PHP 1

Introduction

Owen Mundy | Spring 2011

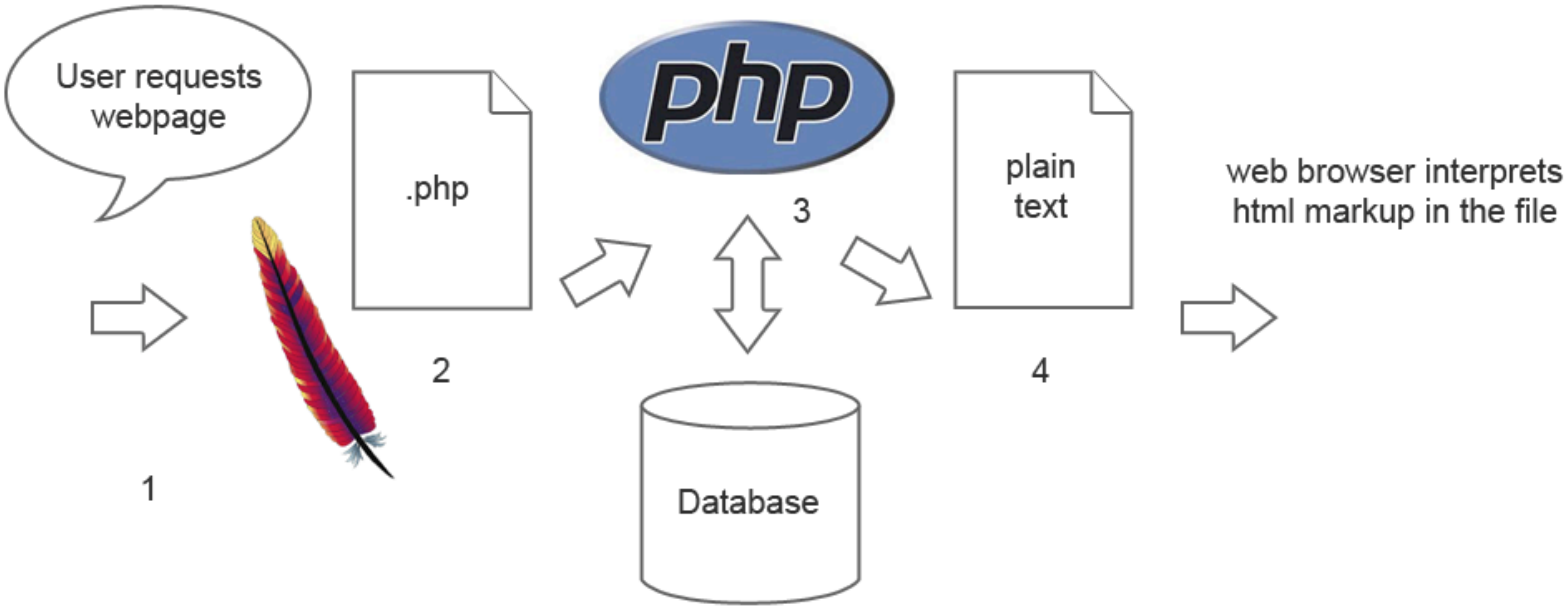
Overview

- Introduction
- Installation
- Syntax
- Variables

What is PHP?

- Simple yet powerful open source server-side scripting language designed for creating dynamic web content
- Integrates easily with HTML
- Can generate many document formats in addition to HTML (PDF, GIF, JPG, and PNG images, Flash movies, XML)
- Wide range of database support (e.g. MySQL, PostgreSQL, Oracle)
- Non-compiled: Constructed using single text pages on a server (unlike Java or C++ for example)

How does it work?



1. User requests .php file
2. Apache (web server) sends file to the PHP interpreter
3. Interpreter runs the script, connecting to a database if needed
4. and outputs plain text.

History

- Created by Rasmus Lerdorf in 1995 as “Personal Home Page” creator
- Can run on any operating system, even your personal computer
- Installed on more than 20 million websites and 1 million web servers
- Latest major release: PHP 5.2.6 (May 1, 2008)
- Many popular websites: myspace.com, facebook.com, etc.

Installation

Most common web servers use LAMP (Linux Apache MySQL PHP). PHP files can be uploaded with FTP and tested in a web browser or shell.

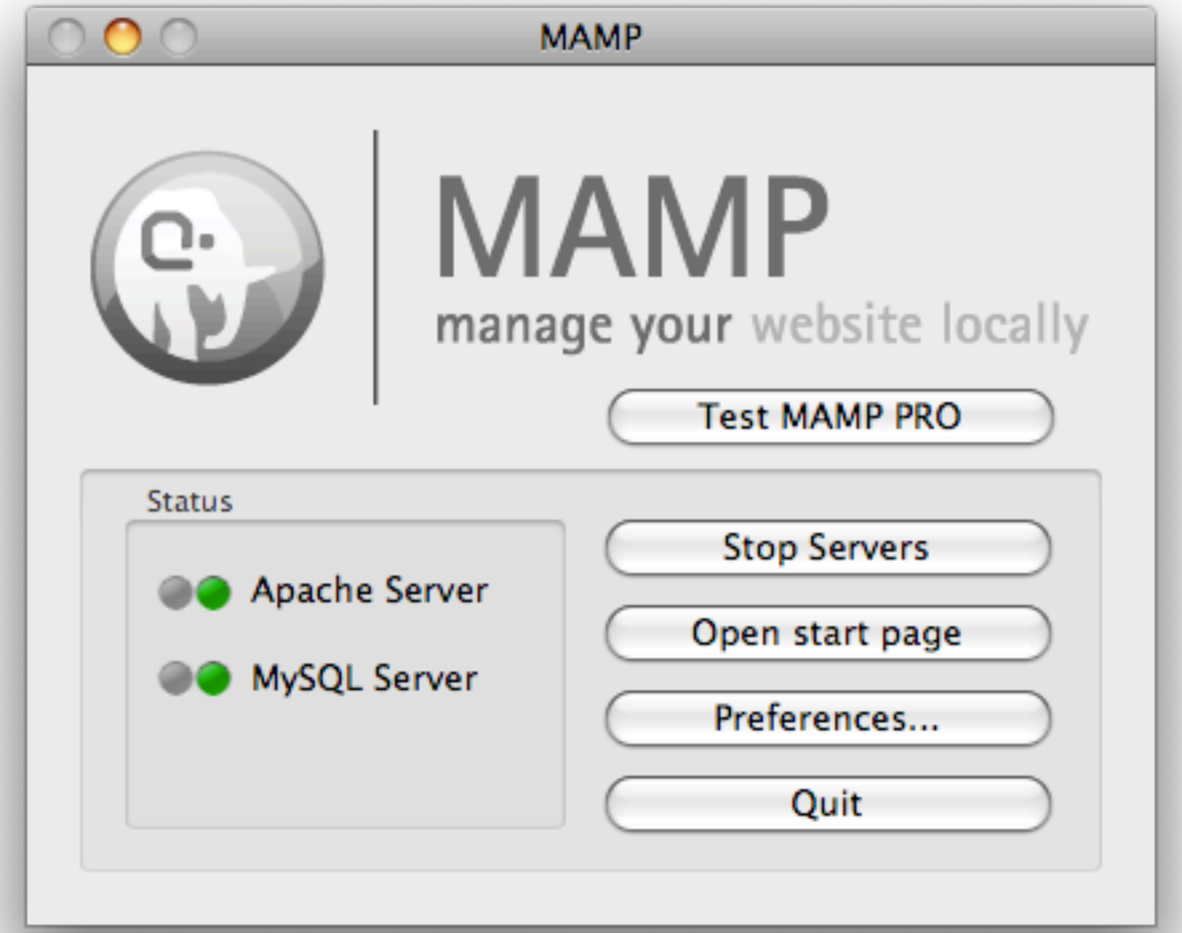
PHP can also be installed on any Mac, Windows, or Linux personal computer.

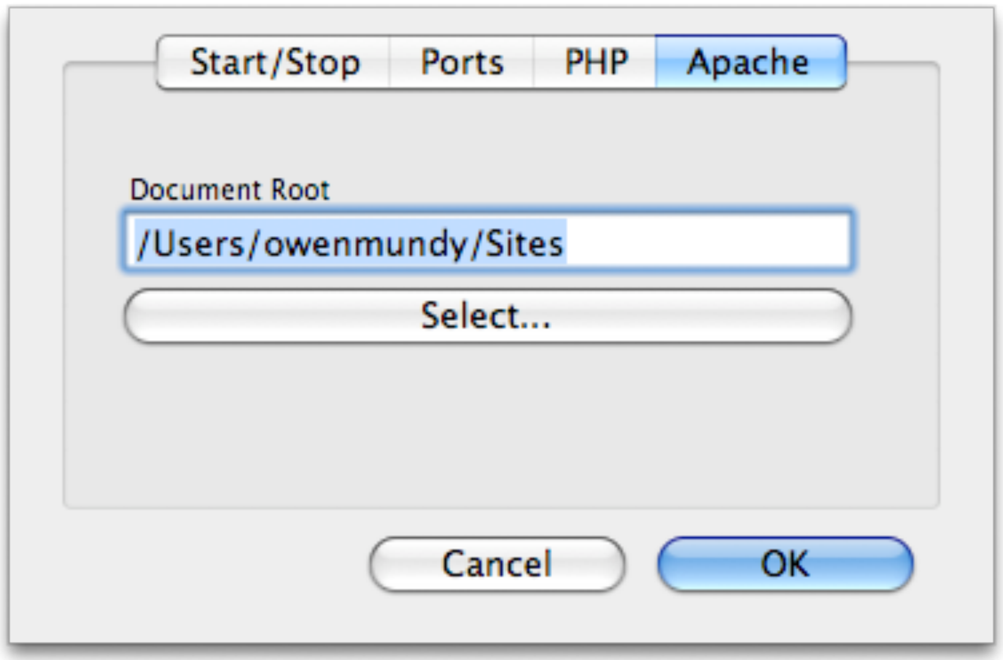
Macintosh: PHP is installed by default, but MySQL is not.
Try: MAMP (Mac Apache MySQL PHP) http://www.mamp.info

For Windows try WAMP http://www.wampserver.com
or XAMPP (for Linux too!) http://www.apachefriends.org

MAMP Configuration

Once you've installed MAMP you have to turn it on.
Set Document Root to ~/username/Sites folder.
Access it here: http://localhost/





Document Root: /Users/owenmundy/Sites

Language Basics

PHP is strongly influenced by other programming languages, therefore the concepts behind the lexical structure of PHP can be easily applied to/from other languages, e.g. ActionScript, Java or Processing.

```
<?php

echo (“Hello World!”);

?>
```

As in HTML, whitespace doesn’t matter in a PHP script, you can write statements across any number of lines or put them in a single line.

Syntax

Elements of the language and how they are structured.

– Statement —> Must end in a semicolon
– Statement terminator (;)
– Function (Command)
– Parameter

```
      Statement

    echo("Hello World!");
   |      |    |
 function name    parameter    statement terminator


  print ("Hello World!");      <— print() is the same as echo()
  |      |    |
 function name    parameter    statement terminator
```

Variables

– Used to store values
– Have a name/identifier and a value
– Assign values using the equal sign (=)
– Non data-typed so you can store any type of data (unlike many other programming languages)

```
$greeting = “Hello World!”;
echo($greeting);
```

Variable naming conventions

– Always begin variable name with a dollar sign ($)
– Use only letters, numbers, and underscores
– Do not begin with special characters or number
– Case sensitive ($var_name is not the same as $var_Name
– Separate multiple words with underscores (no hyphens or spaces)

```
$name = "Daniel";  // Declare and assign
$number = 32;   // Declare and assign
$counter = 12;   // Assign variable
echo($number);
echo($name);
echo($counter);
```

PHP 2

Data types, If...Else

Owen Mundy | Spring 2011

Overview

- Keywords
- Data Types
- Relational Operators
- Conditional Statements

Keywords

A keyword is a word reserved my the PHP programming language for its core functionality. You cannot give a variable, function, or class name the same name as a keyword. Your software will represent keywords in appropriate color to let you know.

Selected keywords are for example:

echo, print
and, not, or
do, for, while
if, else, elseif, switch
new
true, false
eval
return
...

Data types

Number (1,2,3,...)

```
20038, -33, 2 // Integer, whole numbers
3.1254, 0.1 //Floating Point, numeric values with decimal digits
```

Boolean (true or false only)

```
false, true, 0, “”  // Boolean, “truth value” of conditional statement
```

String (text)

```
‘hot dog’, “Daniel” // String, sequence of characters
```

Relational Operators

Used to compare values:

```
>  (greater than)
<  (less than)
>= (greater than or equal to)
<= (less than or equal to)
!= (inequality)
== (equality)
```

Relational Operators

```
5 > 4    // True
5 < 3    // False
5 > 5    // False
5 >= 5   // True
5 >= 6   // False
5 != 5   // False(not equal)
5 == 5   // True
5 == 4   // False
```

Conditional Statements

Used to make decisions about which code to execute and which to ignore. E.g.:

```
$guess = 7;    // a single equal sign *assigns* the value
$number = 7;

// while a double equal signs test to see if a condition is true
if ($guess < $number){
    echo "Too low!";
} elseif ($guess > $number){
    echo "Too high!";
} else {
    echo "You win!";
}
```

Switch Statements

Switches work like if statements, but with more efficiency and readability. If a condition is true, it executes a block of code...

```
switch (2) {
   case 0:
      echo 'The value is 0';
      break;
   case 1:
      echo 'The value is 1';
      break;
  case 2:
      echo 'The value is 2';
      break;
  default:
      echo "The value isn't 0, 1 or 2";
}
```

Switch Statements

- Accept only one argument (the value we are testing).
- The keyword case is followed by the value being compared and a colon (:).
- The code after that is executed.
- Then we call “break” so that all the other code isn’t executed too.

```
switch (2) {
   case 0:
      echo 'The value is 0';
      break;
   case 1:
      echo 'The value is 1';
      break;
  case 2:
      echo 'The value is 2';
      break;
  default:
      echo "The value isn't 0, 1 or 2";
}
```

Note

In order to understand PHP quickly and effectively, a sound understanding of HTML is required.

Refer to appropriate resources to revisit concepts of HTML, JavaScript and CSS if required.

PHP 3

Arrays & loops

Owen Mundy | Spring 2011

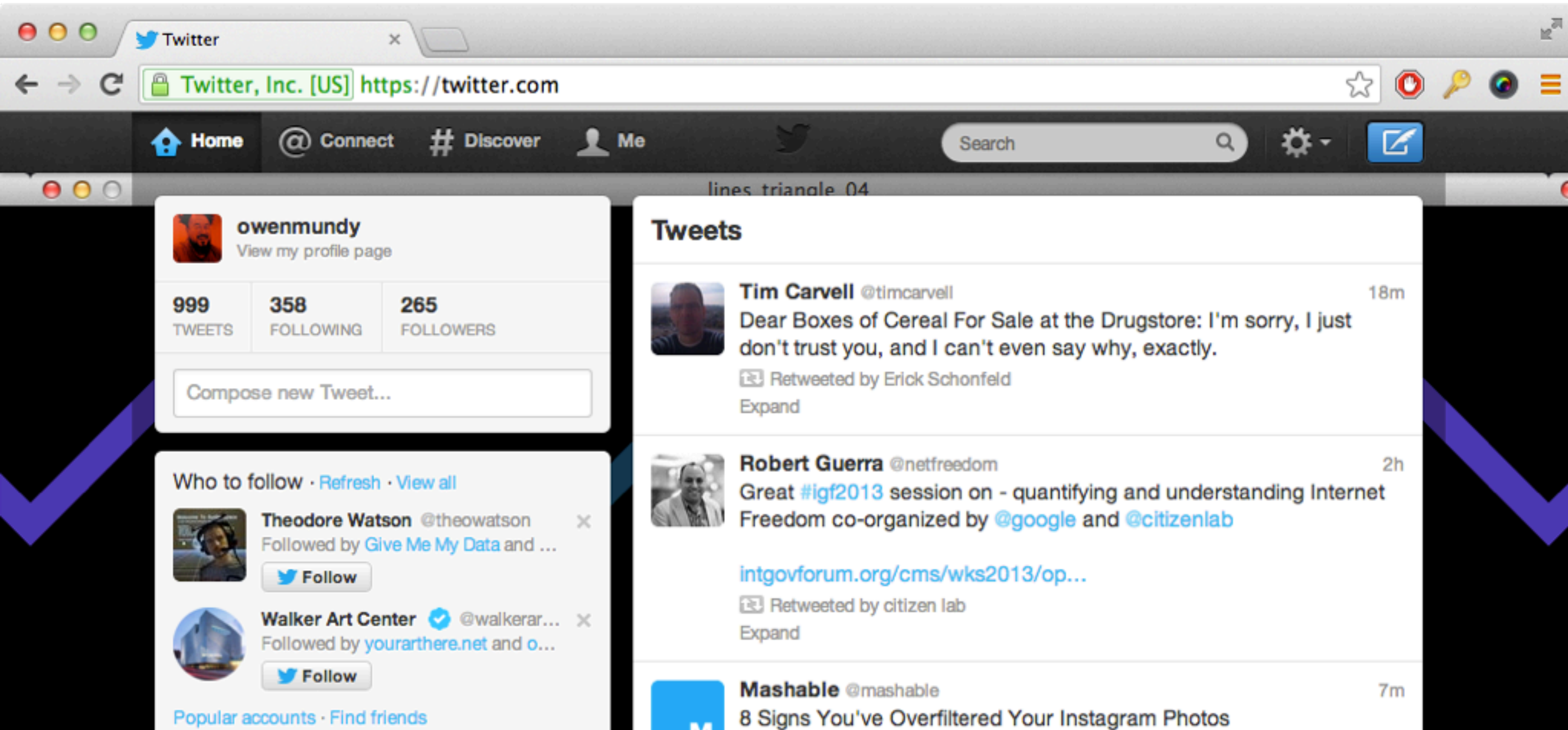
Overview

- Why are lists important?
- Arrays
- Loops

Everything is a list

As an interface designer / artist you will ultimately have to design around, produce markup for, and write code for some form of dynamic data containing more than one of something. Meaning, a list of things.

But when you look around the web, we realize everything is a list. A list of tweets...



owenmundy
View my profile page

999 TWEETS · 358 FOLLOWING · 265 FOLLOWERS

Compose new Tweet...

Who to follow · Refresh · View all

Theodore Watson @theowatson
Followed by Give Me My Data and ...
Follow

Walker Art Center @walkerar...
Followed by yourarthere.net and o...
Follow

Popular accounts · Find friends

Tweets

Tim Carvell @timcarvell 18m
Dear Boxes of Cereal For Sale at the Drugstore: I'm sorry, I just don't trust you, and I can't even say why, exactly.
Retweeted by Erick Schonfeld
Expand

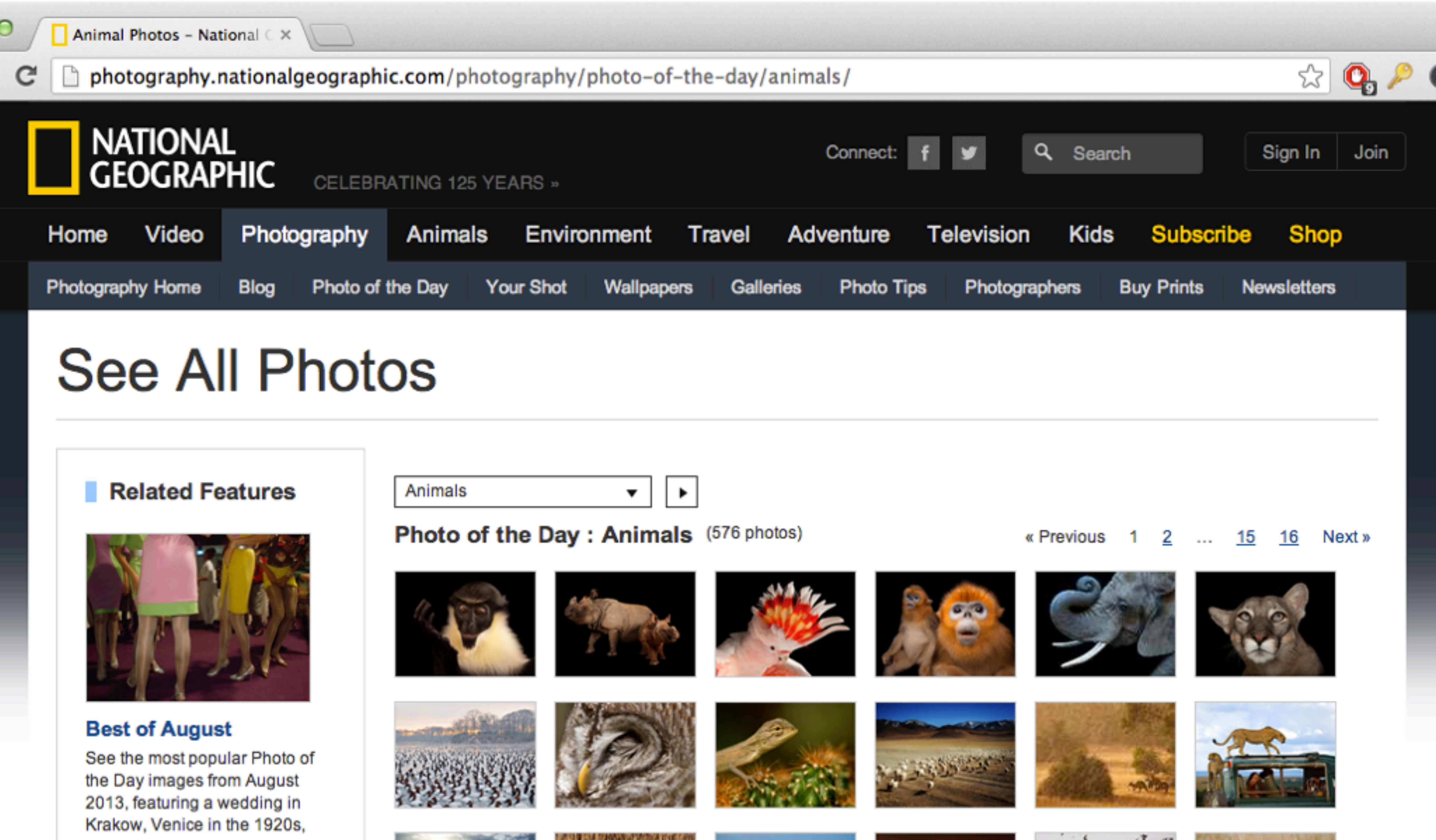
Robert Guerra @netfreedom 2h
Great #igf2013 session on - quantifying and understanding Internet Freedom co-organized by @google and @citizenlab

intgovforum.org/cms/wks2013/op...
Retweeted by citizen lab
Expand

Mashable @mashable 7m
8 Signs You've Overfiltered Your Instagram Photos

Everything is a list

Lists of photos...



See All Photos

Related Features

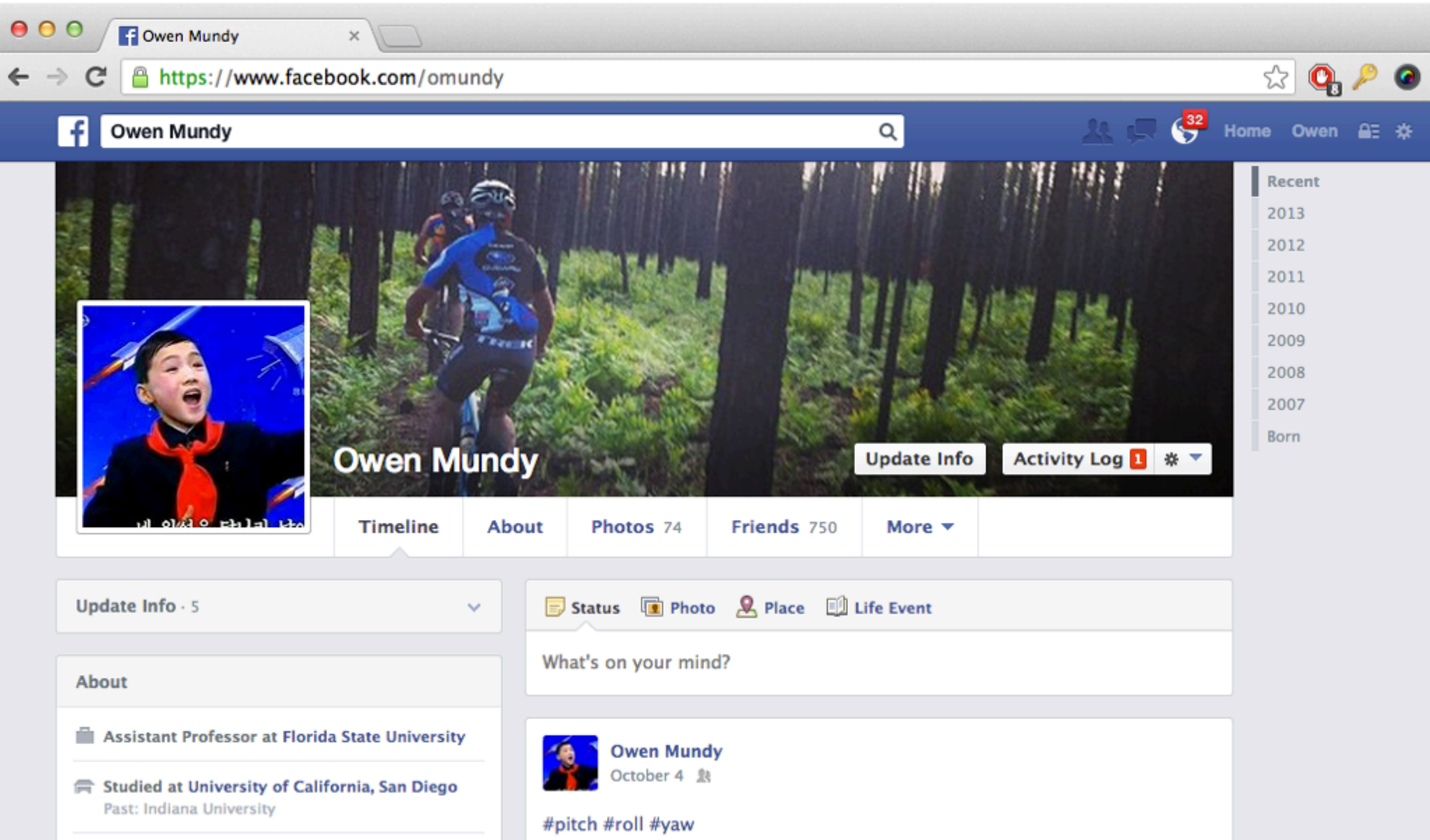
Best of August

See the most popular Photo of the Day images from August 2013, featuring a wedding in Krakow, Venice in the 1920s,

Photo of the Day : Animals (576 photos)

Everything is a list

A list of everything...



Everything is a list

To store and present any list you will need to use:

1. **Arrays** to store the list of data
2. **Loops** to address every item in the list

Arrays

An array is a list of items. It allows you to store multiple like pieces of data in a single variable.

Think about it like this: Arrays are like shopping lists. You could write each item on a separate piece of paper (or variable in our case)...

```
// four variables
$fruit1 = 'apples';
$fruit2 = 'pears';
$fruit3 = 'oranges';
$fruit4 = 'bananas';
```

...but it’s much more efficient to do it in one place.

```
// can be combined into one array
$fruit = array('apples','pears','oranges','bananas');
```

Arrays: Naming conventions

Arrays use the same naming conventions as variables (no spaces or special characters, all lowercase) with a couple syntactical exceptions.

```
// syntax
$name = array('each','item','separated','by','commas');
```

To make a new array you assign a list of values (separated by commas) to a variable using the array() keyword. Viola!

```
// can be combined into one array
$fruit = array('apples','pears','oranges','bananas');
```

Arrays: Accessing elements

Arrays are really storing a bunch of separate variables as a list, which can then be accessed through their index, or number in the list.

```php
// data from the array can be accessed like so
$fruit = array('apples','pears','oranges','bananas');
print $fruit[0];
```

This will print ‘apples.’

Note, all arrays start from zero and count up.

What will this print?

```php
print $fruit[2];
```

Arrays: Populating elements

You can also populate the array indexes by accessing the index or “key.”

```
$fruit[0] = 'peaches';
```

You can view all the data in an array with print_r()

```
// use print_r() to see all values in the array
print_r($fruit);
```

This will produce the following list of “keys” and the “values” they are paired with.

```
Array
(
    [0] => peaches
    [1] => pears
    [2] => oranges
    [3] => bananas
)
```

Hint: look at it in view source to see the formatted view.

Associative arrays

Associative arrays use real words (in the form of strings) as their keys. This example pairs food items (keys) with their location in the grocery store:

```
// associative array
$groceries = array('apples' => 'produce',
                   'bananas' => 'produce',
                   'yogurt' => 'dairy',
                   'granola' => 'cereal');
```

You can print individual values if you know their key:

```
// individual values can be accessed by referencing a key
print $groceries['granola'];
```

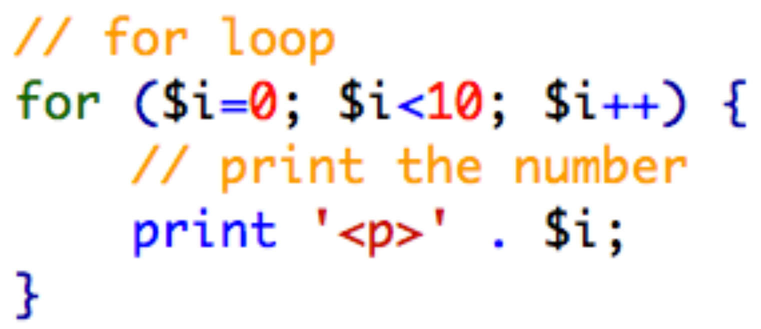
Or use print_r() to see all the keys=>value pairs

```
Array
(
    [apples] => produce
    [bananas] => produce
    [yogurt] => dairy
    [granola] => cereal
)
```

Loops

A loop is code that repeats a series of instructions for you. Every type of loop has:

1. A counter variable to keep track of the number of times a loop runs
2. The limit for the number of loops run
3. The code that increases the value of the counter for every loop



```
// for loop
for ($i=0; $i<10; $i++) {
    // print the number
    print '<p>' . $i;
}
```

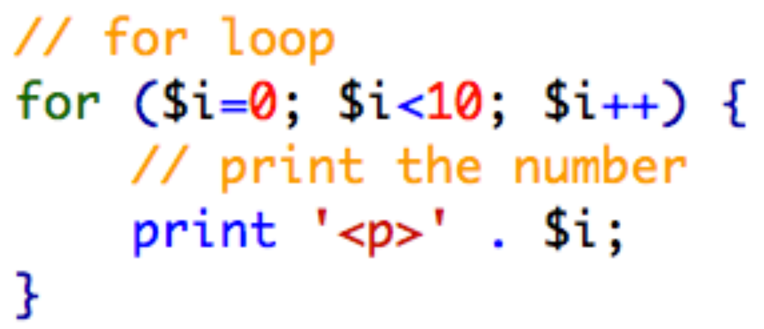
This code above says:
1. starting at **zero ($i)** ...
2. while **$i is less than 10** ...
3. increase **$i by one ($i++ is a shortcut for $i = $i + 1)**...
4. and perform the code in the curly braces each time.

Loops

Now compare these two types of loops.

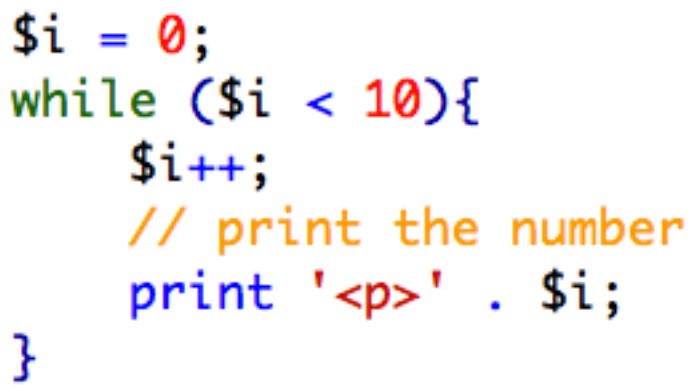
Find the counter, limit, and increment code in each:

For loop



```
// for loop
for ($i=0; $i<10; $i++) {
    // print the number
    print '<p>' . $i;
}
```

While loop



```
$i = 0;
while ($i < 10){
    $i++;
    // print the number
    print '<p>' . $i;
}
```

Counter variable: **$i**
Limit: **10**
Increasing code: **$i++**

Looping through an array

The real power from loops comes from being able to loop through data in arrays.

The following loop uses $i to reference the value of the $fruit array:

```php
$fruit = array('apples','pears','oranges','bananas');

for ($i=0; $i < count($fruit); $i++) {

    print '<p>' . $i .'. '. $fruit[$i];

}
```

So our HTML looks like this:

```
0. apples

1. pears

2. oranges

3. bananas
```