CS 455
Database Management Systems

Department of Mathematics
and Computer Science

Lecture 4
Dynamic Web Programming
with PHP

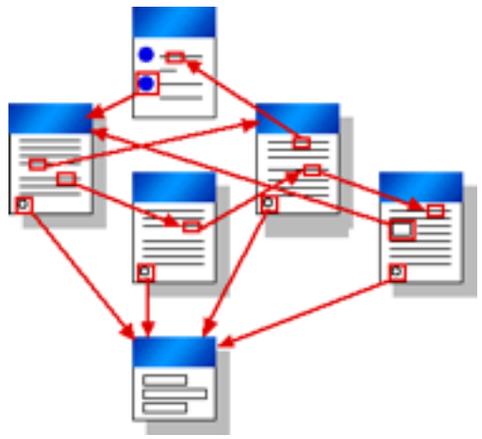
Disclaimer: Materials adapted for use from various sources

History of the Web

- World Wide Web invented by Tim Berners-Lee at CERN (1990)
  - Whaa? CERN is a high-energy physics lab!
  - Goal of project was to build a system that lets scientists browse and exchange *hypertext* documents

- Hypertext Documents? They can link to one another.
  - Hypertext was nothing new... (1940's idea)
  - Computers as delivery system was

Looks kinda like...



Tim Berners-Lee

History (Cont.)

- Given the green light by CERN, Berners-Lee wrote the first web browser and web server:
  - Hypertext browser called WorldWideWeb (client)
  - Hypertext server called CERN httpd (server)

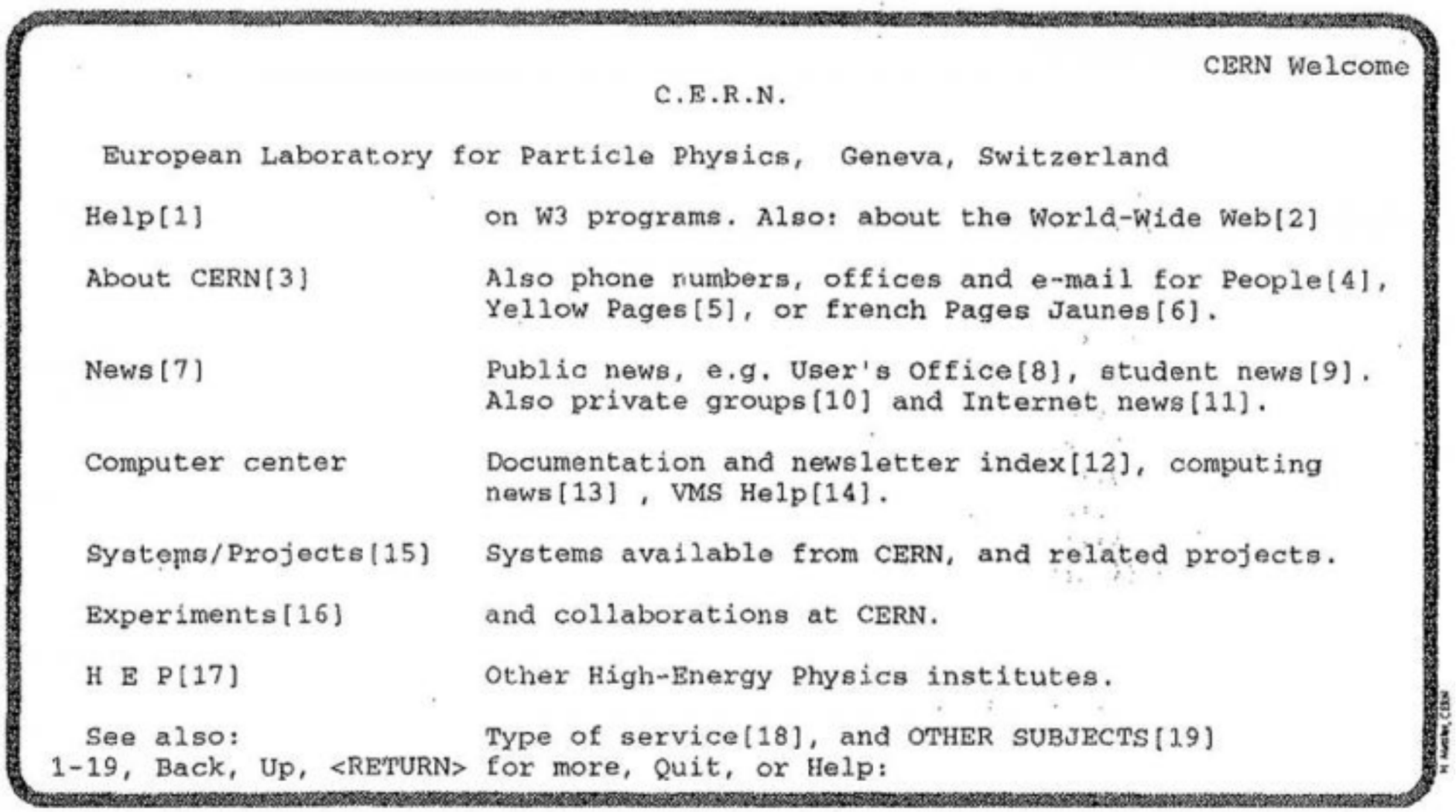
- After some consideration, he invents HTML, which becomes publishing language
  - Named after HTTP
  - \LaTeX was even considered

- Forms W3C to oversee web standards

CERN httpd: World's First Web Server
Source: wikipedia

WorldWideWeb Browser

- The first browsers were line-based
- Re-creation of the very first website
  - http://line-mode.cern.ch/www/hypertext/WWW/TheProject.html



```
                                                                CERN Welcome
                                C.E.R.N.

      European Laboratory for Particle Physics,  Geneva, Switzerland

   Help[1]                    on W3 programs. Also: about the World-Wide Web[2]

   About CERN[3]              Also phone numbers, offices and e-mail for People[4],
                              Yellow Pages[5], or french Pages Jaunes[6].

   News[7]                    Public news, e.g. User's Office[8], student news[9].
                              Also private groups[10] and Internet news[11].

   Computer center            Documentation and newsletter index[12], computing
                              news[13] , VMS Help[14].

   Systems/Projects[15]       Systems available from CERN, and related projects.

   Experiments[16]            and collaborations at CERN.

   H E P[17]                  Other High-Energy Physics institutes.

   See also:                  Type of service[18], and OTHER SUBJECTS[19]
1-19, Back, Up, <RETURN>      for more, Quit, or Help:
```

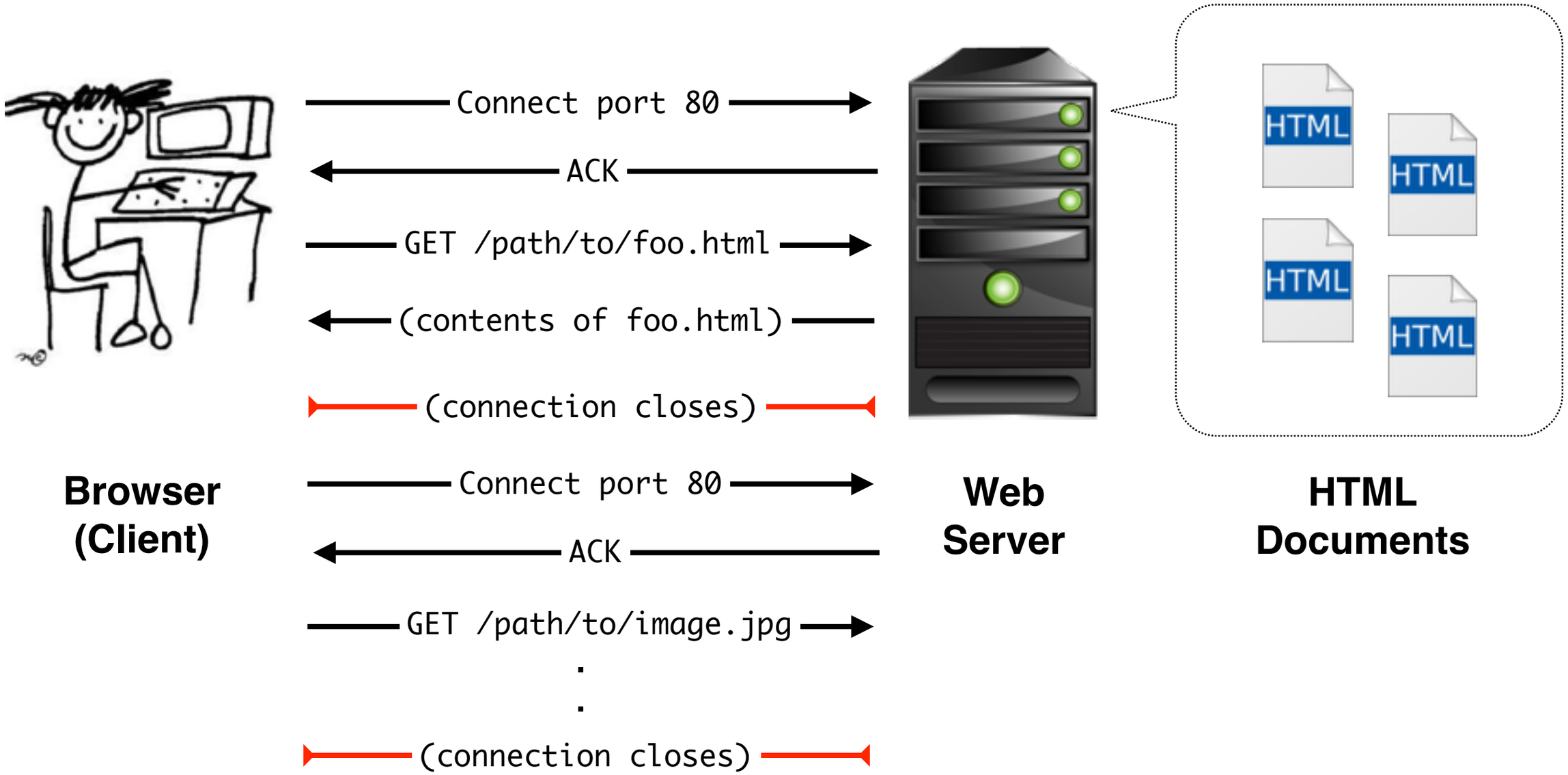
http://home.web.cern.ch/about/updates/2013/06/recreating-one-webs-first-browsers

Web Servers

- Web servers
  - Program that runs on (set of) high-performance machines
  - Deliver HTML pages to clients (browsers) around the globe
    - Protocol known as HTTP or HTTPS (Secure Socket Layer)

- Examples:
  - Apache Web Server
  - Microsoft Internet Information Services (IIS)
  - Apache and IIS combine for roughly 70% of market share

How the Web Works: HTTP/1.1

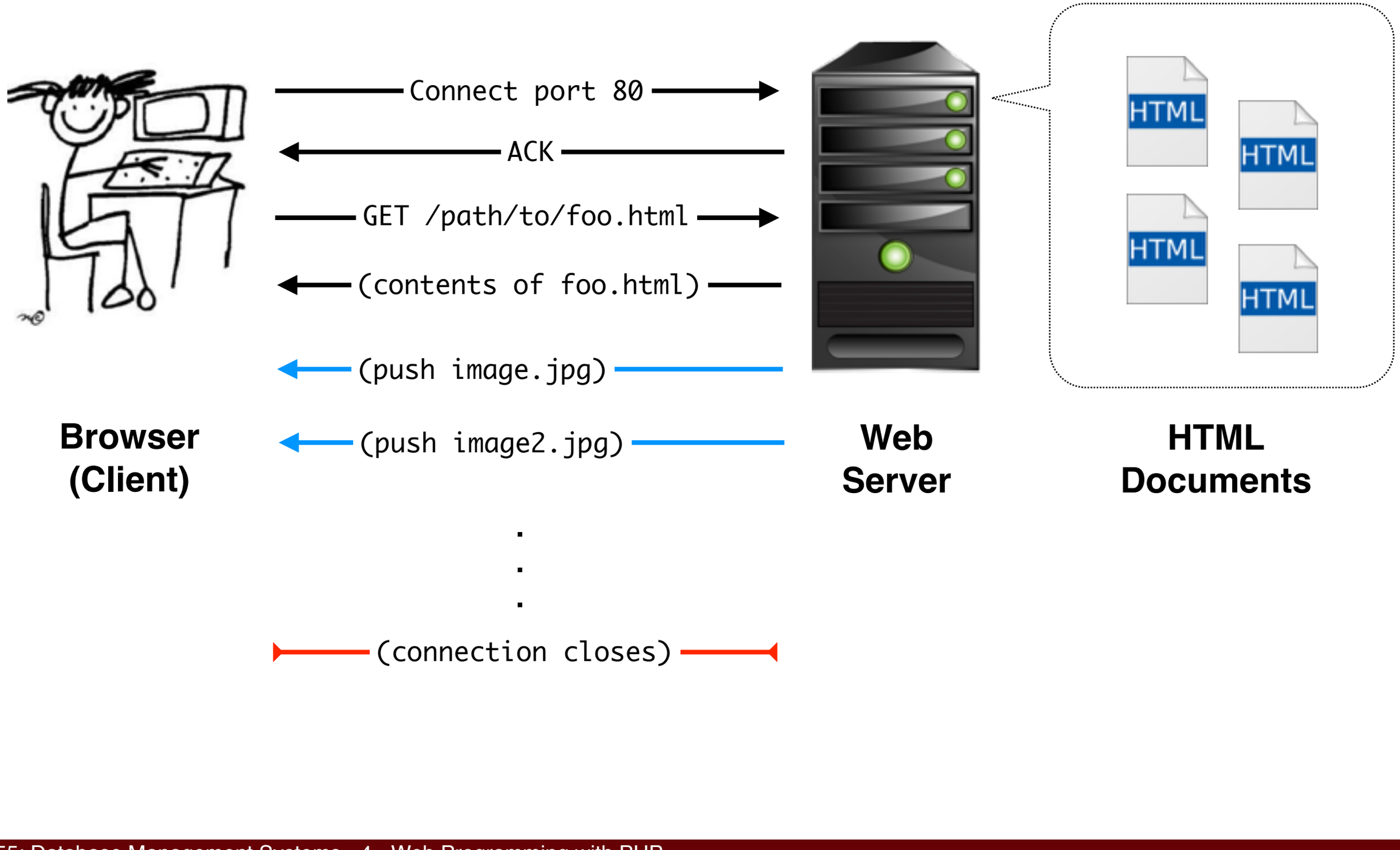


Proof of Inefficiency

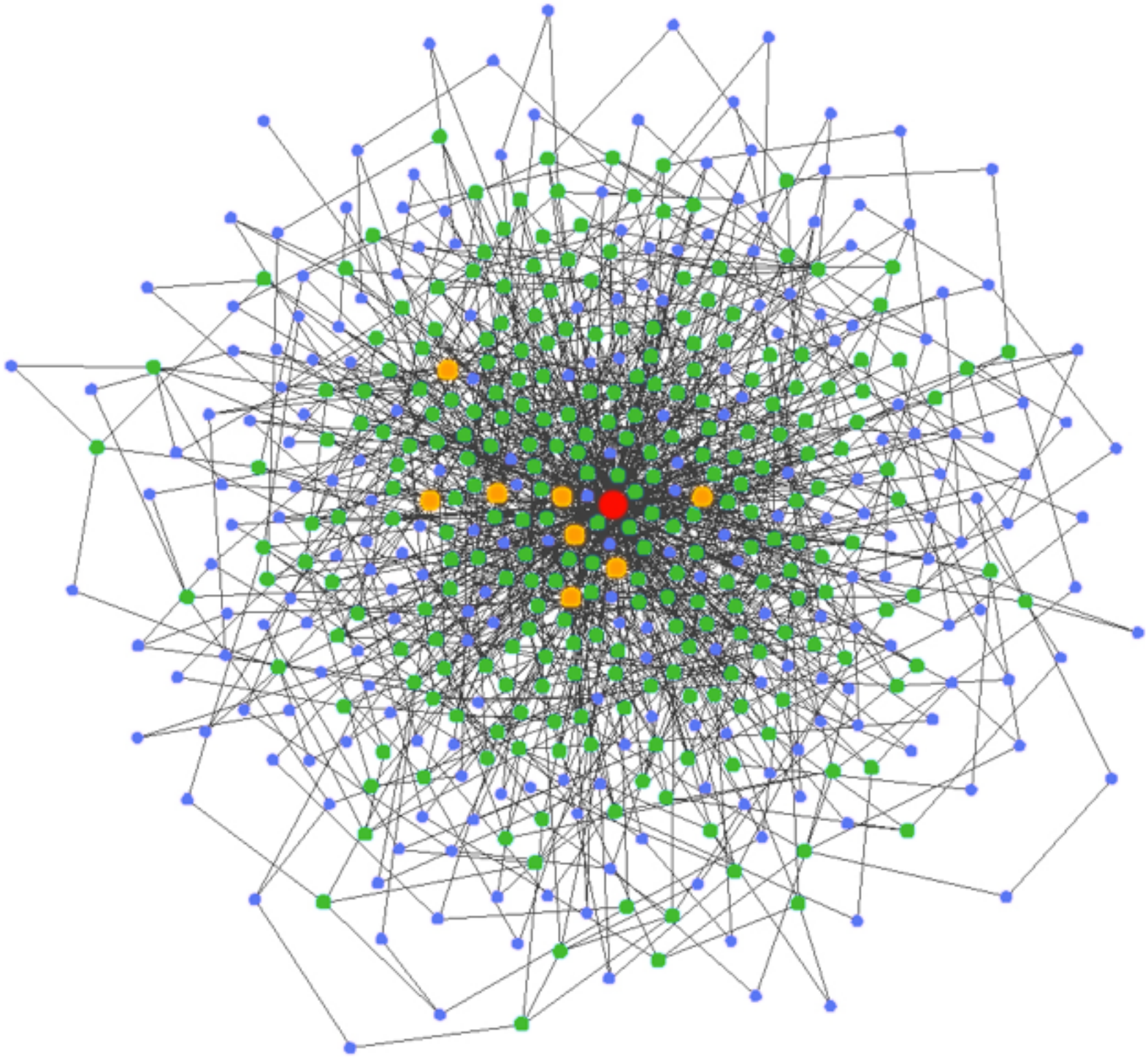
- Here are the apache access logs to fulfill the request for *one* page!

```
98.237.163.225 - - [22/Feb/2015:22:41:17 -0800] "GET /~dchiu/CS161/ HTTP/1.1" 200 9762
"-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_7_5) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/40.0.2214.115 Safari/537.36"

98.237.163.225 - - [22/Feb/2015:22:41:17 -0800] "GET /~dchiu/CS161/hwk3/ HTTP/1.1" 200
7588 "http://cs.pugetsound.edu/~dchiu/CS161/" "Mozilla/5.0 (Macintosh; Intel Mac OS X
10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115 Safari/537.36"

98.237.163.225 - - [22/Feb/2015:22:41:17 -0800] "GET /~dchiu/includes/hwk.css HTTP/1.1"
304 - "http://cs.pugetsound.edu/~dchiu/CS161/hwk3/" "Mozilla/5.0 (Macintosh; Intel Mac
OS X 10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115 Safari/537.36"

98.237.163.225 - - [22/Feb/2015:22:41:17 -0800] "GET /~dchiu/includes/highlight.css
HTTP/1.1" 304 - "http://cs.pugetsound.edu/~dchiu/CS161/hwk3/" "Mozilla/5.0 (Macintosh;
Intel Mac OS X 10_7_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/40.0.2214.115
Safari/537.36"
```

How the Web Works: HTTP/2



The Web Is a Collection of Websites



Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion

HTML Documents

- Hypertext Markup Language (HTML)
  - Made public in 1991 in a document Berners-Lee called "HTML Tags"
- A "markup" language?
  - Uses `<tags>` (also called elements) to describe the *structure* of a document
  - 18 original tags; 11 survives to this day
- HTML5 (current version) now has 128 tags

HTML Tag Structure

- Two types of tags:
  - Non-Empty Tag: <tag> ... stuff ... </tag>

```
<p>This is a new paragraph.</p>
```

```
<ul>
   <li>item 1</li>
   <li>item 2</li>
</ul>
```

  - Empty Tag: <tag/>

```
<br/>
```

```
<img src="path/to/image.jpg"/>
```

HTML Attributes

- Attributes are associated with tags
  - Attributes have the same meaning as they do in relational tables
  - Has a `attributeName="value"` syntax
  - A tag may have multiple attributes

- For instance,
  - This tag lets you include an image

```
<img src="david.jpg" alt="Pic of David" width="100px" height="300px"/>
```

HTML Documents (Cont.)

- An HTML document...
  - Typically has an `.html` or `.htm` file extension
    - or `.php` if it's a PHP page
  - Has two sections: header and body
    - *Header* holds metadata (data *about* the content of the page)
    - *Body* holds actual content
  - Comments are multi-line: `<!-- this is a comment -->`

```
<!DOCTYPE html>
<html>
   <head>
      <!-- header content -->
   </head>
   <body>
      <!-- body content -->
   </body>
</html>
```

Important <head> Elements

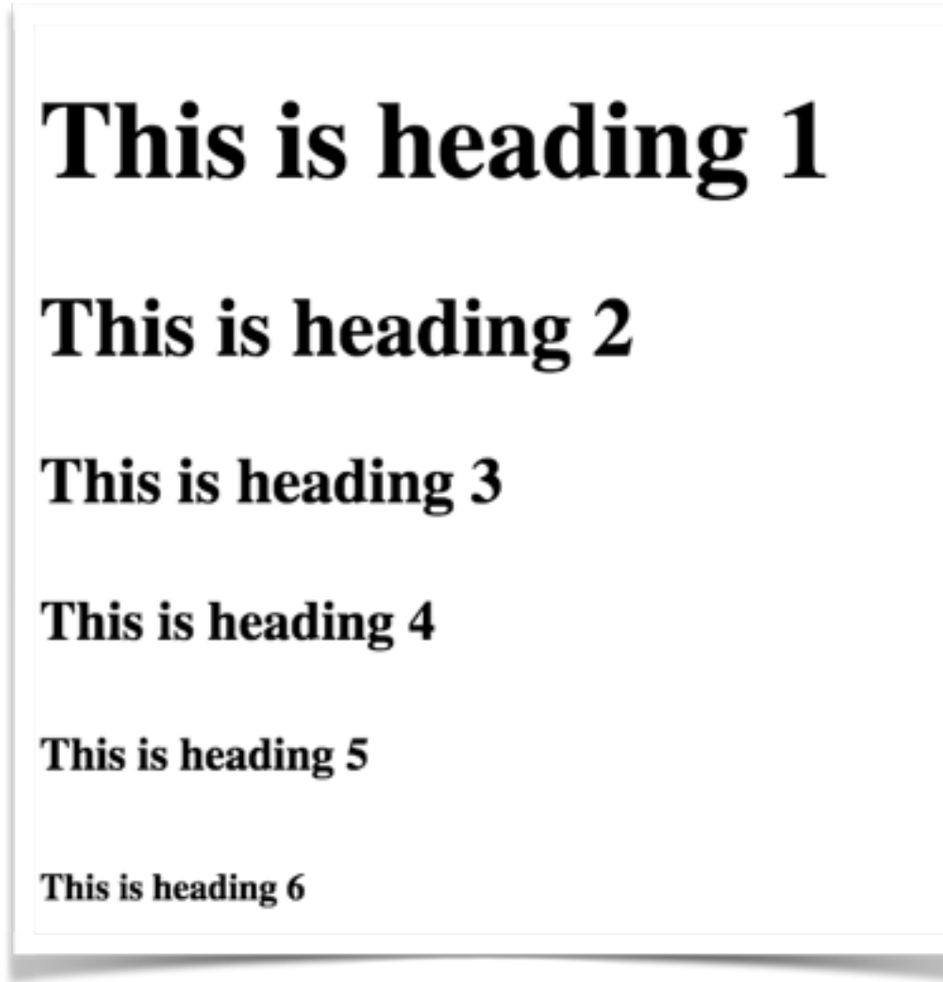
- Header elements
  - `<title>...</title>` Title of page
  - `<link ... />` Links to include stylesheets (css) and JavaScript files
  - `<meta ... />` Used by search engines

```
<!DOCTYPE html>
<html>
   <head>
      <title>Title of Document</title>
      <link rel="stylesheet" type="text/css" href="path/to/file.css" />
      <meta name="author" content="David Chiu" />
   </head>

   <!-- body omitted -->

</html>
```

Important <body> Elements

- What goes inside the `<body>` tag?
- Heading elements
  - `<h1>...</h1>, <h2>...</h2>, <h3>...</h3>`
  - `<h4>...</h4>, <h5>...</h5>, <h6>...</h6>`

```
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<h3>This is heading 3</h3>
<h4>This is heading 4</h4>
<h5>This is heading 5</h5>
<h6>This is heading 6</h6>
```



Important <body> Elements (Cont.)

- Body elements
  - `<p>...</p>` Paragraph `<br/>` Break (new line)
  - `<img src="path/to/image" alt="text"/>` Include image

```
<p>
This paragraph
contains a lot of lines
in the source code, but the browser ignores it.
</p>

<p>
This paragraph<br/>
contains a lot of lines<br/>
in the source code, but the browser ignores it.
</p>
```

This paragraph contains a lot of lines in the source code, but the browser ignores it.

This paragraph
contains a lot of lines
in the source code, but the browser ignores it.

Important <body> Elements (Cont.)

- Anchor (hypertext links)

  `<a href="url/to/page">link text</a>`

```
<p>
 Here's a link to <a href="http://www.pugetsound.edu">Puget Sound</a>!
</p>
```

Important <body> Elements (Cont.)

- Unordered List

```
<ul>
  <li>list item</li>
  <li>list item</li>
</ul>
```

- Ordered List

```
<ol>
  <li>list item</li>
  <li>list item</li>
</ol>
```

```
<h2>Lists</h2>

<ul>
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ul>

<ol>
 <li>Apples</li>
 <li>Bananas</li>
 <li>Lemons</li>
 <li>Oranges</li>
</ol>
```

Lists

- Apples
- Bananas
- Lemons
- Oranges

1. Apples
2. Bananas
3. Lemons
4. Oranges

Important <body> Elements (Cont.)

- Tables

```
<table>
  <tr> <!-- row 1 -->
   <td>cell 1</td>
   <td>...</td>
   <td>cell N</td>
  </tr>
  <tr> <!-- row 2-->
   <td>cell 1</td>
   <td>...</td>
   <td>cell N</td>
  </tr>
</table>
```

```
<table>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>50</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>94</td>
  </tr>
  <tr>
    <td>John</td>
    <td>Doe</td>
    <td>80</td>
  </tr>
</table>
```

Jill	Smith	50
Eve	Jackson	94
John	Doe	80

Cascading Stylesheets (CSS)

- HTML versions 2 and 3 added many design elements and attributes
  - **for bold,** *<i> for italic<i>*, <u>for underline
  - Tags and attributes for colors, alignment, borders, border style, etc.

- HTML became very messy and began to lose its identity...
  - W3C decides to make conscious effort to separate design from structure
  - Deprecated many design-based HTML tags

- Style now defined in *Cascading Stylesheets (CSS)*

CSS (Cont.)

- Syntax:

```
htmlTag {
    styleComponent1: value1;
    styleComponent2: value2;
}
```

- Example

```
/* body style */
body {
    background-color: #ffffff;
    color: #5d5d5d;
    font-family: helvetica;
    font-size: 12pt;
    font-style: normal;
}

/* link style */
a { color: #1ca6cd; }
```

Learn on Your Own

- There are lots more to HTML and CSS
  - But HTML and CSS are both easy to pick up

- Just go here: http://www.w3schools.com
  - Also has tutorials on:
    - JavaScript
    - SQL
    - PHP
    - XML
    - More

The Web Booms

- Tim Berners-Lee's WWW project was successful (obviously)
  - Adopted by scientists to share research
  - Then national labs around the world
  - Making their ways to educational institutions
  - The industry... and the rest is history

- Web BOOMED in mid-90's (tech bubble)
  - Everyone wanting web presence

The Need for *Dynamic* Web Pages

- Static web pages insufficient for amount of content
  - eBay: Create a separate webpage for every item on auction?
  - Amazon: Create a separate webpage for every book?
  - Many other organizations felt the same pressure

- Problems abound:
  - Formatting bug in page template?
    - Fix the template, fix already-made pages
  - Company wants a change in design?
    - Need to apply to all webpages.
  - Company has 2 million items to sell: 2 million webpages

The Need for Dynamic Web Pages (Cont.)

- Soon, people began writing programs that output HTML files
  - These early *dynamic web pages* were written in C and more widely, Perl
  - Apache built in support for these programs
    - Called Common Gateway Interface (CGI) Scripts

- CGI scripts were really a pain to write:
  - Had to have a good handle on HTTP protocol
  - Needed to fake HTTP headers
  - Output HTML as Strings.... became very nasty to manage over time

Example CGI Script in Java

- Name this Perl script Demo.cgi

```
#!/usr/bin/perl

my $string = "java CGIDemo Hello World";
system ($string);
```

- Name this CGIDemo.java

```
public static void main (String args[])
{
    System.out.println("Content-type:text/html\n\n");
    System.out.println("<html>\n<head>\n<title>CGI Demo</title>\n</head>\n<body>");
    System.out.println("<h1 align=\"center\">Hello World!</h1><br/><br/>\n");

    for (String arg : args)
    {
            System.out.println(arg + "<br/>\n");
    }
    System.out.println("</body></html>");
}
```

Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion

PHP Hypertext Preprocessor

- **P**HP **H**ypertext **P**reprocessor
  - Created by Rasmus Lerdorf in 1994
  - The first web-programming language

- Formerly Personal Home Page Tools
  - Created by Rasmus Lerdorf in 1994
  - Set of CGI scripts written in C to create dynamic web pages

- Today: Runs on > 75% of web servers
  - Third most widely-used language (after C and Java) according to langpop.com

Quick Guide

- Variable names start with $:

```
$var = <expression>;
```

- Getting info on variables:

```
var_dump($var);
```

- Printing:

```
echo <expression>;
```

Quick Guide (Cont.)

- PHP Files should end in `.php`
  - HTML code can exist inside a PHP file.
  - Often, PHP code can be interleaved directly within HTML code:

```
<?php $title = "David's Page"; ?>
<head>
   <title> <?php echo $title; ?> </title>
</head>
```

- You can also use command-line mode:

```
$ php -a
Interactive shell

php > $var = 1;
php > $var++;
php > var_dump($var);
int(2)
```

Quick Guide (Cont.)

- This is inside `myscript.php`:

```php
<?php
        echo "<h1>Hello world</h1>\n";
        echo "<p>foo bar</p>\n";
?>
```

- Checking syntax: `php -l <path/to/filename>`

```
$ php -l code/myscript.php
No syntax errors detected in myscript.php
```

- Parse and execute file: `php -f <path/to/filename>`

```
$ php -f code/myscript.php
<h1>Hello world</h1>
<p>foo bar</p>
```

PHP Primitives (Boolean)

- Scalar types: `boolean`, `int`, `float`, `string`

- Boolean Example:
  - true or false

```
<p>
<?php
$largeFont = TruE;  //case-insensitive
if ($largeFont)
   echo '<font size="20">';
else
   echo '<font size="14">';
?>
Hello world!<br/>
</font>
</p>
```

PHP Primitives (Ints and Floats)

- Scalar types: `boolean`, `int`, `float`, `string`
- Integers:

```php
<?php
$a = 1234; // decimal number
$a = -123; // a negative number
$a = 0123; // octal number (equivalent to 83 decimal)
$a = 0x1A; // hexadecimal number (equivalent to 26 decimal)
$a = 0b11111111; // binary number (equivalent to 255 decimal)
?>
```

- Floats (double-precision):

```php
<?php
$a = 1.23456789;
$b = 1.23456780;
$epsilon = 0.00001;

//use this instead of: if ($a == $b)
if (abs($a-$b) < $epsilon)
    //do something useful

?>
```

PHP Primitives (strings)

- Scalar types: `boolean`, `int`, `float`, `string`
- Single-quoted Strings

```php
<?php
$var = "cool!";
echo 'I said, "$var"'; //I said, "$var"
?>
```

- Double-quoted Strings evaluates variables! NICE!

```php
<?php
$var = "cool!"
echo "So I said, \"$var\""; //I said, "cool!"
?>
```

Arrays

- Arrays in PHP are very flexible... more like a *hash map*.

```php
<?php
$array = array(
    "foo" => "bar",
    "bar" => "foo",
        0 => 9,
);

$array[1] = 'moo!'

var_dump($array);
?>
```

```
array(4) {
  ["foo"]=>
  string(3) "bar"
  ["bar"]=>
  string(3) "foo"
  [0]=>
  int(9)
  [1]=>
  string(4) "moo!"
}
```

This is the output from the above code

Type Juggling

- PHP is dynamically typed
  - Known as *Type Juggling* in PHP lingo

```php
<?php

$number_of_toys = 10;
$toys_category = "123 Puzzles";
$toys_age_limit = "5.5";
$toys_price = "2e2";

$result1 = $number_of_toys + $toys_category;
$result2 = $number_of_toys + $toys_age_limit;
$result3 = $number_of_toys + $toys_price;

echo $result1."<br/>"; //output integer value: 133
echo $result2."<br/>"; //output float value: 15.5
echo $result3."<br/>"; //output integer value: 210

?>
```

From http://phppot.com/php/php-type-juggling/

Comparison Operators

	Meaning
`$a == $b`	Equals after type juggling
`$a === $b`	Equals, and are of the same type
`$a != $b`	Not equals after type juggling
`$a !== $b`	Not equals, or are of different types
`$a < $b`	Less than?
`$a > $b`	Greater than?
`$a <= $b`	Less than equals, after type juggling
`$a >= $b`	Greater than equals, after type juggling

Comparison Operators (Cont.)

```php
$foo = 10;

var_dump($foo == 10);  //true

var_dump($foo == '10'); //true!

var_dump($foo === 10); //true

var_dump($foo === '10'); //false!

var_dump($foo <= '10'); //true!
```

Operations

	Meaning	Example
+, -, *, **, /, %	(Usual num ops)	`var_dump(2**3); //8`
.	String concatenation	`var_dump('foo' . 'bar ' . 88) //foobar 88`
&&, ||, !	(Usual boolean ops)	
$a++, ++$a	(Usual num ops)	
$a--, --$a	(Usual num ops)	
+=, -=, *=, /*, **=	(Usual num ops)	
.=	String concat	

Conditionals

- If-then-else

```php
<?php
if (cond)
{
    echo "That was <b>true</b>\n";
}
else
{
    echo "That was <b>false</b>\n";
}
?>
```

- Integration with HTML (same result as above)

```php
<?php if (cond) { ?>
 That was <b>true</b>
<?php } else { ?>
 That was <b>false</b>
<?php } ?>
```

Conditionals Else-If

- Else-Ifs

```
if (cond)
{
    //statement
}
elseif (cond)
{
    //statement
}
elseif (cond)
{
    //statement
}
else
{
    //statement
}
```

Loops (For & While)

- While and For loops also have familiar syntax

```php
<?php

while (cond)
{
    //loop statements
}

for (init; cond; progress)
{
    //loop statements
}

?>
```

Loops (Cont.)

- Loops can also integrate with HTML

```
<?php $n = 5; ?>
<ul>
<?php
    for ($i=0; $i<$n; $i++)
        echo "<li>List item: $i</li>\n";
?>
</ul>
```

- Output:

```
<ul>
 <li>List item: 0</li>
 <li>List item: 1</li>
 <li>List item: 2</li>
 <li>List item: 3</li>
 <li>List item: 4</li>
</ul>
```

Arrays

- Recall: all PHP arrays are actually associative arrays (or HashMaps)
  - Created with the array(...) function

```php
$list = array(
    "foo" => "bar",
    "bar" => true,
        9 => 4,
        0 => "bla",
);
```

- Accessed as expected...

```php
var_dump($list["foo"]); // string(3) "bar"

var_dump($list["9"]);   // int(4)

var_dump($list["8"]);   // NULL
```

Arrays (Cont.)

- Single command to print out all contents of array: `print_r($list)`
  - Good for debugging, but not much else
  - Output:

```
Array
(
    [foo] => bar
    [bar] => foo
    [9] => 4
    [0] => bla
)
```

Array Access (Foreach loop)

- How to access elements in an associative array?
  - No standard index... so how do we know how to loop?

- If you don't care about the key:

```php
foreach (array_expression as $value) {
    //statement
}
```

- If you want the key:

```php
foreach (array_expression as $key => $value) {
    //statement
}
```

Foreach Loops

```php
<?php
$list = array(
    "foo" => "bar",
    "bar" => true,
        9 => 4,
        0 => "bla",
);

foreach ($list as $k => $v) {
    echo "$k holds $v\n";
}
?>
```

- Output:

```
foo holds bar
bar holds true
9 holds 4
0 holds bla
```

Functions

- Functions in PHP are defined as follows:
  - Notice: no return type; just return when needed

```php
<?php
function functionName(paramList)
{
    //body
}
?>
```

- Example:

```php
<?php
function max($a, $b) {
    if ($a < $b)
        return $b;
    return $a;
}

echo "The larger of 4 and 5 is: ". max(4,5); // call the function
?>
```

Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals
  - PDO Database Connectivity
- Conclusion

PHP Superglobals

- *Superglobals* are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

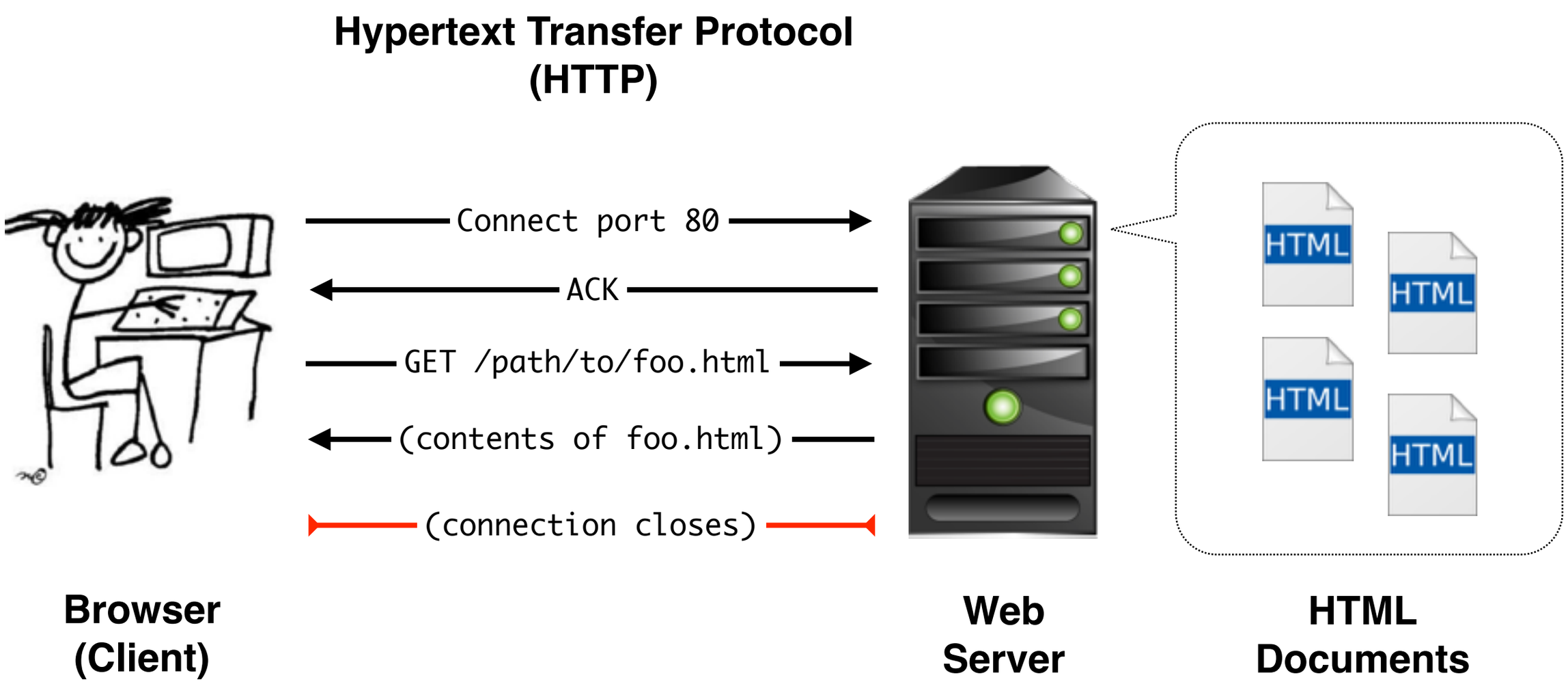
- Here are a few important ones:
  - $_GLOBALS[...]: user-defined (think public static variables in Java)
  - $_COOKIE[...]: a cookie we set on a browser
  - $_GET[...]: variables passed from URLs
  - $_POST[...]: variables passed from HTML forms
  - $_SERVER[...]: information about the web server

PHP Superglobals

- *Superglobals* are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

- Here are a few important ones:
  - $_GLOBALS[...]: user-defined (think public static variables in Java)
  - $_COOKIE[...]: a cookie we set on a browser
  - $_GET[...]: variables passed from URLs
  - $_POST[...]: variables passed from HTML forms
  - $_SERVER[...]: information about the web server

Recall How the Web Works: HTTP



Important: HTTP is "stateless": TCP connection is not persisted. Doesn't remember you next time you request a page. Need *cookies* to remember browsers...

Cookies

- If HTTP is stateless, how do sites like Amazon and Facebook remember that I'm logged in?

- *Cookies* are data that websites can store on your browser so that it can remember you in a later HTTP session.

- PHP has built-in cookie handling mechanisms

Setting Cookies

- Setting a cookie (browser has to accept them)
  - Caveat: Cookies are a part of the HTTP header, and *must be set before* any other content is sent to the browser

Cookie name and cookie value

```php
<?php
   setcookie("userID", "dchiu", time() + (86400 * 30)); // 86400 = 1 day
?>
<!DOCTYPE html>
<html>
   <!-- blah blah blah -->
</html>
```

Expiration (duration): Time from now in seconds.
Value of 0 means end of session (when browser closes)

Reading Cookies

- Later, a user browses back to your web page... to remember who they are, we need to see if the userID cookie is set!
- Enter the `$_COOKIE[...]` superglobal

```php
<?php
//do we know this user?
if (isset($_COOKIE["userID"]))
{
    $userName = $_COOKIE["userID"]; //get the value of the cookie from browser
    logUserIn($userName);
}
else
{
    //don't know this person (or cookie expired)
}
?>

<!DOCTYPE html>
<html>
   <!-- blah blah blah -->
</html>
```

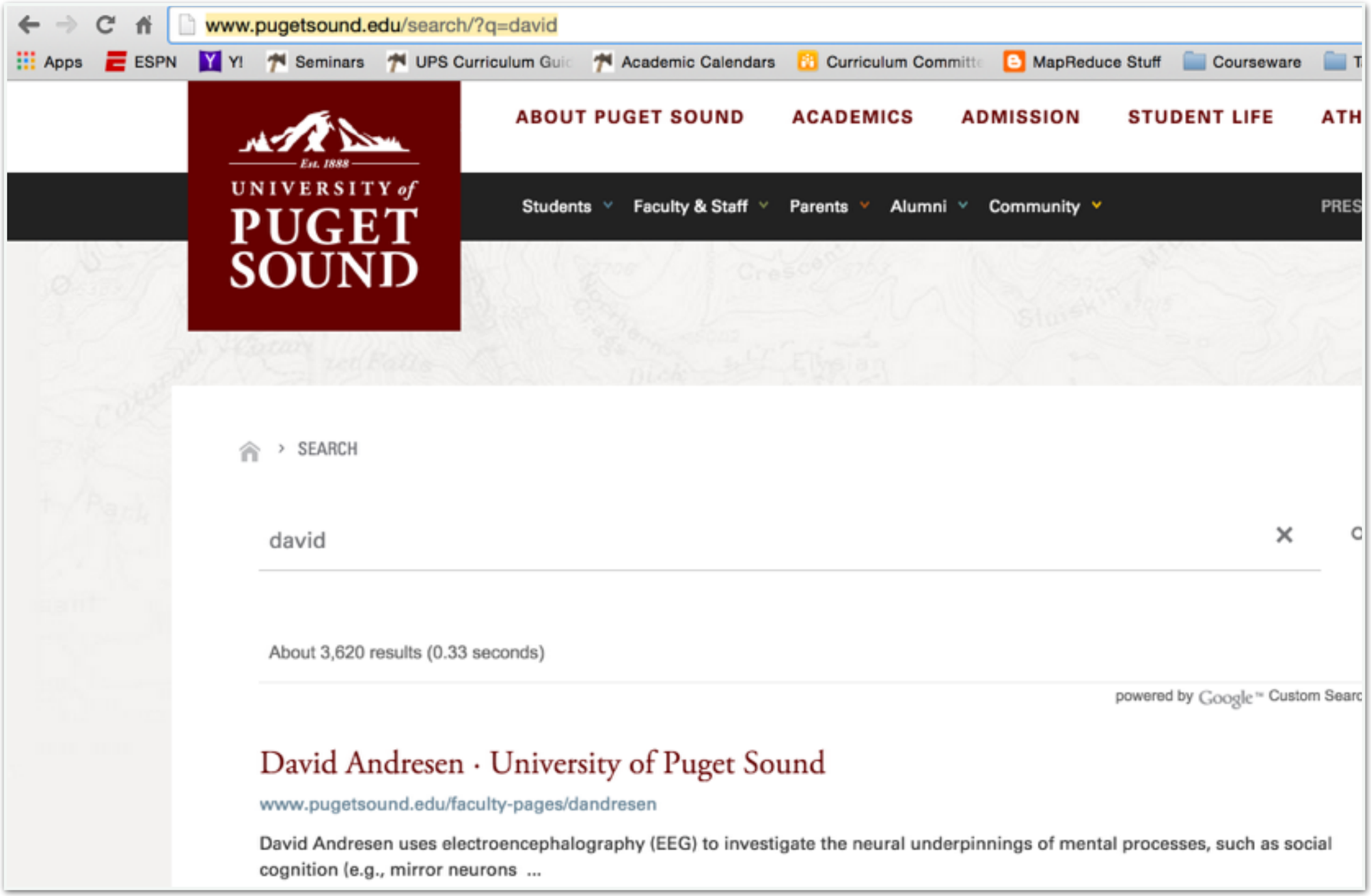
PHP Superglobals

- *Superglobals* are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

- Here are a few important ones:
  - $_GLOBALS[...]: user-defined (think public static variables in Java)
  - $_COOKIE[...]: a cookie we set on a browser
  - $_GET[...]: variables passed from URLs
  - $_POST[...]: variables passed from HTML forms
  - $_SERVER[...]: information about the web server

You Can Pass Variables via a URL

- Ever wonder what ?, = and & mean in a URL?



What's in a URL?

- URL Syntax

```
protocol:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

- Examples:
  - Locates a file on my local machine

```
file://localhost/Users/David/Documents/foo.txt
```

  - Locates a directory on another machine using FTP

```
ftp://ftp.at.debian.org/debian-cd/8.2.0/i386/iso-dvd
```

What's in a URL? (Cont.)

- URL Syntax

```
protocol:[//[user:password@]host[:port]][/]path[?query][#fragment]
```

- Examples:
  - Get Lecture 2 from my course page (login automatically)

```
http://CS455:p4ssword@cs.pugetsound.edu/~dchiu/CS455/notes/CS455_1-intro.pdf
```

  - Sends a "query" (i.e., variables) to the server

```
http://cs.pugetsound.edu/~dchiu/CS455/webstuff/showGetvars.php?foo=1&bar=test
```

Inside showGetvars.php

- Just use the $_GET[...] superglobal to access any variable and its value that was passed!

- showGetvars.php:

```
<!DOCTYPE html>
<html>
<body>
  <?php
      if ($_GET["foo"] == 10)
          echo "Foo!";
      if ($_GET["bar"] == 20)
          echo "Bar!";
  ?>
</body>
</html>
```

PHP Superglobals

- *Superglobals* are variables that are accessible in all scopes.
  - They are all associative arrays (hashmaps)

- Here are a few important ones:
  - $_GLOBALS[...]: user-defined (think public static variables in Java)
  - $_COOKIE[...]: a cookie we set on a browser
  - $_GET[...]: variables passed from URLs
  - $_POST[...]: variables passed from HTML forms
  - $_SERVER[...]: information about the web server

HTML Forms

- You can make forms with HTML:

Where does it take you when you click the *submit* button?

Which HTTP method to use to send data? Possible values: *post or get (USE POST ALWAYS)*

```
<form action="formHandler.php" method="post">

  Name: <input type="text" name="name"/><br/>

  E-mail: <input type="text" name="email"/><br/>

  <input type="submit"/>
</form>
```

The *submit* button

HTML Forms (Cont.)

- You can make forms with HTML:


```
<form action="formHandler.php" method="post">

  Name: <input type="text" name="name"/><br/>

  E-mail: <input type="text" name="email"/><br/>

  <input type="submit"/>
</form>
```

Draw a *textbox*

Name of the variable

HTML Forms (Cont.)

- Password Field

```
Enter your password: <input type="password" name="pwd"/>
```

- Checkbox

```
Today I am:<br/>
<input type="checkbox" name="happy"/> Happy<br/>
<input type="checkbox" name="angry"/> Angry<br/>
<input type="checkbox" name="sad"/> Sad<br/>
```

- Dropdown List

```
<select name="country">
  <option value="ca">Canada</option>
  <option value="zn">China</option>
  <option value="fr">France</option>
  <option value="in">India</option>
  <option selected="selected" value="us">U.S.</option>
</select>
```

HTML Forms (Cont.)

- File

```
Upload a file:<br/>
<input type="file" name="filename"/>
```

Upload a file:
Choose File No file chosen Submit

- Hidden *(actually really useful!)*

```
<input type="hidden" name="var" value="val" />
```

- Radio Options

```
Your pet is a:<br/>
<input type="radio" name="species" value="cat"/> Cat<br/>
<input type="radio" name="species" value="dog"/> Dog<br/>
<input type="radio" name="species" value="fish"/> Fish<br/>
<input type="radio" name="species" value="lizard"/> Lizard<br/>
```

Your pet is a:
- Cat
- Dog
- Fish
- Lizard

Where Does the Form Take Us?

- We need a (PHP) script to process the form data!
  - The superglobal $_POST[...] hold all those variables from the form
    - Assuming you used the "post" method in your form

```php
<?php

var_dump($_POST["name"]);
var_dump($_POST["email"]);

?>
```

  - Typically, this PHP script would insert the collected data into a database...

Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion

PHP Database Connectivity

- There are many free PHP database libraries...
  - We focus on PHP Data Objects (PDO)
  - Need to be installed as an add-on library to PHP
- From your Linux shell:

```
$ sudo yum install php-pdo
```

- PDO is not the only way... other libraries exist

Assumptions

- Caveat: This tutorial written for SQLite3

- Assumptions:
  - SQLite3 database already exists on filesystem (*i.e.*, you used `.save` or `.backup` to create the file)
  - Apache web server needs write access to both the database file and the directory where it's located

- The PDO library is object-oriented. Pro-tip:

```
$obj = new Class(..);  //instantiation
$obj->method(..);      //method call
```

(Dis)Connecting to/from the Database

- PDO Object Instantiation: `new PDO(string $pathToDBFile)`

```php
<?php
    try
    {
        //open the sqlite database file
        $db = new PDO('sqlite:database/airport.sqlite3');

        // Set errormode to exceptions
        $db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

        //queries and stuff down here

        //disconnect from database
        $db = null;
    }
    catch(PDOException $e)
    {
        echo 'Exception : '.$e->getMessage();
    }
?>
```

"Write" SQL Statements (Insert, Delete, Update)

- Use this: `public int exec(string $statement)`
  - Executes given SQL statements and returns number of affected rows

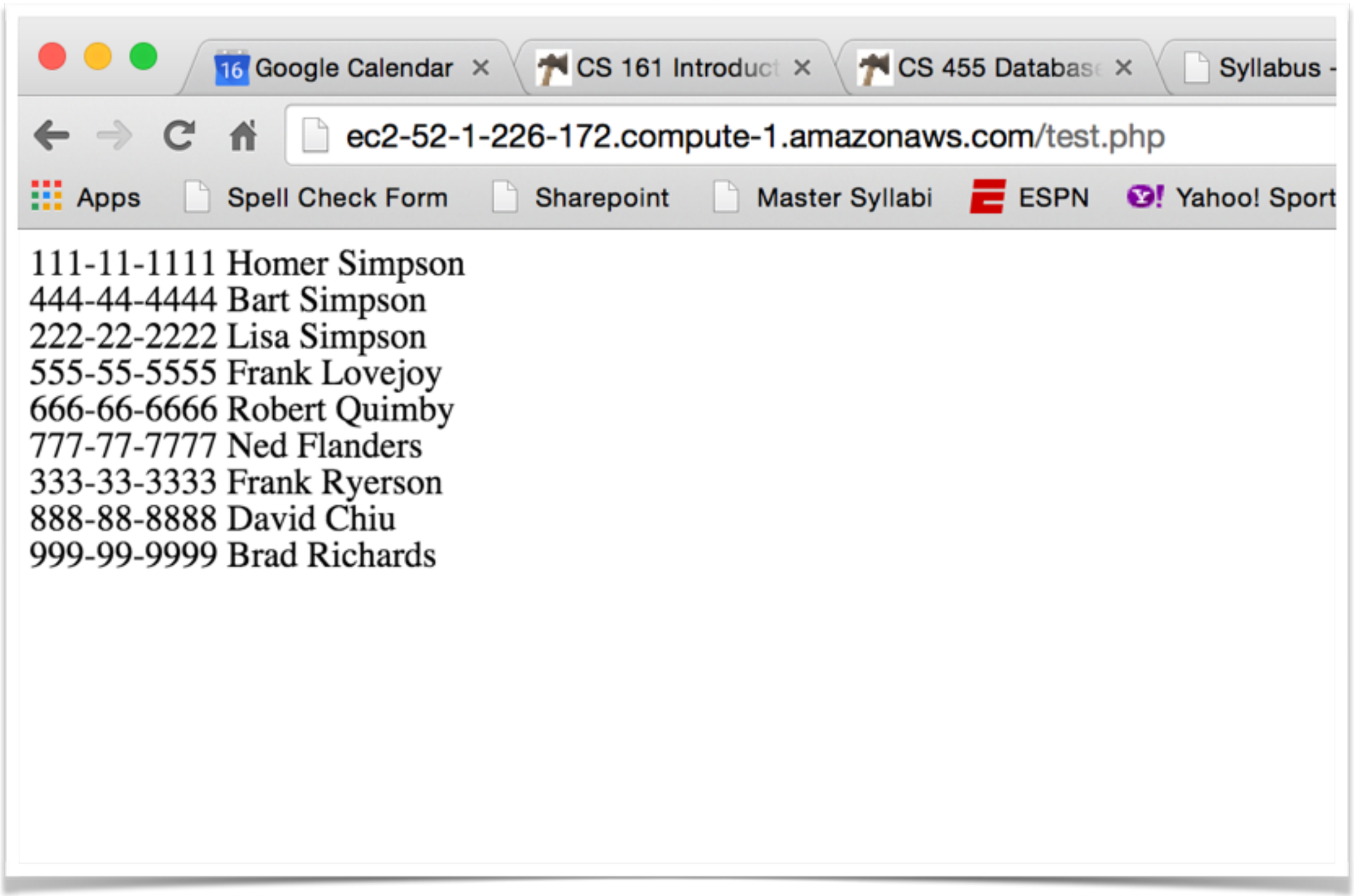
```php
<?php
try {
    //open the sqlite database file
    //(code omitted)

    //insert some new tuples into the passenger relation
    $db->exec("insert into passengers values ('David', NULL, 'Chiu', '888-88-8888');");
    $db->exec("insert into passengers values ('Brad', NULL, 'Richards', '999-99-9999');");

    //now put Brad and David on the same flight
    $db->exec("insert into onboard values ('888-88-8888',4,'32B')");
    $db->exec("insert into onboard values ('999-99-9999',4,'32C')");

    //disconnect from database
    $db = null;
}
catch(PDOException $e) {
    echo 'Exception : '.$e->getMessage();
}
?>
```

"Read" SQL Statements: Select

- With select, we don't care about number of rows affected, we want the result set that was returned!
- Syntax: `public PDOStatement query(string $statement)`
- Return Value: An array of tuples
  - Each tuple is an associative array of *attribute => value* pairs

```php
//select all passengers
$result = $db->query('SELECT * FROM passengers;');

foreach($result as $tuple)
   echo $tuple['ssn']." ".$tuple['f_name']." ".$tuple['l_name']."<br/>";
```

Results from Previous Query



Putting Results in a <table>

```php
<?php
try {
    //open the sqlite database file (code omitted)

    echo '<table border="1">';
    echo '<tr><td>SSN</td><td>Name</td><td>Seat</td><td>Departure</td><td>Arrival</td></tr>';


    //select all passengers
    $result = $db->query('SELECT * FROM passengers NATURAL JOIN onboard NATURAL JOIN flight');

    foreach($result as $tuple) {
      echo "<tr><td>".$tuple['ssn']."</td>";
      echo "<td>".$tuple['f_name']." ".$tuple['l_name']."</td>";
      echo "<td>".$tuple['seat']."</td>";
      echo "<td>".$tuple['dep_loc']." ".$tuple['dep_time']."</td>";
      echo "<td>".$tuple['arr_loc']." ".$tuple['arr_time']."</td>";
      echo "</tr>";
    }
   //disconnect from database
   $db = null;
}
catch(PDOException $e)
{
    echo 'Exception : '.$e->getMessage();
}
?>
```

Group Activity

- Left half of room
  - Write an HTML form addPassenger.html that POSTs to insert.php
  - Text-boxes for firstname, middlename, lastname, ssn (hidden)

- Right half of room
  - Insert a new tuple into the Passenger table with the given form-data
  - The passenger was created using:

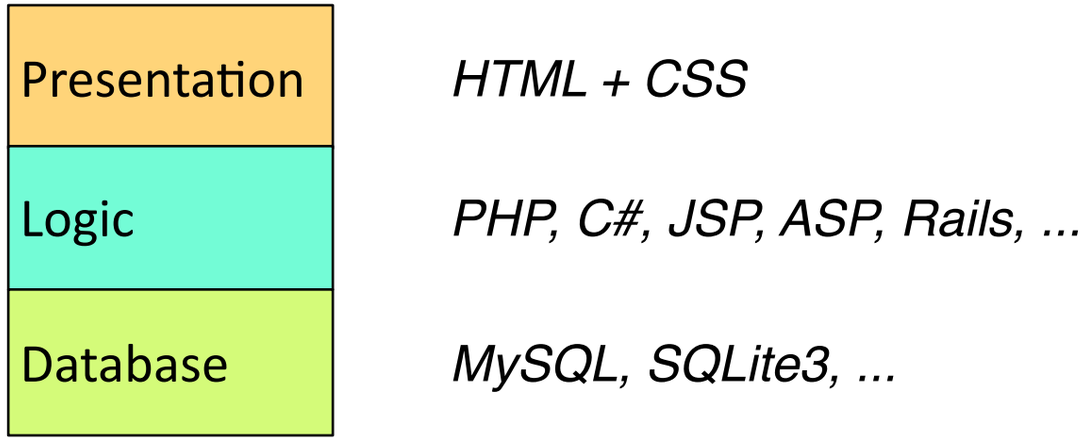
```
create table Passenger (
  ssn    TEXT     PRIMARY KEY,
  first  TEXT     NOT NULL,
  middle TEXT,
  last   TEXT     NOT NULL
);
```

Outline

- History of the Web
- Introduction to HTML
- Dynamic Web Programming with PHP
  - PHP Basics
  - Superglobals: Cookies and Form Handling
  - PDO Database Connectivity
- Conclusion

Conclusion

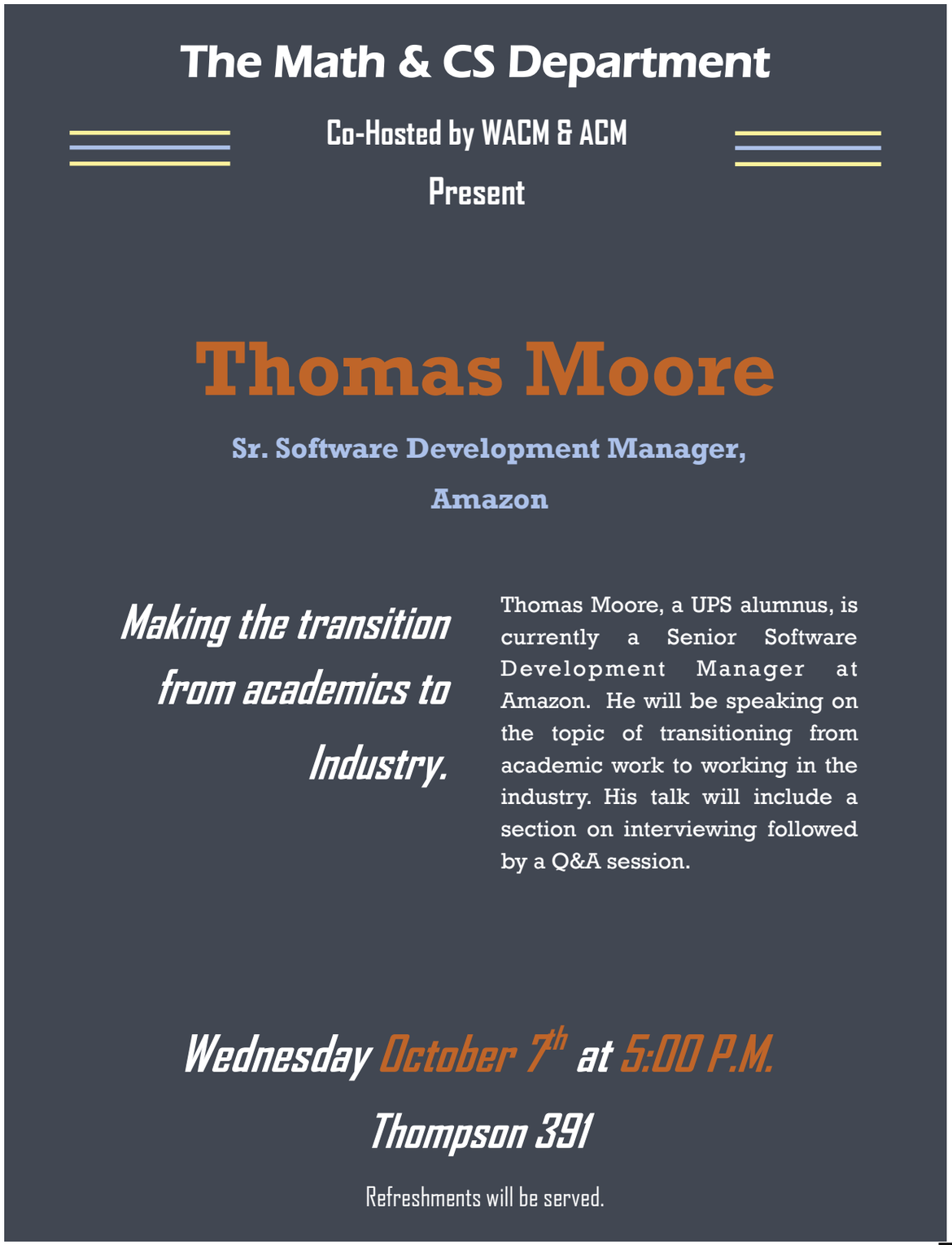
- Dynamic web programming boot camp
  - PHP is a huge language... highly recommend that you learn more on your own
- Many of today's websites follow the 3-tier architecture:



- Further topics for exploration for the Web-curious:
  - JavaScript, NodeJS, Ajax, MongoDB, XML (DTD, XPath, XQuery)

Administrivia: 10/05

- Talk tonight!
  - Thomas Moore from Amazon
- HW2 graded:
  - 81 average
- Project 2 extended to Oct 21
- Exam Friday:
  - Calculator allowed
  - Half-page of notes allowed



Administrivia: 10/12

- Thoughts on Exam I?

- Talk on Wednesday:
  - What *Is* Machine Learning, Anyway?
    - Alex King (Hitachi Consulting)
  - Location: Thompson 395, 5pm

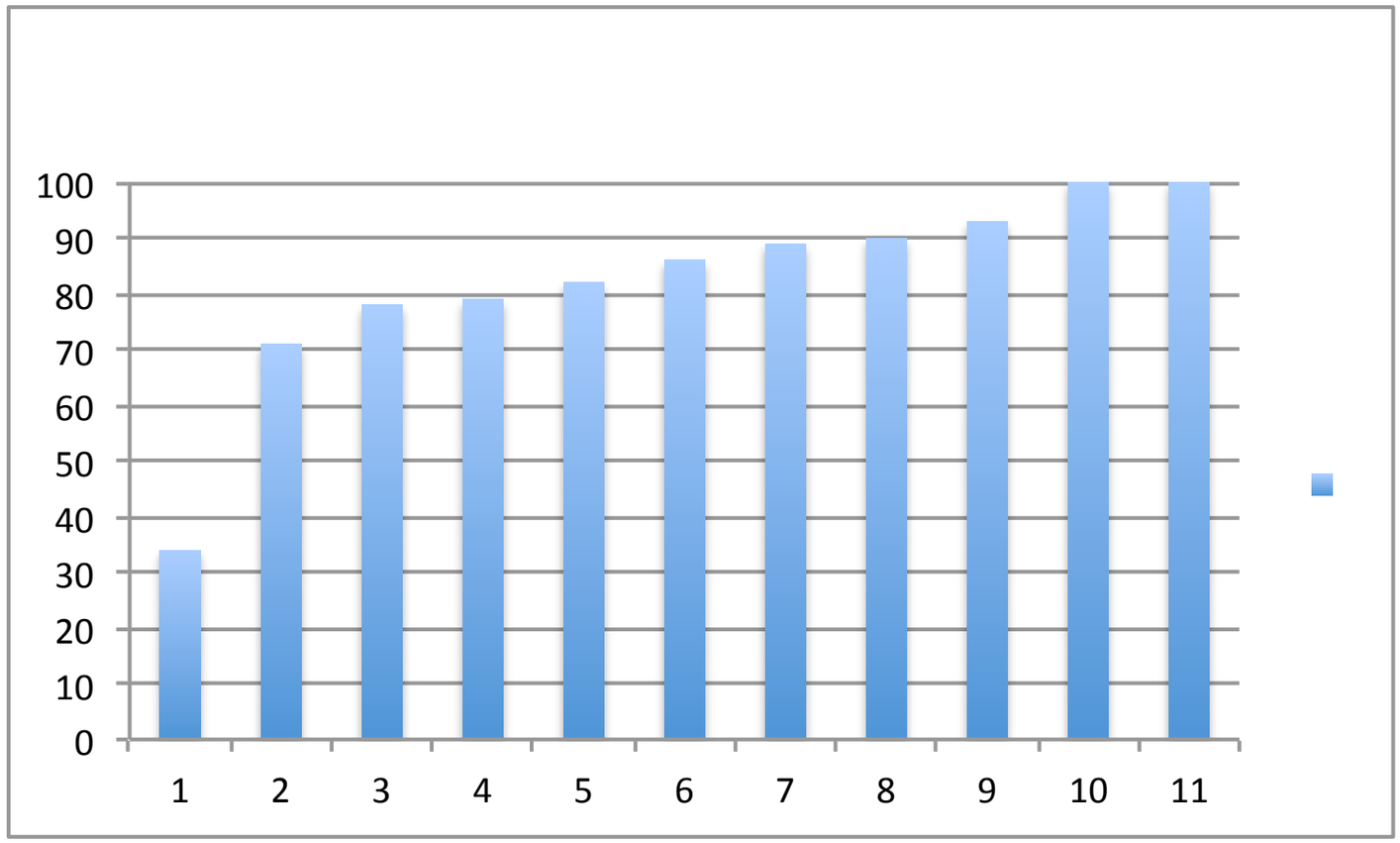
Administrivia: 10/14

- Project 2 status?
  - Due in a week (10/21)

- Talk tonight!
  - What *Is* Machine Learning, Anyway?
    - Alex King (Hitachi Consulting)
  - Location: Thompson 395, 5pm
- Talk tomorrow!
  - Horn's Conjecture: Solving Problems with Puzzles
    - Natalie Hobson (University of Georgia)
  - Location: Thompson 395, 5pm

Exam 1 Results

- Exam I results
  - Avg = 82, stdev = 18
  - Worth 15% of final grade
- The good:
  - T/F, multi-choice
  - Keys, create table DDL
- Needs work:
  - The tougher SQL queries



Administrivia: 10/16

- Project 2 due next Wednesday

- Homework 4 posted:
  - Entity-Relationship Diagrams (ERD)

- Posted soon: Project 3

TWiCS: Tesla Autopilot (Sort of)

- Tesla launches Autopilot: $2,500 software update
  - Hands-free, pedal-free highway driving
  - Lane change still initiated by driver (signaling)
  - "... uses sensors to scan the road in all directions"
    - Autopilot automatically adjusts steering, pedal, and brakes

- Think: What are some things that could screw up autopilot?

- http://www.nytimes.com/2015/10/16/automobiles/tesla-adds-high-speed-autonomous-driving-to-its-bag-of-tricks.html