What is a Cookie?

A cookie is often used to identify a user. A cookie is a small file that the server embeds on the user's computer. Each time the same computer requests a page with a browser, it will send the cookie too. With PHP, you can both create and retrieve cookie values.

Create Cookies With PHP

A cookie is created with the `setcookie()` function.

Syntax

```
setcookie(name, value, expire, path, domain, secure, httponly);
```

Only the *name* parameter is required. All other parameters are optional.

PHP Create/Retrieve a Cookie

The following example creates a cookie named "user" with the value "John Doe". The cookie will expire after 30 days (86400 * 30). The "/" means that the cookie is available in entire website (otherwise, select the directory you prefer).

We then retrieve the value of the cookie "user" (using the global variable $_COOKIE). We also use the `isset()` function to find out if the cookie is set:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "John Doe";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/"); //
86400 = 1 day
?>
<html>
<body>
<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
```

```
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Note: The `setcookie()` function must appear BEFORE the <html> tag.

Note: The value of the cookie is automatically URLencoded when sending the cookie, and automatically decoded when received (to prevent URLencoding, use `setrawcookie()` instead).

Modify a Cookie Value

To modify a cookie, just set (again) the cookie using the `setcookie()` function:

Example

```
<?php
$cookie_name = "user";
$cookie_value = "Alex Porter";
setcookie($cookie_name, $cookie_value, time() + (86400 * 30), "/");
?>
<html>
<body>

<?php
if(!isset($_COOKIE[$cookie_name])) {
  echo "Cookie named '" . $cookie_name . "' is not set!";
} else {
  echo "Cookie '" . $cookie_name . "' is set!<br>";
  echo "Value is: " . $_COOKIE[$cookie_name];
}
?>

</body>
</html>
```

Delete a Cookie

To delete a cookie, use the `setcookie()` function with an expiration date in the past:

Example

```
<?php
// set the expiration date to one hour ago
setcookie("user", "", time() - 3600);
?>
<html>
<body>

<?php
echo "Cookie 'user' is deleted.";
?>

</body>
</html>
```

Check if Cookies are Enabled

The following example creates a small script that checks whether cookies are enabled. First, try to create a test cookie with the `setcookie()` function, then count the $_COOKIE array variable:

Example

```
<?php
setcookie("test_cookie", "test", time() + 3600, '/');
?>
<html>
<body>

<?php
if(count($_COOKIE) > 0) {
  echo "Cookies are enabled.";
} else {
  echo "Cookies are disabled.";
}
?>
</body>
</html>
```

PHP Filters

Validating data = Determine if the data is in proper form.

Sanitizing data = Remove any illegal character from the data.

The PHP Filter Extension

PHP filters are used to validate and sanitize external input.

The PHP filter extension has many of the functions needed for checking user input, and is designed to make data validation easier and quicker.

The `filter_list()` function can be used to list what the PHP filter extension offers:

Example

```
<table>
  <tr>
    <td>Filter Name</td>
    <td>Filter ID</td>
  </tr>
  <?php
  foreach (filter_list() as $id =>$filter) {
    echo '<tr><td>' . $filter . '</td><td>' . filter_id($filter)
. '</td></tr>';
  }
  ?>
</table>
```

Why Use Filters?

Many web applications receive external input. External input/data can be:

- User input from a form
- Cookies
- Web services data
- Server variables
- Database query results

You should always validate external data!
Invalid submitted data can lead to security problems and break your webpage!
By using PHP filters you can be sure your application gets the correct input!

PHP filter_var() Function

The `filter_var()` function both validate and sanitize data.

The `filter_var()` function filters a single variable with a specified filter. It takes two pieces of data:

- The variable you want to check
- The type of check to use

Sanitize a String

The following example uses the `filter_var()` function to remove all HTML tags from a string:

Example

```php
<?php
$str = "<h1>Hello World!</h1>";
$newstr = filter_var($str, FILTER_SANITIZE_STRING);
echo $newstr;
?>
```

Validate an Integer

The following example uses the `filter_var()` function to check if the variable $int is an integer. If $int is an integer, the output of the code below will be: "Integer is valid". If $int is not an integer, the output will be: "Integer is not valid":

Example

```php
<?php
$int = 100;

if (!filter_var($int, FILTER_VALIDATE_INT) === false) {
  echo("Integer is valid");
```

```
} else {
  echo("Integer is not valid");
}
?>
```

Tip: filter_var() and Problem With 0

In the example above, if $int was set to 0, the function above will return "Integer is not valid". To solve this problem, use the code below:

Example

```
<?php
$int = 0;

if (filter_var($int, FILTER_VALIDATE_INT) === 0 || !filter_var($int,
FILTER_VALIDATE_INT) === false) {
  echo("Integer is valid");
} else {
  echo("Integer is not valid");
}
?>
```

Validate an IP Address

The following example uses the `filter_var()` function to check if the variable $ip is a valid IP address:

Example

```
<?php
$ip = "127.0.0.1";

if (!filter_var($ip, FILTER_VALIDATE_IP) === false) {
  echo("$ip is a valid IP address");
} else {
  echo("$ip is not a valid IP address");
}
?>
```

Sanitize and Validate an Email Address

The following example uses the `filter_var()` function to first remove all illegal characters from the $email variable, then check if it is a valid email address:

Example

```
<?php
$email = "john.doe@example.com";

// Remove all illegal characters from email
$email = filter_var($email, FILTER_SANITIZE_EMAIL);

// Validate e-mail
if (!filter_var($email, FILTER_VALIDATE_EMAIL) === false) {
  echo("$email is a valid email address");
} else {
  echo("$email is not a valid email address");
}
```

Sanitize and Validate a URL

The following example uses the `filter_var()` function to first remove all illegal characters from a URL, then check if $url is a valid URL:

Example

```
<?php
$url = "https://www.w3schools.com";

// Remove all illegal characters from a url
$url = filter_var($url, FILTER_SANITIZE_URL);

// Validate url
if (!filter_var($url, FILTER_VALIDATE_URL) === false) {
  echo("$url is a valid URL");
} else {
  echo("$url is not a valid URL");
}
?>
```

Callback Functions

A callback function (often referred to as just "callback") is a function which is passed as an argument into another function.

Any existing function can be used as a callback function. To use a function as a callback function, pass a string containing the name of the function as the argument of another function:

Example

Pass a callback to PHP's `array_map()` function to calculate the length of every string in an array:

```
<?php
function my_callback($item) {
  return strlen($item);
}

$strings = ["apple", "orange", "banana", "coconut"];
$lengths = array_map("my_callback", $strings);
print_r($lengths);
?>
```

Starting with version 7, PHP can pass anonymous functions as callback functions:

Example

Use an anonymous function as a callback for PHP's `array_map()` function:

```
<?php
$strings = ["apple", "orange", "banana", "coconut"];
$lengths = array_map( function($item) { return strlen($item); } ,
$strings);
print_r($lengths);
?>
```

Callbacks in User Defined Functions

User-defined functions and methods can also take callback functions as arguments. To use callback functions inside a user-defined function or method, call it by adding parentheses to the variable and pass arguments as with normal functions:

Example

Run a callback from a user-defined function:

```
<?php
function exclaim($str) {
  return $str . "! ";
```

```
}

function ask($str) {
  return $str . "? ";
}

function printFormatted($str, $format) {
  // Calling the $format callback function
  echo $format($str);
}

// Pass "exclaim" and "ask" as callback functions to printFormatted()
printFormatted("Hello world", "exclaim");
printFormatted("Hello world", "ask");
?>
```

What is JSON?

JSON stands for JavaScript Object Notation, and is a syntax for storing and exchanging data.

Since the JSON format is a text-based format, it can easily be sent to and from a server, and used as a data format by any programming language.

PHP and JSON

PHP has some built-in functions to handle JSON.

First, we will look at the following two functions:

- `json_encode()`
- `json_decode()`

PHP - json_encode()

The `json_encode()` function is used to encode a value to JSON format.

Example

This example shows how to encode an associative array into a JSON object:

```
<?php
$age = array("Peter"=>35, "Ben"=>37, "Joe"=>43);

echo json_encode($age);
?>
```

Example

This example shows how to encode an indexed array into a JSON array:

```
<?php
$cars = array("Volvo", "BMW", "Toyota");

echo json_encode($cars);
?>
```

PHP - json_decode()

The `json_decode()` function is used to decode a JSON object into a PHP object or an associative array.

Example

This example decodes JSON data into a PHP object:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

var_dump(json_decode($jsonobj));
?>
```

The `json_decode()` function returns an object by default.
The `json_decode()` function has a second parameter, and when set to true, JSON objects are decoded into associative arrays.

Example

This example decodes JSON data into a PHP associative array:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';
```

```
var_dump(json_decode($jsonobj, true));
?>
```

PHP - Accessing the Decoded Values

Here are two examples of how to access the decoded values from an object and from an associative array:

Example

This example shows how to access the values from a PHP object:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$obj = json_decode($jsonobj);

echo $obj->Peter;
echo $obj->Ben;
echo $obj->Joe;
?>
```

Example

This example shows how to access the values from a PHP associative array:

```
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$arr = json_decode($jsonobj, true);

echo $arr["Peter"];
echo $arr["Ben"];
echo $arr["Joe"];
?>
```

PHP - Looping Through the Values

You can also loop through the values with a `foreach()` loop:

Example

This example shows how to loop through the values of a PHP object:

```php
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$obj = json_decode($jsonobj);

foreach($obj as $key => $value) {
  echo $key . " => " . $value . "<br>";
}
?>
```

Example

This example shows how to loop through the values of a PHP associative array:

```php
<?php
$jsonobj = '{"Peter":35,"Ben":37,"Joe":43}';

$arr = json_decode($jsonobj, true);

foreach($arr as $key => $value) {
  echo $key . " => " . $value . "<br>";
}
?>
```