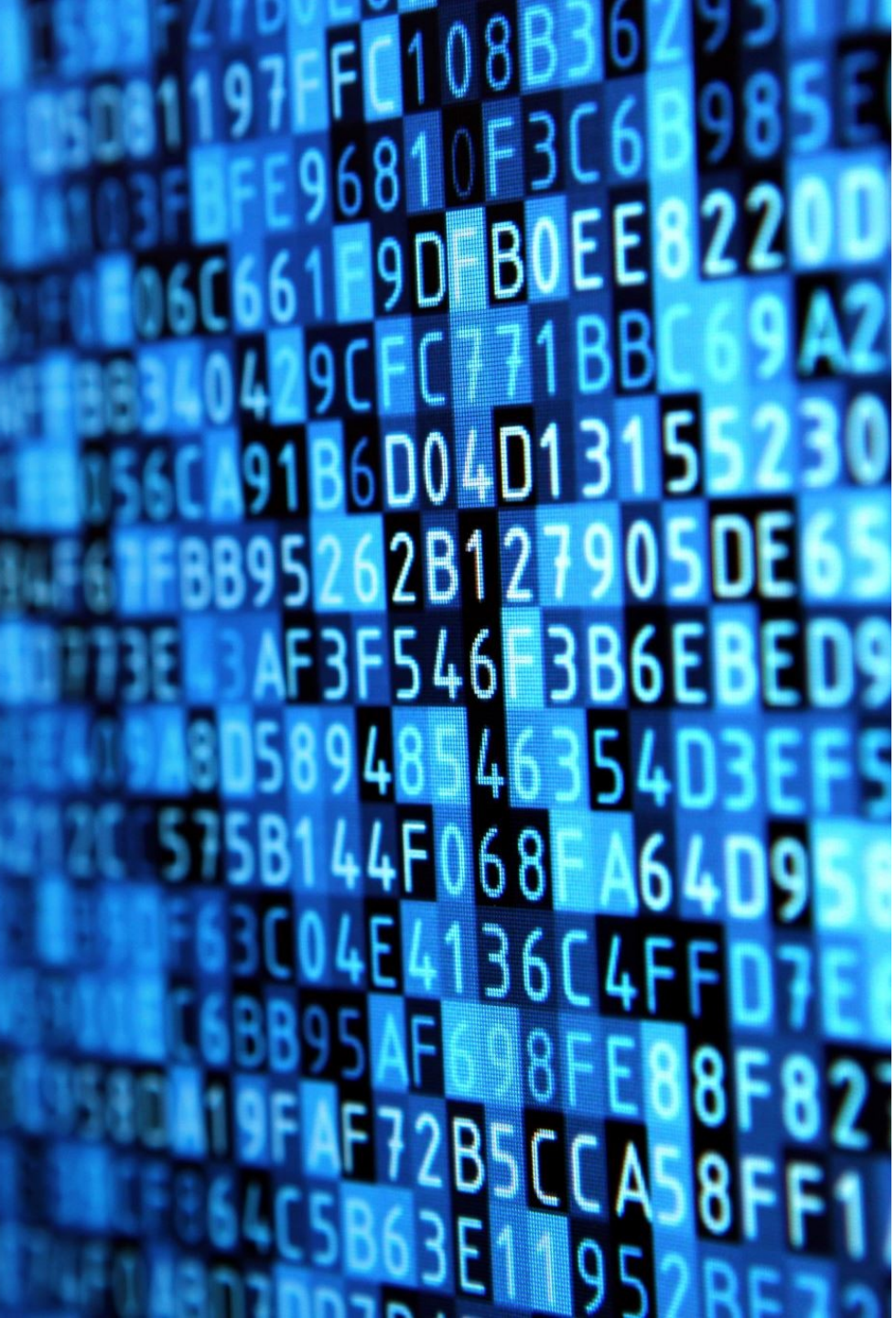
Web Programming

CS-406

Lecture # 09

PHP State Information Variables



Understanding State Information

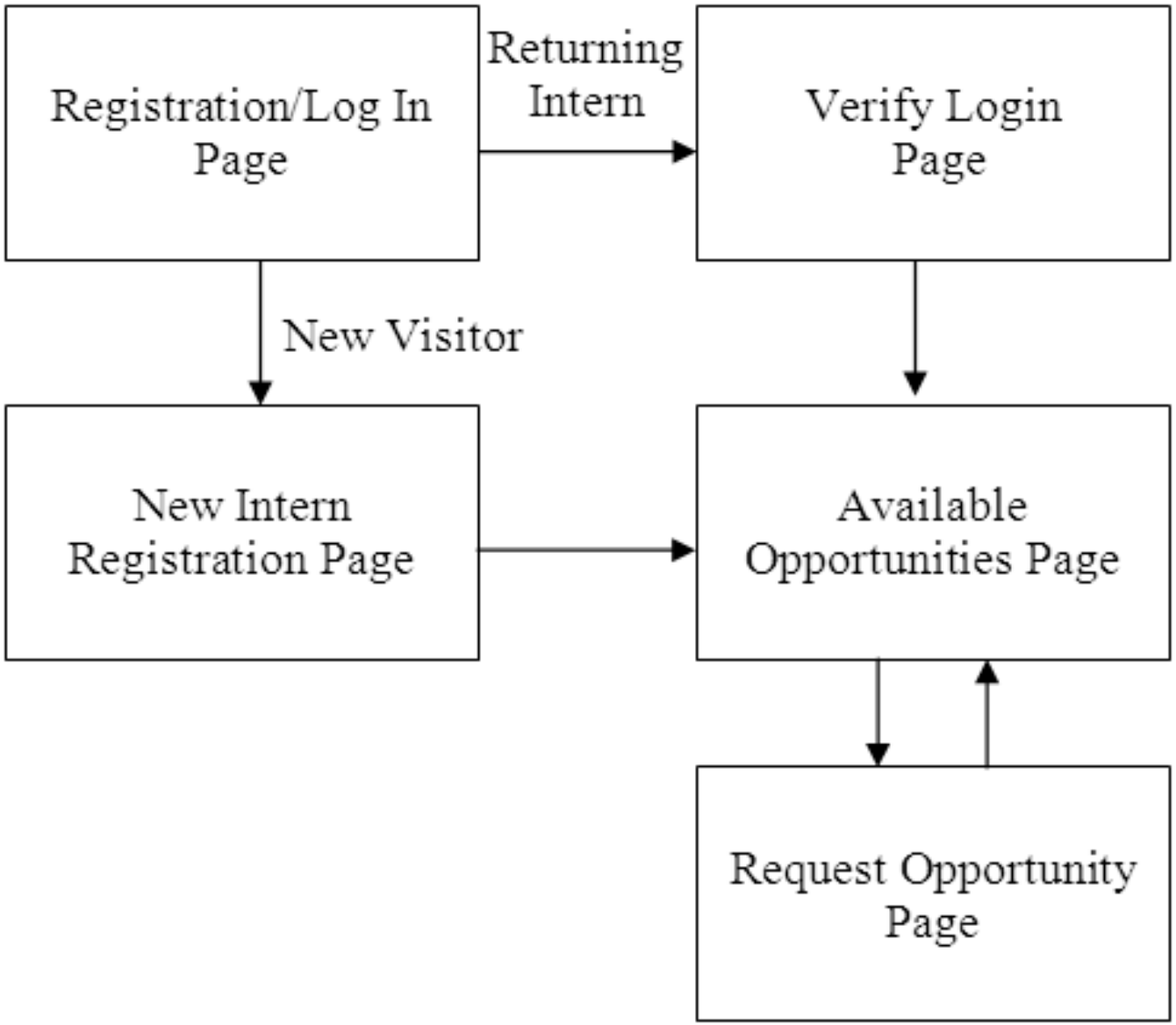
- Information about individual visits to a Web site is called **state information**
- HTTP was originally designed to be **stateless** – Web browsers store no persistent data about a visit to a Web site
- **Maintaining state** means to store persistent information about Web site visits with hidden form fields, query strings, cookies, and sessions

Understanding State Information (continued)

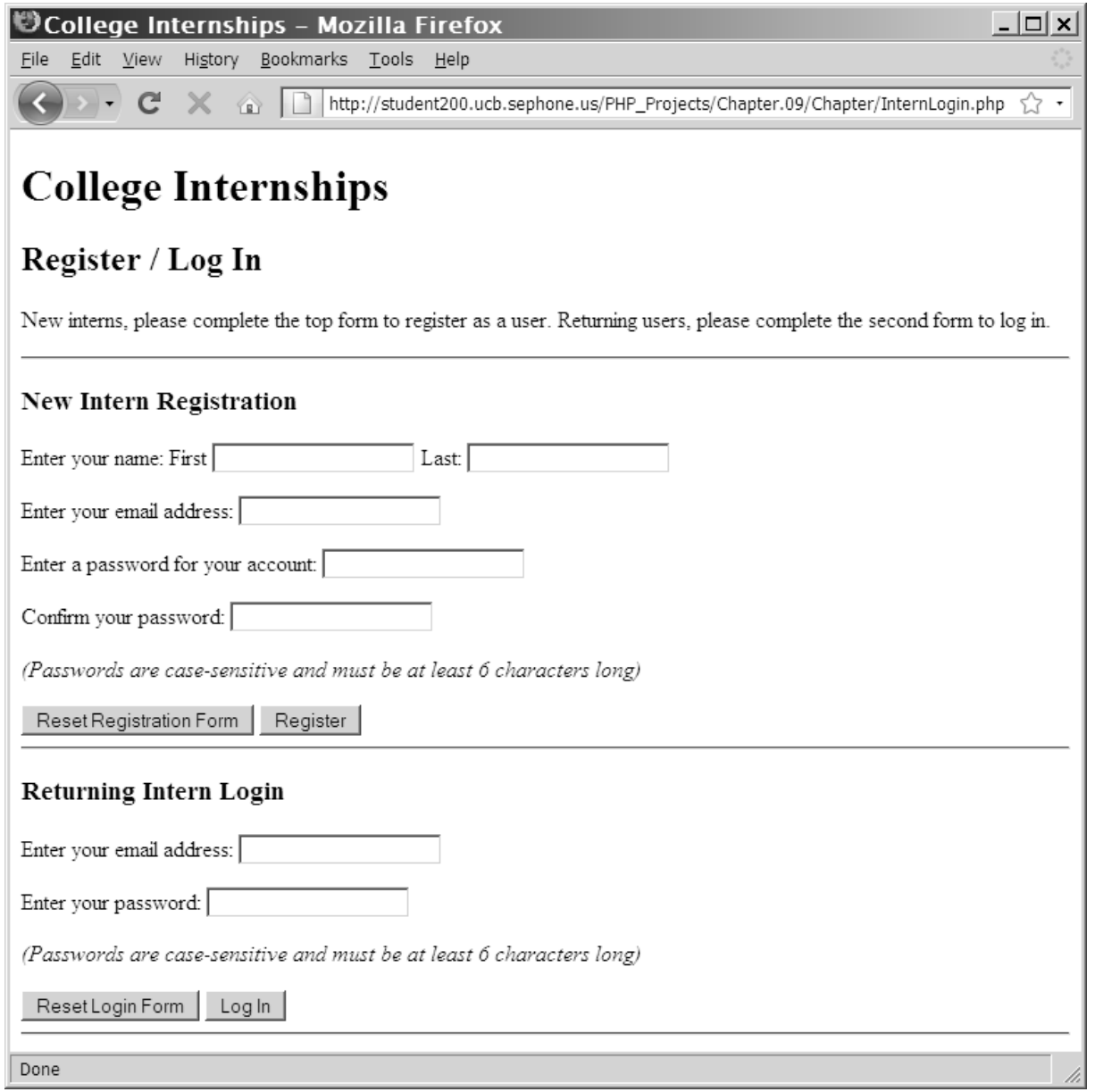
- Customize individual Web pages based on user preferences
- Temporarily store information for a user as a browser navigates within a multipart form
- Allow a user to return to specific locations within a Web site
- Provide shopping carts that store order information

Understanding State Information (continu

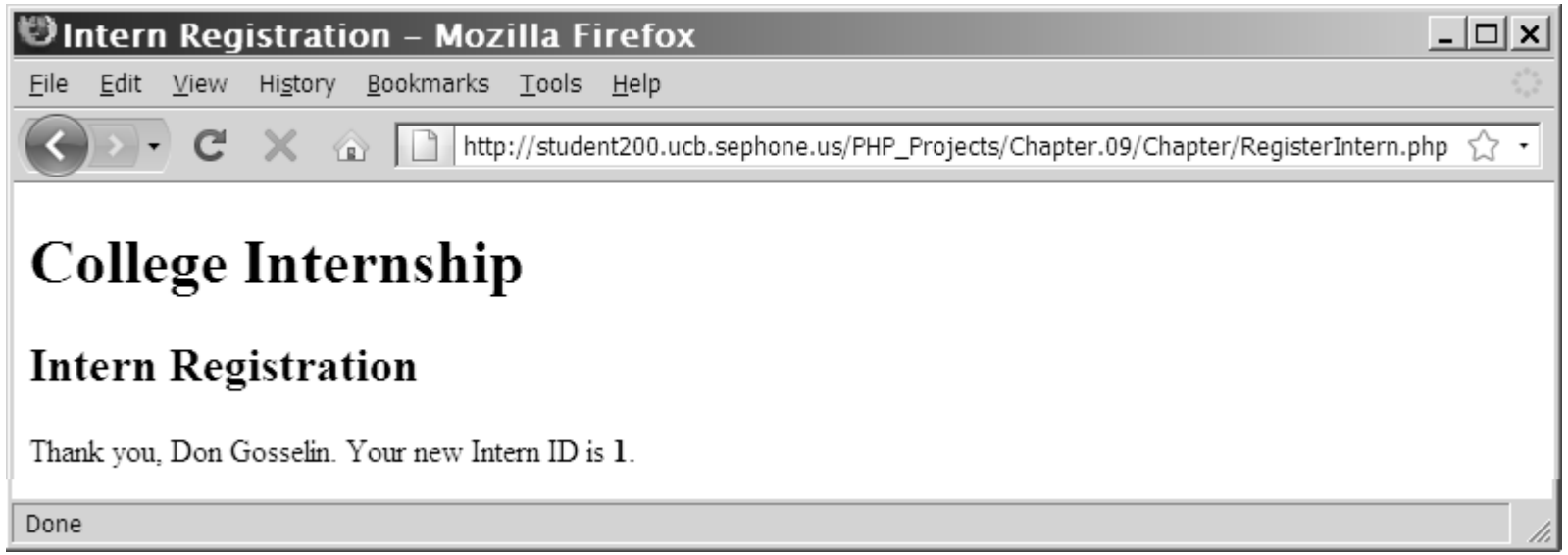
- Store user IDs and passwords
- Use counters to keep track of how many times a user has visited a site
- The four tools for maintaining state information with PHP are:
  - Hidden form fields
  - Query strings
  - Cookies
  - Sessions


Registration/Log In Page
Returning Intern
Verify Login Page
New Visitor
New Intern Registration Page
Available Opportunities Page
Request Opportunity Page

Understanding State Information (continued)



Understanding State Information (continued)



Understanding State Information (continued)

Understanding State Information (continued)

Using Hidden Form Fields to Save State Information

- Create hidden form fields with the `<input>` element
- **Hidden form fields** temporarily store data that needs to be sent to a server that a user does not need to see
- Examples include the result of a calculation
- The syntax for creating hidden form fields is:

```
<input type="hidden">
```

Using Hidden Form Fields to Save State Information (continued)

- Hidden form field attributes are **name** and **value**
- When submitting a form to a PHP script, access the values submitted from the form with the `$_GET[]` and `$_POST[]` autoglobals
- To pass form values from one PHP script to another PHP script, store the values in hidden form fields

Using Hidden Form Fields to Save State Information (continued)

```
echo "<form method='post' " .
        "
action='AvailableOpportunities.php'>\n";
echo "<input type='hidden' name='internID'
" .
        " value='$InternID'>\n";
echo "<input type='submit' name='submit' "
.
        " value='View Available
Opportunities'>\n";
echo "</form>\n";
```

Using Query Strings to Save State Information

- A **query string** is a set of name=value pairs appended to a target URL
- Consists of a single text string containing one or more pieces of information
- Add a question mark (?) immediately after the URL followed by the query string that contains the information you want to preserve in name/value pairs

Using Query Strings to Save State Information (continued)

- Separate individual name=value pairs within the query string using ampersands (&)
- A question mark (?) and a query string are automatically appended to the URL of a server-side script for any forms that are submitted with the GET method

```
<a href="http://www.example.com/TargetPage
.php?firstName=Don&lastName=Gosselin&
occupation=writer">Link Text</a>
```

Using Query Strings to Save State Information (continued)

```
echo
  "{$_GET['firstName'
  ]}
  {$_GET['lastName']}
 is a
  {$_GET['occupation'
  ]}. ";
```

Using Cookies to Save State Inform

- Query strings do not permanently maintain state information
- After a Web page that reads a query string closes, the query string is lost
- To store state information beyond the current Web page session, Netscape created cookies
- **Cookies**, or magic cookies, are small pieces of information about a user that are stored by a Web server in text files on the user's computer

Using Cookies to Save State Information (continued)

- **Temporary cookies** remain available only for the current browser session
- **Persistent cookies** remain available beyond the current browser session and are stored in a text file on a client computer
- Each individual server or domain can store between 20 and 70 cookies on a user's computer
- Total cookies per browser cannot exceed 300
- The largest cookie size is 4 kilobytes

Creating Cookies

- The syntax for the `setcookie()` function is:

  `setcookie(`*`name`*
  `[,`*`value ,expires, path, domain, secure`*`])`
- You must pass each of the arguments in the order specified in the syntax
- To skip the `value`, `path`, and `domain` arguments, specify an empty string as the argument value
- To skip the `expires` and `secure` arguments, specify 0 as the argument value

Creating Cookies (continued)

- Call the `setcookie()` function before sending the Web browser any output, including white space, HTML elements, or output from the `echo()` or `print()` statements
- Users can choose whether to accept cookies that a script attempts to write to their system
- A value of TRUE is returned even if a user rejects the cookie

The name and value Arguments

- Cookies created with only the `name` and `value` arguments of the `setcookie()` function are temporary cookies because they are available for only the current browser session

```
<?php
setcookie("firstName", "Don");
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>College Internships</title>
...
```

The name and value Arguments (continued)

- The `setcookie()` function can be called multiple times to create additional cookies – as long as the `setcookie()` statements come before any other output on a Web page

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
```

The `name` and `value` Arguments (continued)

- The following code creates an associative cookie array named `professional[]` that contains three cookie values:

```
setcookie("professional['firstName']", "Don");
setcookie("professional['lastName']", "Gosselin");
setcookie("professional['occupation']", "writer");
```

The expires Argument

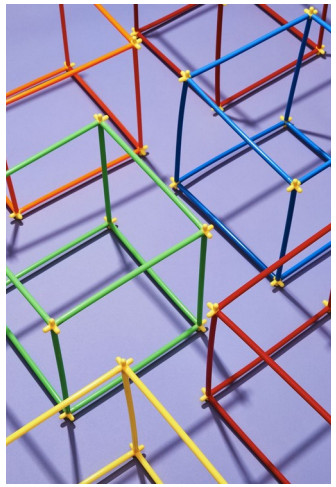
- The `expires` argument determines how long a cookie can remain on a client system before it is deleted
- Cookies created without an `expires` argument are available for only the current browser session
- To specify a cookie's expiration time, use PHP's `time()` function

```
setcookie("firstName", "Don", time()+3600);
```

The `path` Argument

- The path argument determines the availability of a cookie to other Web pages on a server
- Using the path argument allows cookies to be shared across a server
- A cookie is available to all Web pages in a specified path as well as all subdirectories in the specified path

```
setcookie("firstName", "Don", time()+3600, "/marketing/");
```



The `domain` Argument

- The `domain` argument is used for sharing cookies across multiple servers in the same domain
- Cookies cannot be shared outside of a domain

```
setcookie("firstName", "Don”, time()
  +3600, "/", ".gosselin.com");
```

The `secure` Argument

- The `secure` argument indicates that a cookie can only be transmitted across a secure Internet connection using HTTPS or another security protocol
- To use this argument, assign a value of 1 (for TRUE) or 0 (for FALSE) as the last argument of the `setcookie()` function

```
setcookie("firstName”,"Don", time()+3600,"/",
 ".gosselin.com", 1);
```

Reading Cookies

- Cookies that are available to the current Web page are automatically assigned to the `$_COOKIE` autoglobal
- Access each cookie by using the cookie name as a key in the associative `$_COOKIE[]` array

```
echo $_COOKIE['firstName'];
```

- Newly created cookies are not available until after the current Web page is reloaded

Reading Cookies (continued)

- To ensure that a cookie is set before you attempt to use it, use the `isset()` function

```
setcookie("firstName", "Don");
setcookie("lastName", "Gosselin");
setcookie("occupation", "writer");
if (isset($_COOKIE['firstName'])
     && isset($_COOKIE['lastName'])
     && isset($_COOKIE['occupation']))
     echo "{$_COOKIE['firstName']} {$_COOKIE['lastName']}
           is a {$_COOKIE['occupation']}.";
```

Reading Cookies (continued)

- Use multidimensional array syntax to read each cookie value

```
setcookie("professional[0]", "Don");
setcookie("professional[1]", "Gosselin");
setcookie("professional[2]", "writer");
if (isset($_COOKIE['professional']))
     echo "{$_COOKIE['professional'][0]}
          {$_COOKIE['professional'][1]} is a
          {$_COOKIE['professional'][2]}.";
```

Deleting Cookies

- To delete a persistent cookie before the time assigned to the `expires` argument elapses, assign a new expiration value that is sometime in the past
- Do this by subtracting any number of seconds from the `time()` function

```
   setcookie("firstName", "",
time()-3600);
   setcookie("lastName", "",
time()-3600);
   setcookie("occupation", "",
time()-3600);
```

Using Sessions to Save State Information

- **Spyware** gathers user information from a local computer for marketing and advertising purposes without the user's knowledge
- A **session** refers to a period of activity when a PHP script stores state information on a Web server
- Sessions allow you to maintain state information even when clients disable cookies in their Web browsers

Starting a Session

- The `session_start()` function starts a new session or continues an existing one
- The `session_start()` function generates a unique session ID to identify the session
- A **session ID** is a random alphanumeric string that looks something like:

  `7f39d7dd020773f115d753c71290e11f`

- The `session_start()` function creates a text file on the Web server that is the same name as the session ID, preceded by `sess_`

Starting a Session (continued)

- Session ID text files are stored in the Web server directory specified by the `session.save_path` directive in your php.ini configuration file
- The `session_start()` function does not accept any arguments, nor does it return a value that you can use in your script

```
<?php
session_start();
...
```

Starting a Session (continued)

- You must call the `session_start()` function before you send the Web browser any output
- If a client's Web browser is configured to accept cookies, the session ID is assigned to a temporary cookie named `PHPSESSID`
- Pass the session ID as a query string or hidden form field to any Web pages that are called as part of the current session

Starting a Session (continued)

```
<?php
session_start();
...
?>
<p><a href='<?php echo "Occupation.php?PHPSESSID="
     . session_id() ?>'>Occupation</a></p>
```

Working with Session Variables

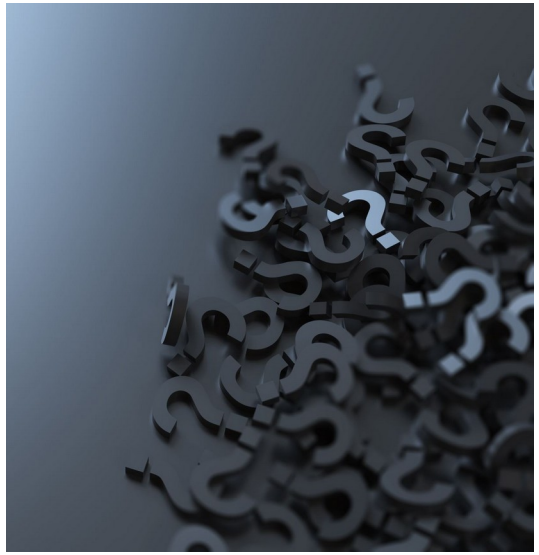
- Session state information is stored in the `$_SESSION` autoglobal
- When the `session_start()` function is called, PHP either initializes a new `$_SESSION` autoglobal or retrieves any variables for the current session (based on the session ID) into the `$_SESSION` autoglobal

Working with Session Variables (continued)

```
<?php
session_start();
$_SESSION['firstName'] = "Don";
$_SESSION['lastName'] = "Gosselin";
$_SESSION['occupation'] = "writer";
?>
<p><a href='<?php echo "Occupation.php?"
 . session_id() ?>'>Occupation</a></p>
```

Working with Session Variables (continued)

- Use the `isset()` function to ensure that a session variable is set before you attempt to use it



```
<?php
session_start();
if (isset($_SESSION['firstName']) &&
  isset($_SESSION['lastName'])
  && isset($_SESSION['occupation']))
  echo "<p>" . $_SESSION['firstName'] . " "
  . $_SESSION['lastName'] . " is a "
  . $_SESSION['occupation'] . "</p>";
?>
```

Deleting a Session

- To delete a session manually, perform the following steps:
  1. Execute the `session_start()` function
  2. Use the `array()` construct to reinitialize the `$_SESSION` autoglobal
  3. Use the `session_destroy()` function to delete the session

Deleting a Session (continued)

```
<?php
session_start();
$_SESSION = array();
session_destroy();
?>
```