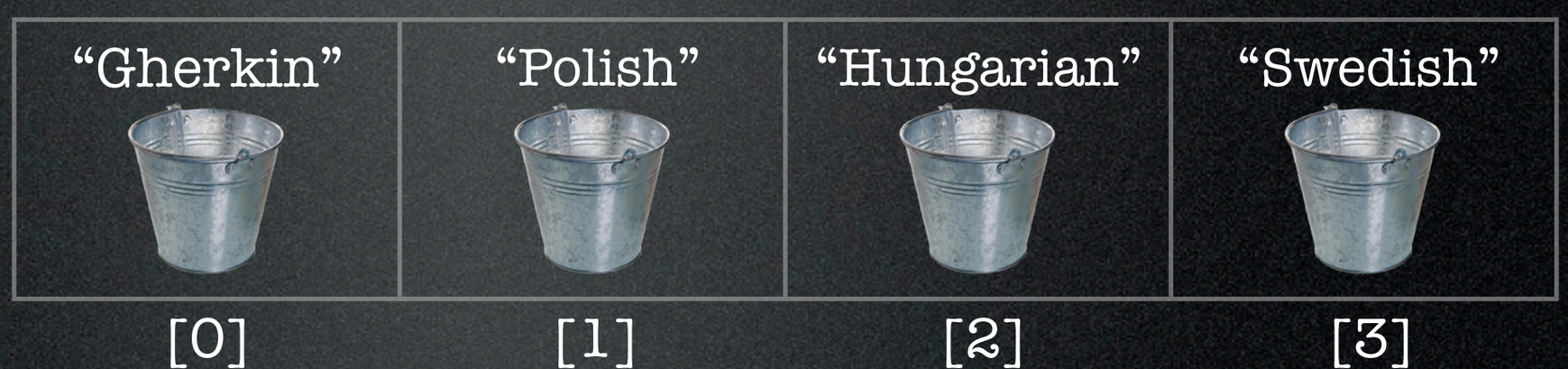
php: linear arrays

arrays: linear

```
<?php

$pickleNames = array(
    ‘Gherkin’,
    ‘Polish’,
    ‘Hungarian’,
    ‘Swedish’
);
```





php: linear arrays

access to linear array

```
<?php

$pickleNames = array(
[0]     ‘Gherkin’,
[1]     ‘Polish’,
[2]     ‘Hungarian’,
[3]     ‘Swedish’
    );

    print $pickleNames[2];
```

zero-based index

php

iteration over linear array

```
<?php

$pickleNames = array(
     ‘Gherkin’,
     ‘Polish’,
     ‘Hungarian’,
     ‘Swedish’
);

foreach ($pickleNames as $pickleName) {
    print $pickleName.’, ‘;
}

for ($i = 0; $i < count($pickleNames); $i++) {
   print $pickleNames[$i].’, ‘;
}
```

variable takes on the value of each item in turn.

php: associative arrays

arrays: associative

```
<?php

$pickleOrigins = array(
    ‘Gherkin’ => ‘West Indian’,
    ‘Polish’ => ‘Poland’,
    ‘Hungarian’ => ‘Hungary’,
    ‘Swedish’ => ‘Sweden’
);
```

keys

values

php

arrays: associative

```
<?php

$pickleOrigins = array(
    ‘Gherkin’ => ‘West Indian’,
    ‘Polish’ => ‘Poland’,
    ‘Hungarian’ => ‘Hungary’,
    ‘Swedish’ => ‘Sweden’
);
```

“Gherkin”	“Polish”	“Hungarian”	“Swedish”
“West Indian”	“Poland”	“Hungary”	“Sweden”

php

access to associative array

```
<?php

$pickleOrigins = array(
    ‘Gherkin’ => ‘West Indian’,
    ‘Polish’ => ‘Poland’,
    ‘Hungarian’ => ‘Hungary’,
    ‘Swedish’ => ‘Sweden’
);

print $pickleOrigins[‘Hungarian’];
```

php

iteration over associative array

```
<?php

$pickleOrigins = array(
      ‘Gherkin’ => ‘West Indian’,
      ‘Polish’ => ‘Poland’,
      ‘Hungarian’ => ‘Hungary’,
      ‘Swedish’ => ‘Sweden’
);
```

becomes key

becomes value

```
foreach ($pickleOrigins as $pickleName => $pickleOrigin) {
    print $pickleName.’ comes from ‘.$pickleOrigin.’<br>’;
}
```

php exercise

- exercise: write code that prints out pickles and their country of origin
  - use an associative array as the data source
  - use definition lists (DL, DT, DD) as markup

php

- HTTP form input
  - shows up in two associative arrays:
    - $_GET or $_POST
      - depending on the form's method attribute ("get" or "post")
      - (defaults to $_GET)

php

```
<form>
  <label>
    <span>How many pickles would you like to make?</span>
    <input type="text" name="number-of-pickles">
  </label>
  <input type="submit" value="MAKE ME SOME PICKLES">
</form>
```

php

```
<form>
  <label>
    <span>How many pickles would you like to make?</span>
    <input type="text" name="number-of-pickles">
  </label>
  <input type="submit" value="MAKE ME SOME PICKLES">
</form>
```

```
<dl>
<?php
  foreach ($_GET as $key => $value) {
    print '
      <dt>'.$key.'</dt>
      <dd>'.$value.'</dd>
    ';
  }
?>
</dl>
```

php

```
<form>
  <label>
    <span>How many pickles would you like to make?</span>
    <input type="text" name="number-of-pickles">
  </label>
  <input type="submit" value="MAKE ME SOME PICKLES">
</form>

<pre>
  <?php print_r($_GET); ?>
</pre>
```

php

- exercise: pickle-maker app
- print out “ PICKLE! ” for the number of pickles requested.
- extra credit: number the pickle output

```
<!doctype html>
<html>
<head>
    <title>Pickle-Maker</title>
</head>
<body>
    <form>
        <label>
            <span>How many pickles would you like to make?</span>
            <input type="text" name="number-of-pickles">
        </label>
        <input type="submit" value="MAKE ME SOME PICKLES">
    </form>
</body>
```

GET vs POST

- what is the difference between get and post?
- when would your form use one or the other?