WELCOME
TO

Server Side Languages

About You

- ✳Your Name
- ✳What is your preference SS or CS? Why?
- ✳ASP, DBML, .NET, PERL, JAVA
- ✳How important is client-side in this class?
- ✳This course is parallel taught with database course. What does this mean?
- ✳How far are we going to get?
- ✳Next month (ASL).

About Me

Jay Bunner
Previous Classes (SSL, ASL, WPF)
Bachelors in Electrical Engineering Technology
(Purdue University)
Masters in Information & Communication Sciences
(Ball State University)

wbunner@fullsail.com

ext: 8253

Direct: (407) 842-1455

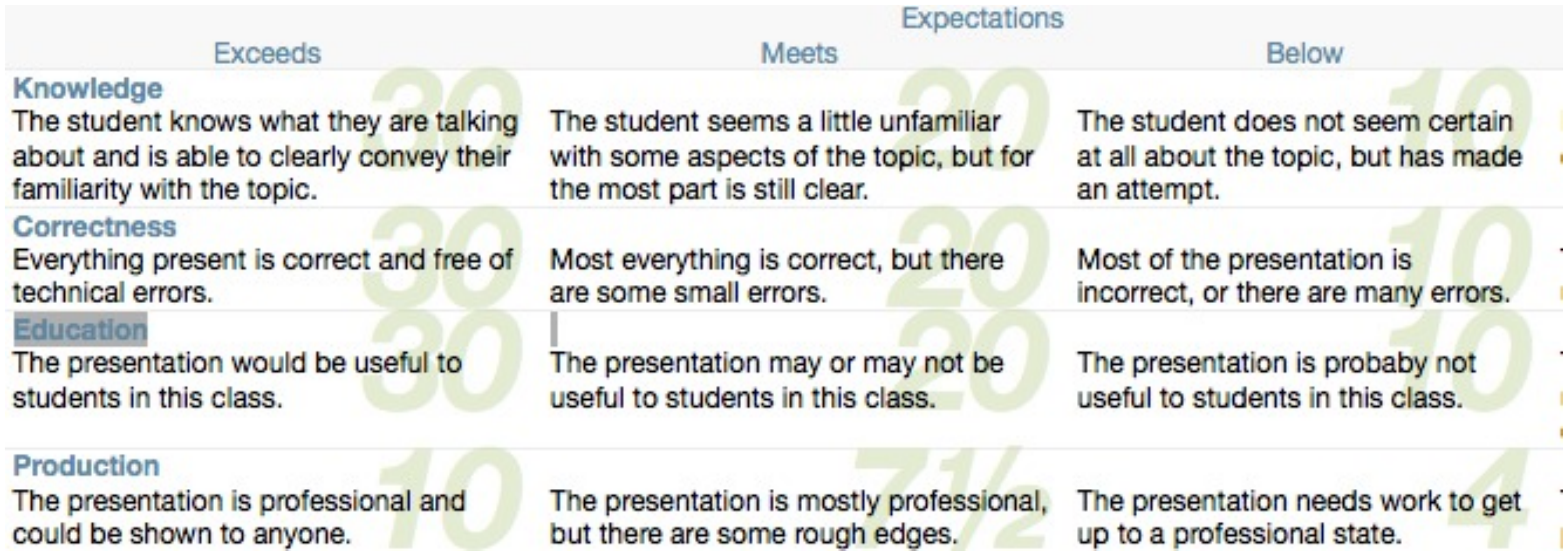
My office hours this month

VB6
ColdFusion
PHP
Classic ASP
.Net
AS2/3
FLEX

SSL Rubric

Labs	30
Videos	20
Practical Exam	30
Team Challenge	5
Student Lecture	5
GPS	10
TOTAL	100

Student Videos



		Expectations	
	Exceeds	Meets	Below
Knowledge	30 The student knows what they are talking about and is able to clearly convey their familiarity with the topic.	20 The student seems a little unfamiliar with some aspects of the topic, but for the most part is still clear.	10 The student does not seem certain at all about the topic, but has made an attempt.
Correctness	30 Everything present is correct and free of technical errors.	20 Most everything is correct, but there are some small errors.	10 Most of the presentation is incorrect, or there are many errors.
Education	30 The presentation would be useful to students in this class.	20 The presentation may or may not be useful to students in this class.	10 The presentation is probaby not useful to students in this class.
Production	10 The presentation is professional and could be shown to anyone.	7½ The presentation is mostly professional, but there are some rough edges.	4 The presentation needs work to get up to a professional state.

About SSL

- **LECTURE OR LAB NO FB, ESPN, MOVIES, ... YOU WILL BE ASKED TO LEAVE!**
- REMEMBER TO SILENCE PHONES
- DRINKS NOT ALLOWED!
- ALL DEADLINES ARE FINAL
  No late work will be accepted

•10 DAY COURSE
•YOU CAN MISS 8 HOURS WITHOUT DOCUMENTATION
•YOU CAN MISS A MAXIMUM OF 16 HOURS (WITH EXCUSABLE DOCUMENTATION)

•Either two 15-minute breaks or one 30-min during lecture
•In labs, no official break.
•If you arrive 15 minutes after the start of lecture or lab, you are 2 hours out
•If you will miss class, contact me ahead of time. Be proactive!

About SSL

- Turning in:
- DO NOT COMMIT YOUR VIDEOS!

  send the video link in a .txt file document instead

  to your YouTube or Vimeo link.

  - All labs/assignments will be turned in through Full Sail Online Course Page.
  - Videos: Send link to YouTube, Vimeo, etc.
  - Assignments: Submit through FSO under correct lab assignment. Put all source code files in zip file. Include your SQL dump (if applicable.) **Lastname_Firstname_Lab1.zip**

About SSL

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages.

It is usually used to provide interactive web sites, that interface to databases or other data stores.

Different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript.

The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

What knowledge is expected?

xhtml

- Know how to create static web page.
- Know how to validate a page

css

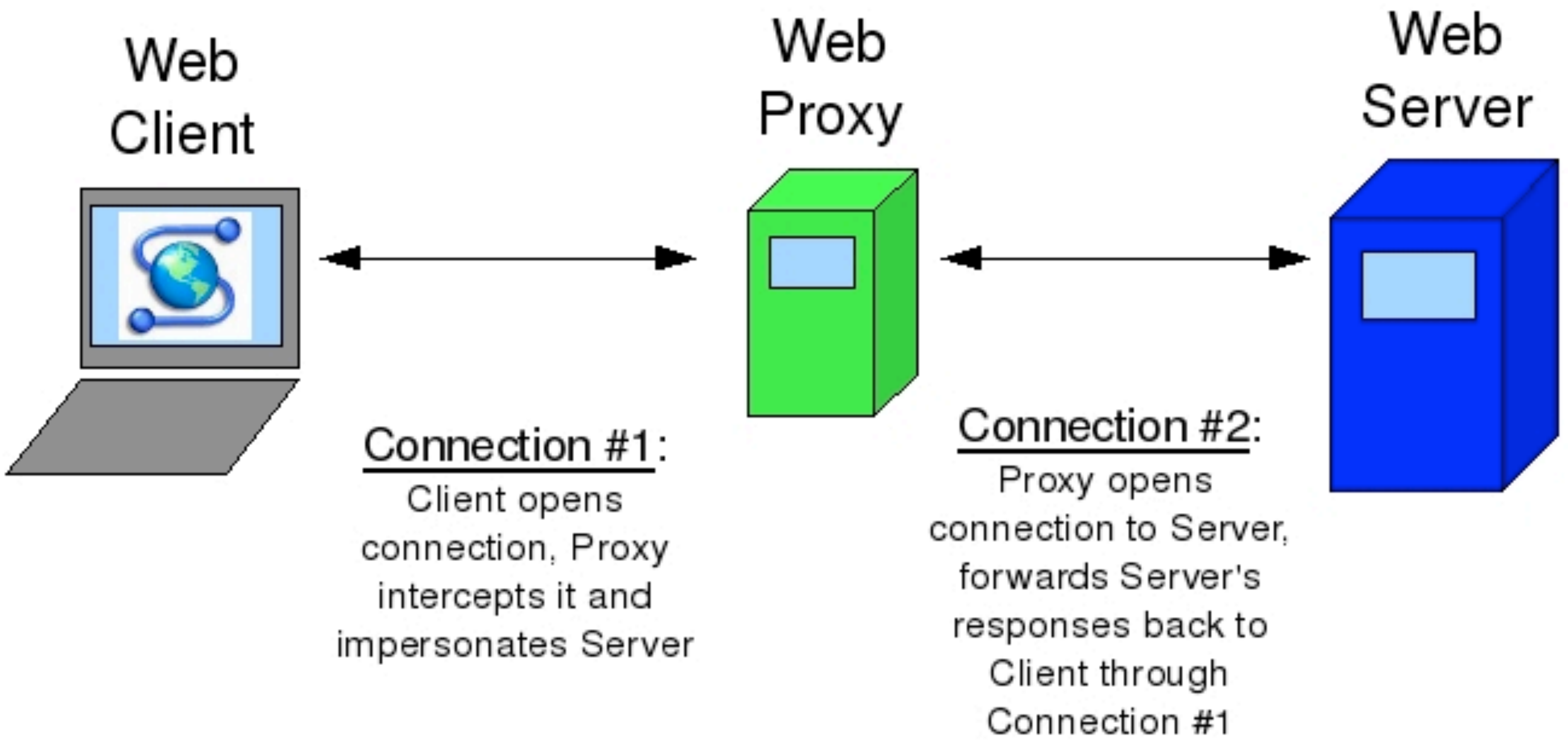
- Know how to apply style to a page using css

server / client relationship

oop concepts

ftp

Proxy Servers


Web Client
Web Proxy
Web Server
Connection #1: Client opens connection, Proxy intercepts it and impersonates Server
Connection #2: Proxy opens connection to Server, forwards Server's responses back to Client through Connection #1

Source: http://www.linuxjournal.com/magazine/paranoid-penguin-building-secure-squid-web-proxy-part-i

RESTful

http://www.ibm.com/developerworks/webservices/library/ws-restful/
Representational State Transfer (REST) has gained widespread acceptance across the Web as a simpler alternative to SOAP- and Web Services Description Language (WSDL)-based Web services.

Use HTTP methods explicitly.
Be stateless.
Expose directory structure-like URIs.
Transfer XML, JavaScript Object Notation (JSON), or both.

Use HTTP methods explicitly

One of the key characteristics of a RESTful Web service is the explicit use of HTTP methods in a way that follows the protocol as defined by RFC 2616.

REST asks developers to use HTTP methods explicitly and in a way that's consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.

According to this mapping:
To create a resource on the server, use POST.
To retrieve a resource, use GET.
To change the state of a resource or to update it, use PUT.
To remove or delete a resource, use DELETE.

Use HTTP methods explicitly

One of the key characteristics of a RESTful Web service is the explicit use of HTTP methods in a way that follows the protocol as defined by RFC 2616.

REST asks developers to use HTTP methods explicitly and in a way that's consistent with the protocol definition. This basic REST design principle establishes a one-to-one mapping between create, read, update, and delete (CRUD) operations and HTTP methods.

According to this mapping:
To create a resource on the server, use POST.
To retrieve a resource, use GET.
To change the state of a resource or to update it, use PUT.
To remove or delete a resource, use DELETE.

Listing 1. Before GET /adduser?name=Robert HTTP/1.1

Listing 2. After

```
POST /users HTTP/1.1
Host: myserver
Content-Type: application/xml
<?xml version="1.0"?>
<user>
  <name>Robert</name>
</user>
```

Expose directory structure-like URIs

http://www.myservice.org/discussion/2008/12/10/{topic}

http://www.myservice.org/discussion/{year}/{day}/{month}/{topic}

Some additional guidelines to make note of while thinking about URI structure for a RESTful Web service are:
•Hide the server-side scripting technology file extensions (.jsp, .php, .asp), if any, so you can port to something else without changing the URIs.
•Keep everything lowercase.
•Substitute spaces with hyphens or underscores (one or the other).
•Avoid query strings as much as you can.
•Instead of using the 404 Not Found code if the request URI is for a partial path, always provide a default page or resource as a response.

URIs should also be static so that when the resource changes or the implementation of the service changes, the link stays the same. This allows bookmarking. It's also important that the relationship between resources that's encoded in the URIs remains independent of the way the relationships are represented where they are stored.

What is server side scripting?

Server-side scripting is a web server technology in which a user's request is fulfilled by running a script directly on the web server to generate dynamic HTML pages. It is usually used to provide interactive web sites, that interface to databases or other data stores.

This is different from client-side scripting where scripts are run by the viewing web browser, usually in JavaScript. The primary advantage to server-side scripting is the ability to highly customize the response based on the user's requirements, access rights, or queries into data stores.

Some server side scripting languages

Perl

Ruby

Python

ASP

ColdFusion

PHP

Perl

In computer programming, Perl is a high-level, general-purpose, interpreted, dynamic programming language. Perl was originally developed by Larry Wall, a linguist working as a systems administrator for NASA, in 1987, as a general purpose Unix scripting language to make report processing easier. Since then, it has undergone many changes and revisions and became widely popular among programmers. Larry Wall continues to oversee development of the core language, and its newest version, Perl 6.

Ruby

Ruby is a dynamic, reflective, general purpose object-oriented programming language that combines syntax inspired by Perl with Smalltalk-like features. Ruby originated in Japan during the mid-1990s and was initially developed and designed by Yukihiro "Matz" Matsumoto. Ruby supports multiple programming paradigms, including functional, object oriented, imperative and reflection. It also has a dynamic type system and automatic memory management; it is therefore similar in varying respects to Python, Perl, Lisp, Dylan, and CLU.The standard 1.8.6 (stable) implementation is written in C, as a single-pass interpreted language.There is currently no specification of the Ruby language, so the original implementation is considered to be the de facto reference.

Python

Python is a general-purpose, high-level programming language. Its design philosophy emphasizes programmer productivity and code readability. Python's core syntax and semantics are minimalistic, while the standard library is large and comprehensive. Its use of whitespace as block delimiters is unusual among popular programming languages. Python supports multiple programming paradigms (primarily object oriented, imperative, and functional) and features a fully dynamic type system and automatic memory management, similar to Perl, Ruby, Scheme, and Tcl. Like other dynamic languages, Python is often used as a scripting language. Python was first released by Guido van Rossum in 1991. The language has an open, community-based development model managed by the non-profit Python Software Foundation, which also maintains the de facto standard definition of the language in CPython, the reference implementation.

ColdFusion

ColdFusion Markup Language, more commonly known as CFML, is the scripting language used by Adobe ColdFusion, BlueDragon and Railo, as well as other CFML server CFML is a tag-based versatile scripting language that is self-contained and easy-to-learn. It promotes rapid web application development. ColdFusion code is short and concise. It is easy to understand exactly what the code does, even without programming knowledge. CFML enhances standard HTML files with database commands, conditional operators, high-level formatting functions, and other elements to rapidly produce easy-to-maintain web applications. The pages in a ColdFusion application include the server-side CFML tags in addition to HTML (XHTML) tags. When a browser requests a page in a ColdFusion application, it is automatically pre-processed by the ColdFusion Application Server.

PHP

PHP is a widely-used general-purpose scripting language that is especially suited for Web development and can be embedded into HTML.
PHP is a recursive acronym that stands for:
PHP: Hypertext Preprocessor

PHP originally stood for Personal Home Page. It began in 1994 as a set of Common Gateway Interface binaries written in the C programming language by the Danish/Greenlandic programmer Rasmus Lerdorf. Lerdorf initially created these Personal Home Page Tools to replace a small set of Perl scripts he had been using to maintain his personal homepage. On May 22, 2000, PHP 4, powered by the Zend Engine 1.0, was released. On July 13, 2004, PHP 5 was released, powered by the new Zend Engine II. In 2008, PHP 5 became the only stable version under development.

Static Vs. Dynamic

Static: A static Web page is a Web page that always comprises the same information in response to all download requests from all users. It displays the same information for all users, from all contexts, providing the classical hypertext, where navigation is performed through "static" documents.

Disadvantages

Difficult to maintain when a site gets large.

Difficult to keep consistent and up to date.

Offers little visitor personalization (all would have to be client side).

Dynamic: Using server-side scripting to change the supplied page source between pages, adjusting the sequence or reload of the web pages or web content supplied to the browser. Server responses may be determined by such conditions as data in a posted HTML form, parameters in the URL, the type of browser being used, the passage of time, or a database or server state.

Let’s get started

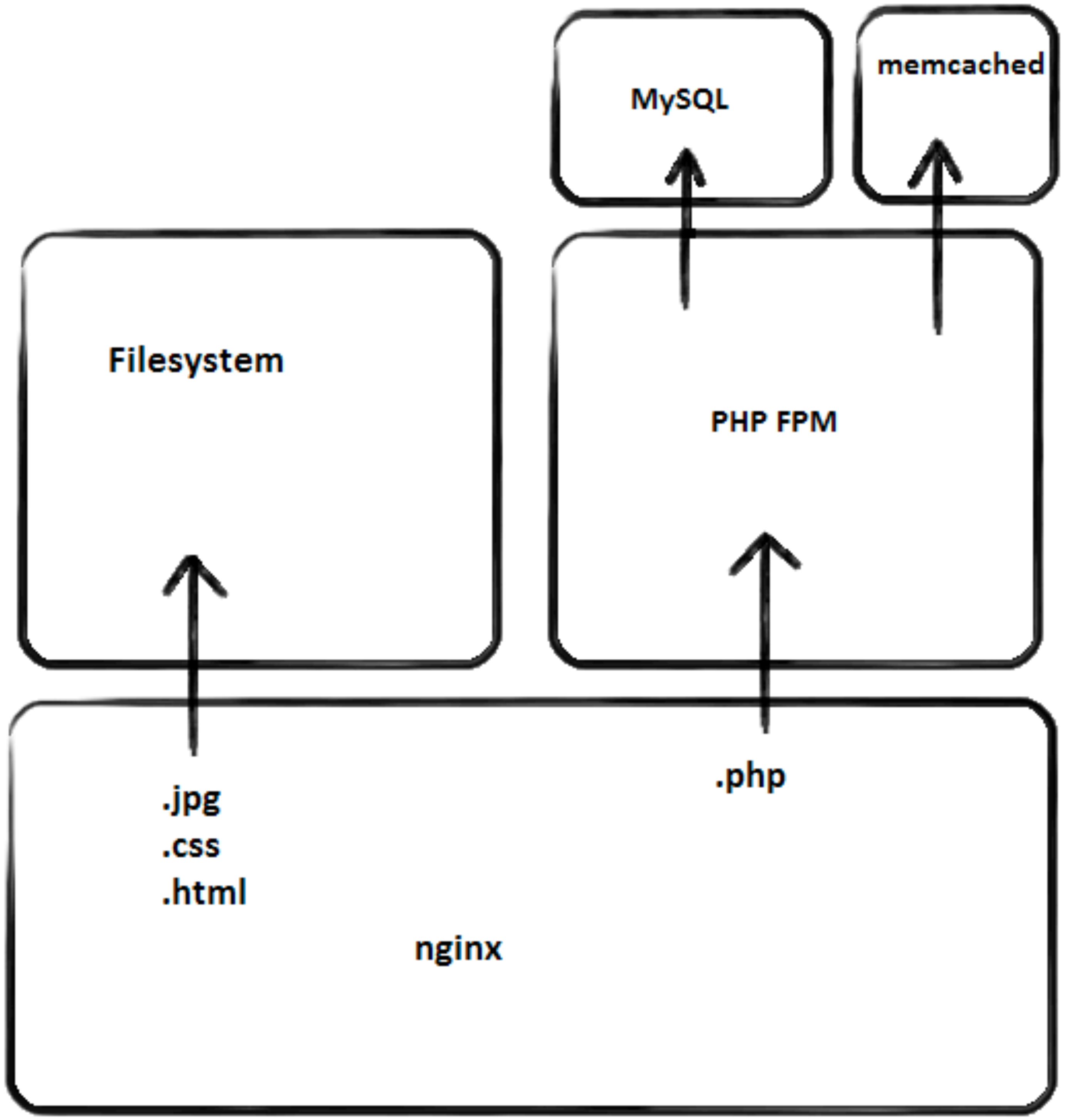
MAMP
RAILO
FIREFOX
FIREBUG
CODE EDITOR
GIT REPO
KEYS SETUP

How do PHP and Web Server work together?

Web server (Apache, nginx, etc.) accept URL request, pass off to PHP interpreter for file extensions ending in *.php*
(see httpd.conf for Apache configuration, addtype application)

PHP “engine” interprets and generates response
PHP modules - Applications/MAMP/Library/Modules/*.so
php.ini - specifies global configuration options. Some options can be overridden locally in scripts at runtime.

How PHP Integrates with Web Servers



Source: http://www.alwaysgetbetter.com/blog/2012/01/30/setting-wordpress-nginx-fastcgi/

Let’s get started

Download MAMP - http://www.mamp.info

Test MAMP - http://localhost:8888
Create an SSL directory, then Day 1 subdirectory
Point MAMP web root to “./SSL/Day 1” directory you just created
Create file called phptest.php under Day 1
Insert the following code in phptest.php file

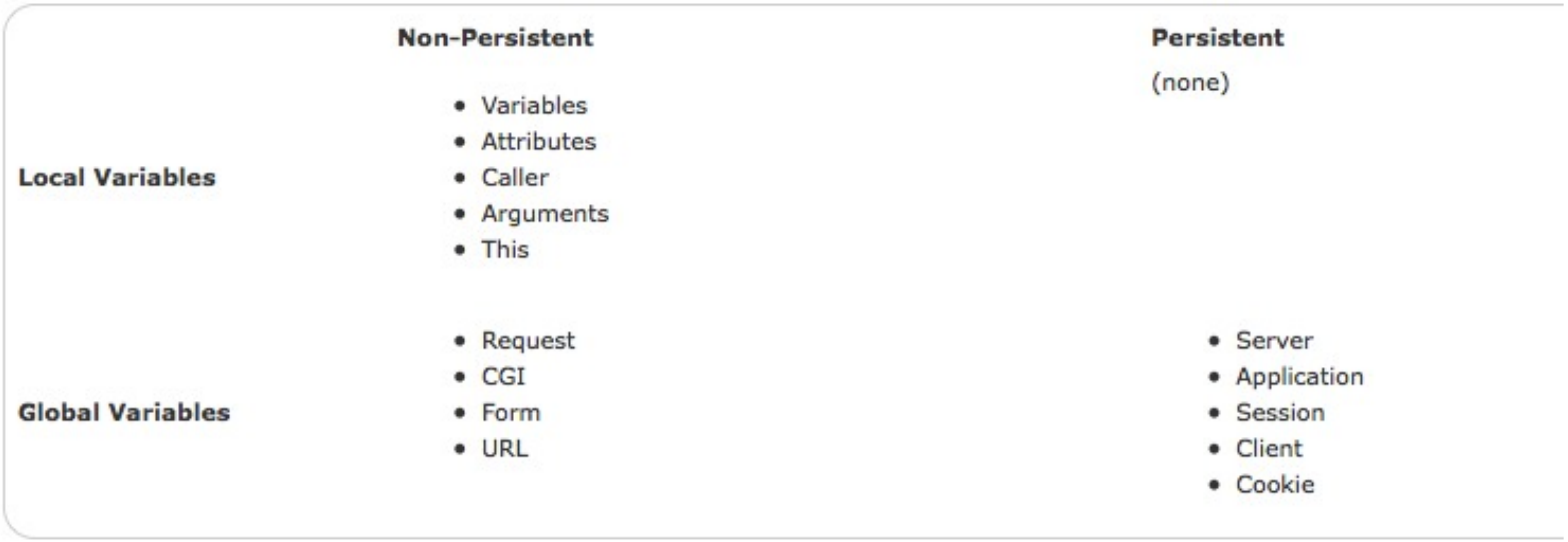
```
<?php phpinfo(); ?>
```

Test to make sure file executes - http://localhost:8888/phptest.php

PHP Syntax

Files must end in *.php*
Go into PHP Mode with: <?php
Get out of PHP Mode with: ?>
Every time you give a command, end it with a semicolon ;
Comment: // Single line
Comment: /* multiline */

$name = "Bob";
$age = 29;
echo "I am $name and I am $age years old."; // sends information to the browser
I have <?=$foo?> foo. //Shortcut Syntax



	Non-Persistent	Persistent
Local Variables	• Variables • Attributes • Caller • Arguments • This	(none)
Global Variables	• Request • CGI • Form • URL	• Server • Application • Session • Client • Cookie

PHP Variables

Letters, words, blocks of text (String)
True/False values (Boolean)
Numbers (Integer)
Numbers with decimals (Floating Point)
Array
Object
Null

All variables begin with a $
You assign values to variables with the =
$name = ‘Bob’;
Can begin with letters or the _underscore
CANNOT begin with a number
Variables ARE case sensitive

PHP Variables

```
$myvar=Array();
$yourvar=1234;
$hervar=”Joe”;
$hisvar=true;
```

PHP Arrays

```
$myAr = array();  or array(“a”,true);
$myAssocArray =
array(“name”=>”orcun”);
count($myAr);
array_push($myAr,”new val”);
```

Arrays

http://us1.php.net/manual/en/ref.array.php

http://us1.php.net/manual/en/language.types.array.php

An array can be created using the array() language construct. It takes any number of comma-separated key => value pairs as arguments.

```
array(
    key  => value,
    key2 => value2,
    key3 => value3,
    ...
)
$array = array(
    "foo" => "bar",
    "bar" => "foo",
);

$array = array("foo", "bar", "hallo", "world");
var_dump($array);
array(4) {
  [0]=>
  string(3) "foo"
  [1]=>
  string(3) "bar"
  [2]=>
  string(5) "hallo"
  [3]=>
  string(5) "world"
}
```

PHP Quotes

echo “I am $name and I am $age.”;
//Double quotes considered complex variables will be replaced
echo ‘I am $name and I am $age.’;
//Single quotes considered literal variables will not be replaced
echo ‘I am ‘ . $name .‘ and I am ‘ . $age;
//Concatenate strings and variables using dot

Escape Sequences

echo “<p class=\“name\”>$name</p>”; //backslash esc double-quote
echo “\t<p>Tabs work too!</p>”; //tab in source code
echo “<p>Newlines are Pretty</p>\n”; //new line in source code

PHP Reference

php.net

Sample PHP Pages

Exercise 1:
Save the file as index.php in your Day 1 directory.
Set a variable.
Echo out the variable along with string text.
Browse to http://localhost:8888 and see if your page executes.
index.php is the *default page* that is shown by the web server if no page is specified in the path.

Exercise 2:
Take your index.php page, and include the phptest.php page AFTER your exercise 1 items.
Use the include() function in PHP
See php.net documentation for details on how to use include() function.

PHP Functions

http://us1.php.net/manual/en/functions.arguments.php

User-defined functions take on the form:

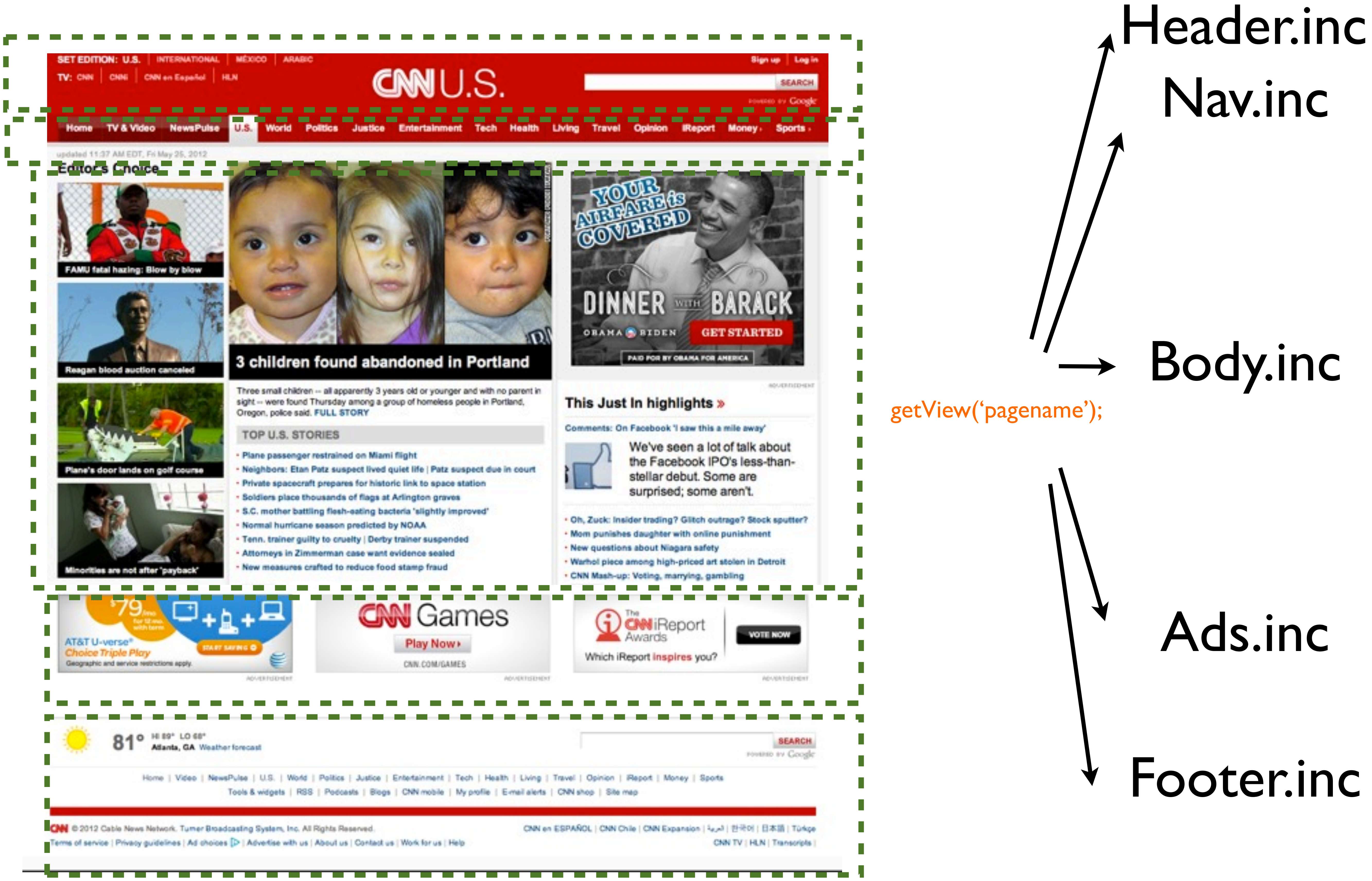
```
<?php
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
?>
```

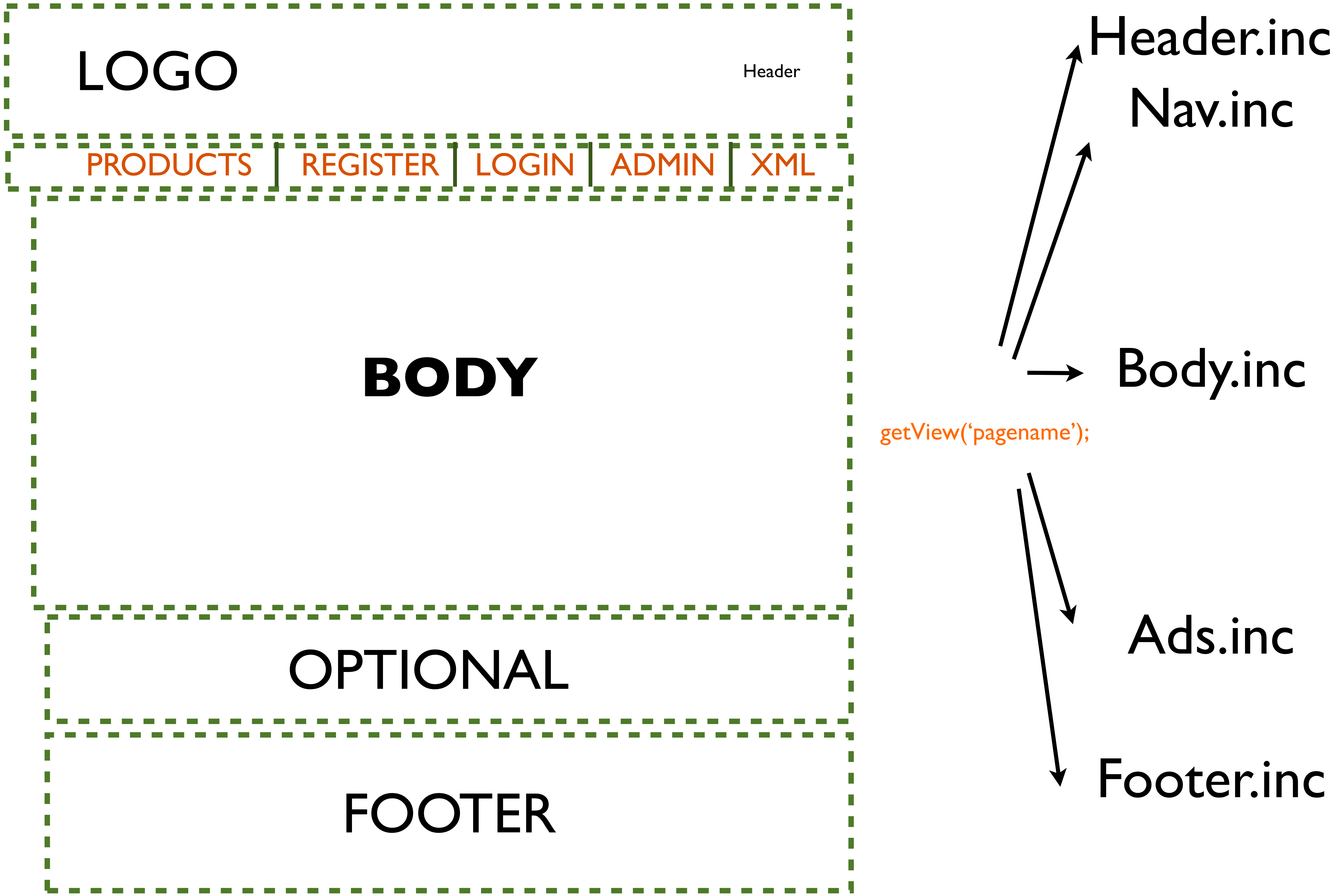
Returning Values

```
<?php
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

Create Simple Views

Take a static page, divide up into individual components.




LOGO
Header
PRODUCTS | REGISTER | LOGIN | ADMIN | XML
BODY
OPTIONAL
FOOTER
getView('pagename');
Header.inc
Nav.inc
Body.inc
Ads.inc
Footer.inc

Today's Lab - LAB 1

In this lab you will divide static HTML files into include files that encapsulate common page and site elements, such as headers, footers, etc. and then add some dynamic elements to the page.
1a: Create separate files for each of the page sections (Header, body, footer) and display the completed page as the root of your website.
1b: Add dynamic code in the header that displays the current date and time the page was loaded in the format "Monday, August 1st, 2013 10:23 a.m."
1c: In the Body section of your layout, generate 3 sets of Power Ball numbers that change each time the page is refreshed.
Use a Function to generate an array with each set of Power Ball numbers

SUBMIT: Source code zipped and uploaded on FSO Lab 1; Link to screencast video explaining code uploaded on FSO Video 1.

Lab #1

PHP

```
<? require_once("mypage.php"); ?>
<? include("mpage.php");?>
<? header(location:mypage.php); ?>
```

Your Output should look something like this:

```
(Header) (Date/Time)
PowerBall Set 1: 5 12 15 27 38 7
PowerBall Set 2: 6 11 16 26 28 2
PowerBall Set 3: 7 10 17 25 18 6
(Footer)
```