Tishk International University
Faculty of Applied Science
Information Technology Department

Form Validation, Cookies, and Sessions

Lecture 7

Fall 2025

Course Code: IT349

Grade 3

Islam Abdulazeez
islam.abdulaziz@tiu.edu.iq

November 25, 2025

Web Programming

Outlines

- Introduction to Form Validation
- Common Validation Functions in PHP
- HTTP and Statelessness
- Cookies
- Sessions
- Comparison: Cookies vs. Sessions

Learning Outcomes

- **At the end of today’s session, you will be able to:**

  - ✓ Define form validation, cookies, and sessions in PHP.
  - ✓ Differentiate client-side and server-side validation methods.
  - ✓ Apply PHP functions to validate user input and manage data.
  - ✓ Implement cookies and sessions to maintain user state.

What is Validation?

- Validation in PHP is the process of checking and verifying that the data entered by a user (usually through a form) is **correct**, **complete**, and **secure** before it's processed or stored (e.g., in a database).

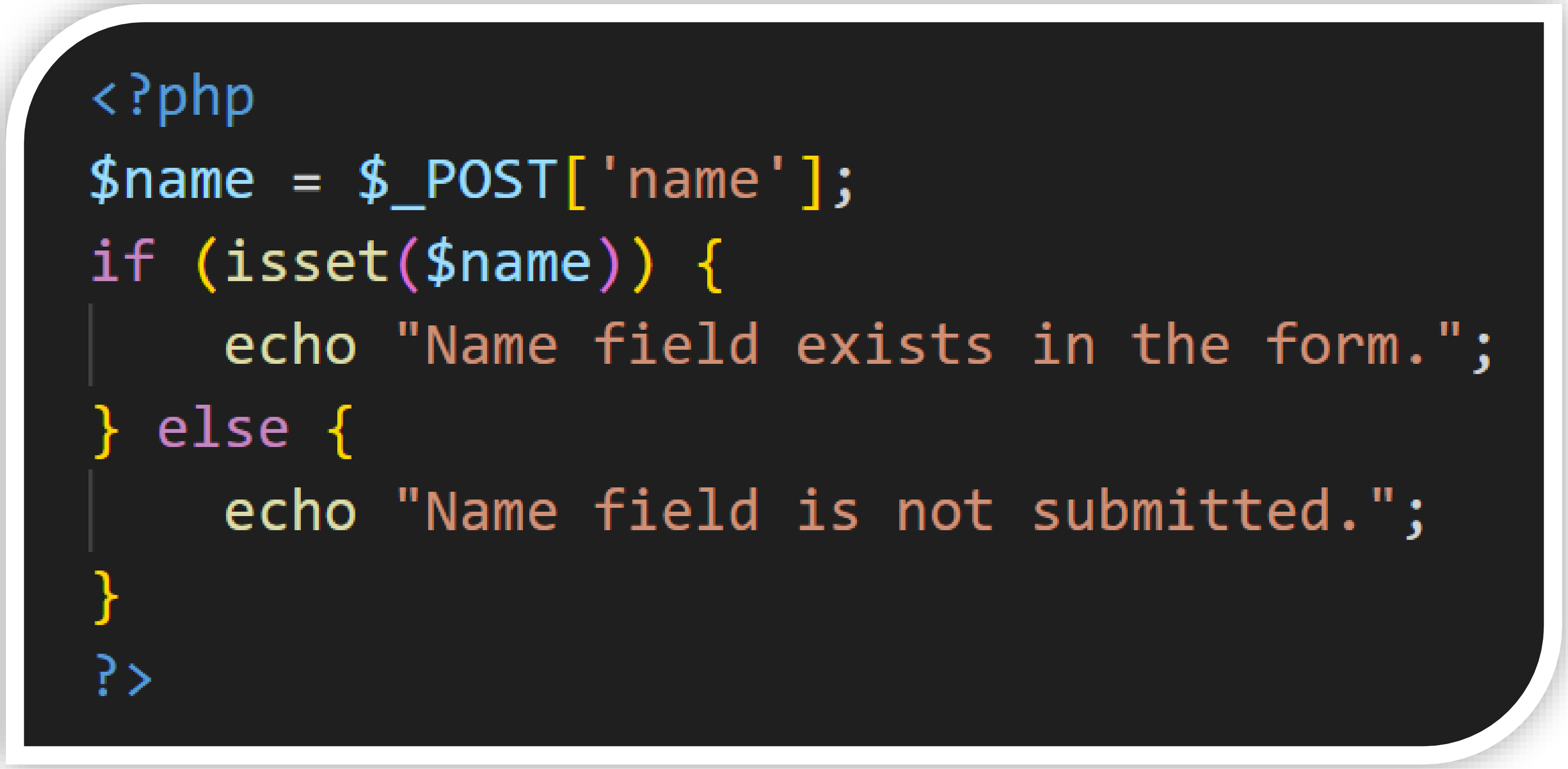
- **Types of Validation**

1. Client-Side Validation: It's done using HTML or JavaScript before sending data to the server (like JavaScript checks).

2. Server-Side Validation: It's done in PHP after the data is submitted. This is more secure because it can't be bypassed.

Validations

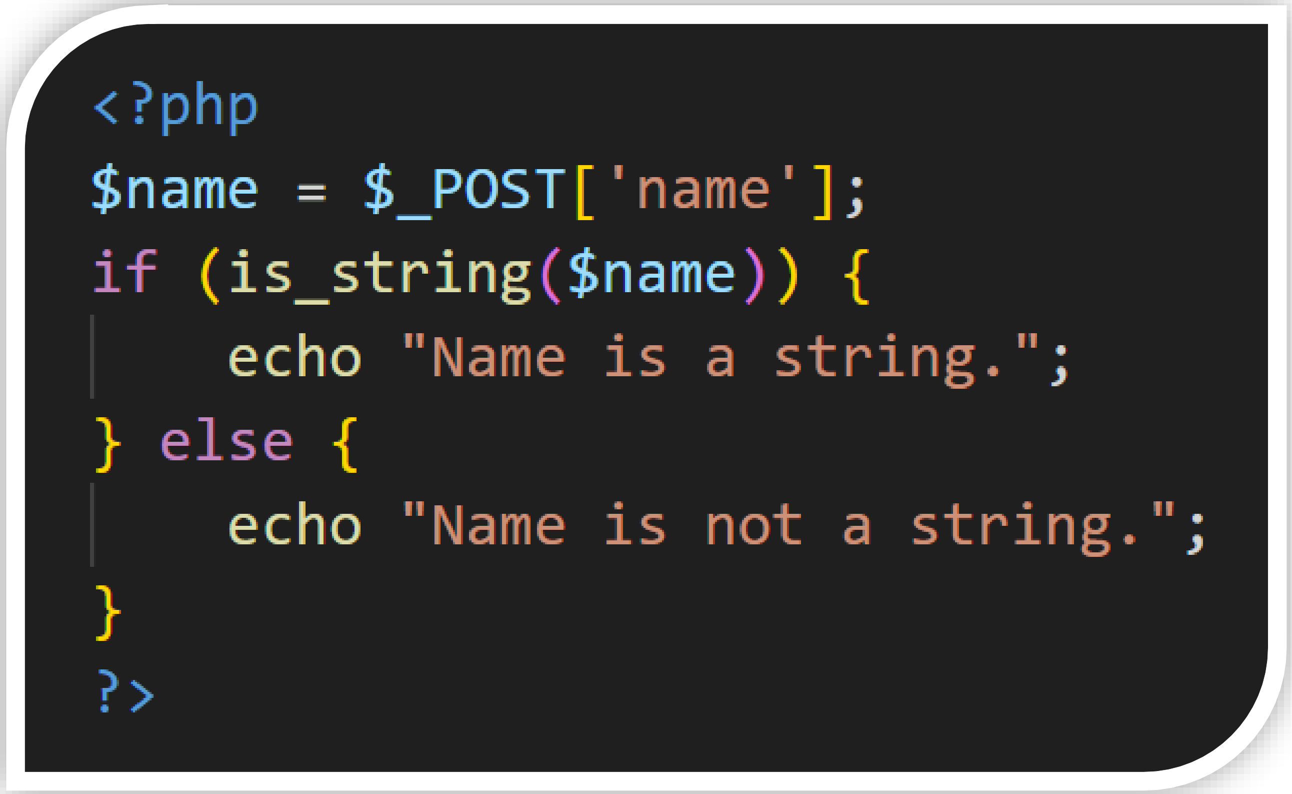
- isset() – Checks whether a variable is declared and its value is not NULL.



```php
<?php
$name = $_POST['name'];
if (isset($name)) {
    echo "Name field exists in the form.";
} else {
    echo "Name field is not submitted.";
}
?>
```

Validations

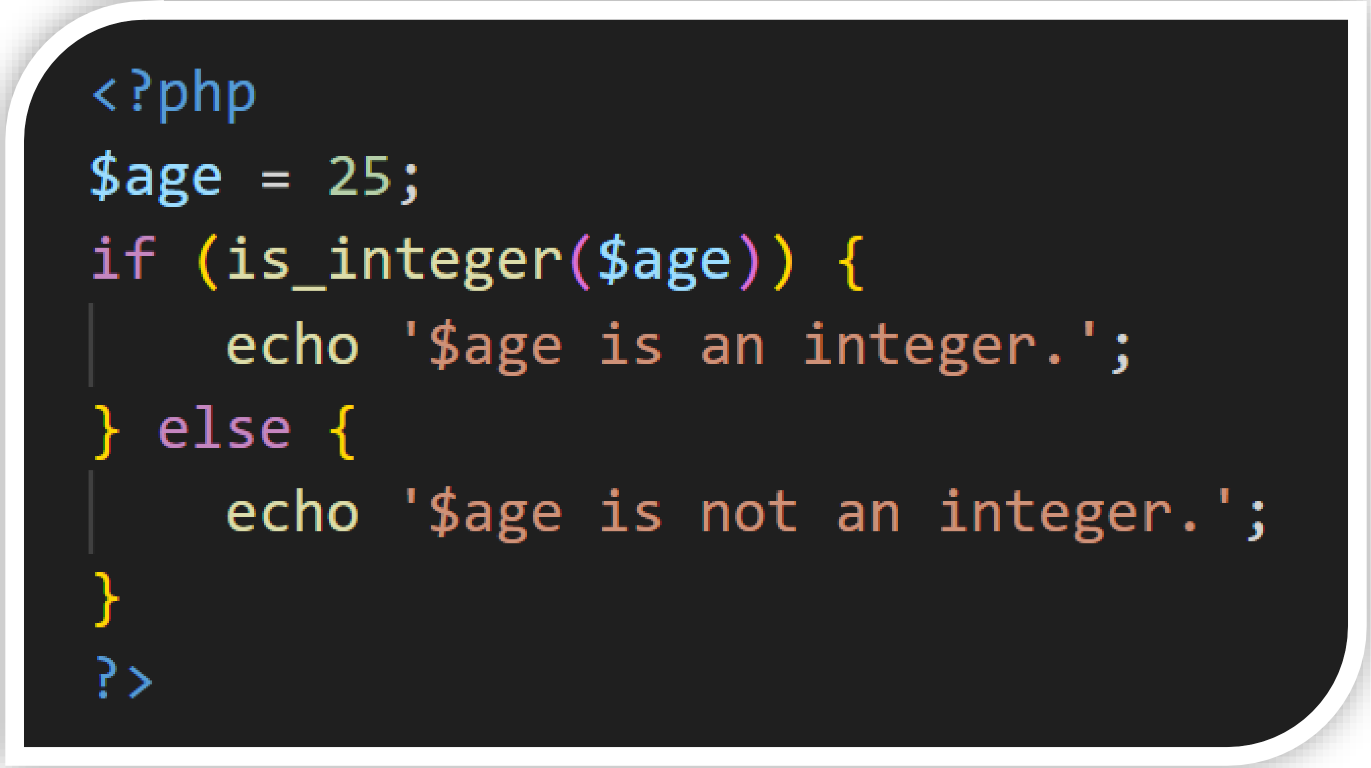
- is_string( ) – Checks whether a variable is of type string or not.



```php
<?php
$name = $_POST['name'];
if (is_string($name)) {
    echo "Name is a string.";
} else {
    echo "Name is not a string.";
}
?>
```

Validations

- is_integer( ) – Checks whether a variable is an integer.

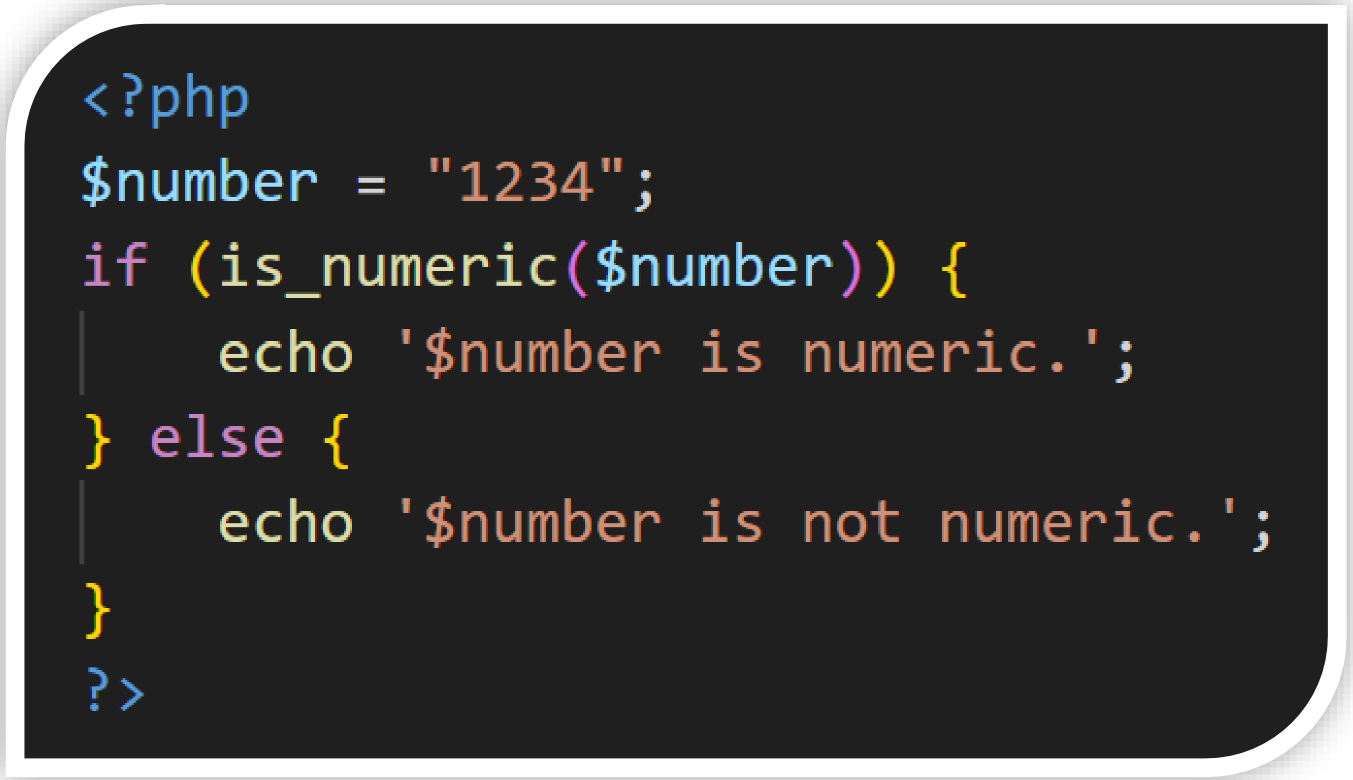


```php
<?php
$age = 25;
if (is_integer($age)) {
    echo '$age is an integer.';
} else {
    echo '$age is not an integer.';
}
?>
```

// $age is an integer.

Validations

- is_numeric( ) – Checks whether a variable is a number or a numeric string, or not.

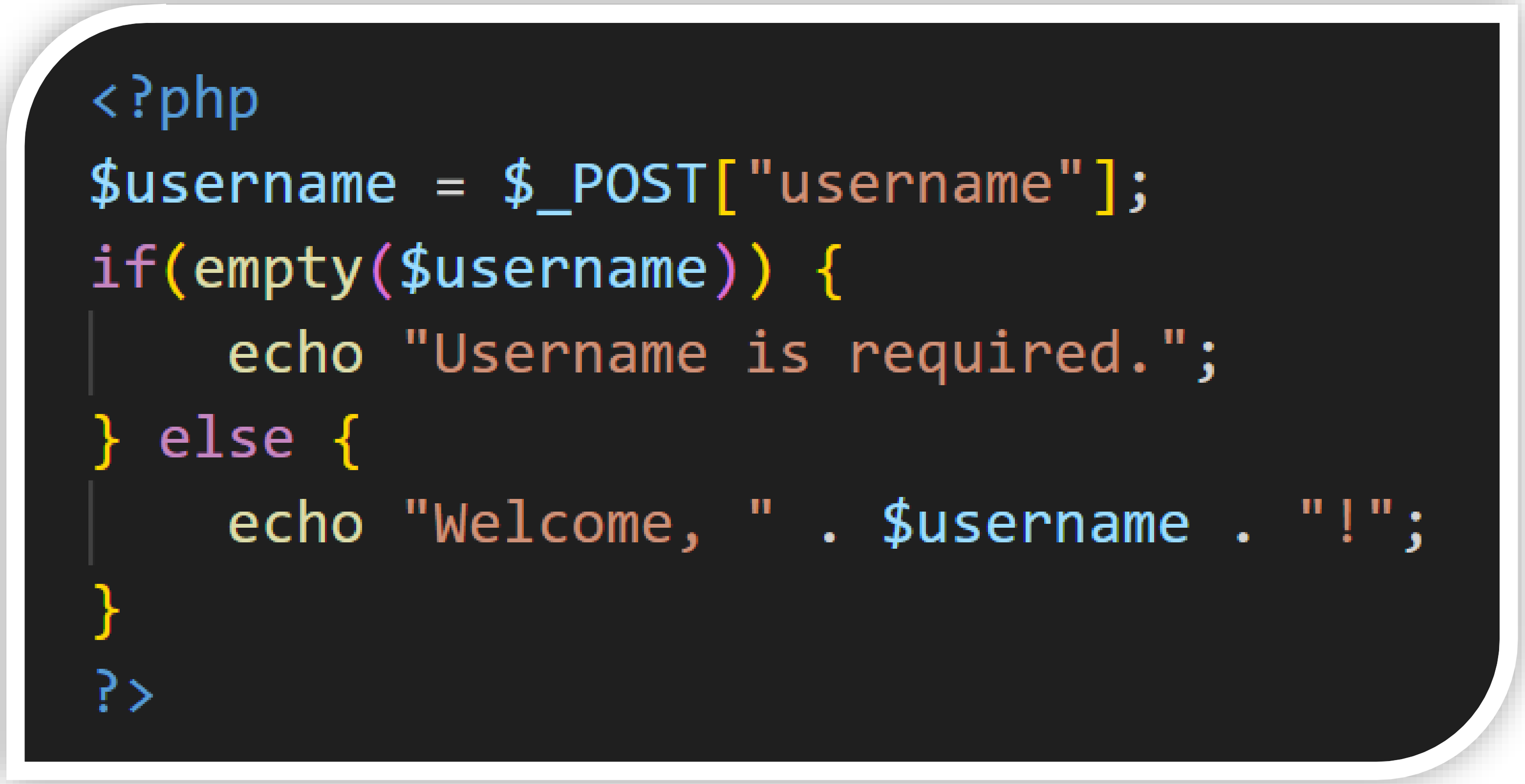


```
<?php
$number = "1234";
if (is_numeric($number)) {
    echo '$number is numeric.';
} else {
    echo '$number is not numeric.';
}
?>
```

// $number is numeric.

Validations

- empty( ) – Checks whether a variable is empty.



```php
<?php
$username = $_POST["username"];
if(empty($username)) {
    echo "Username is required.";
} else {
    echo "Welcome, " . $username . "!";
}
?>
```

Cookies and Sessions

- The Hypertext Transfer Protocol (HTTP) is a stateless technology, meaning that it has no built-in method for tracking a user or remembering data from one page of an application to the next.
- This is a serious problem, because e-commerce applications, user registration and login systems, and other common online services rely on being able to follow the same user from page to page.

Cookies and Sessions

Fortunately, maintaining state is quite simple with PHP. In this lecture we will discusses the two primary methods for tracking data: **cookies** and **sessions**. You'll start by learning how to **create**, **read**, **modify**, and **delete** cookies. Then you'll see how easy it is to master sessions, a more potent option for maintaining state

What Are Cookies

Cookies are small pieces of data that a website stores in **your browser (computer)** so it can remember information about you.

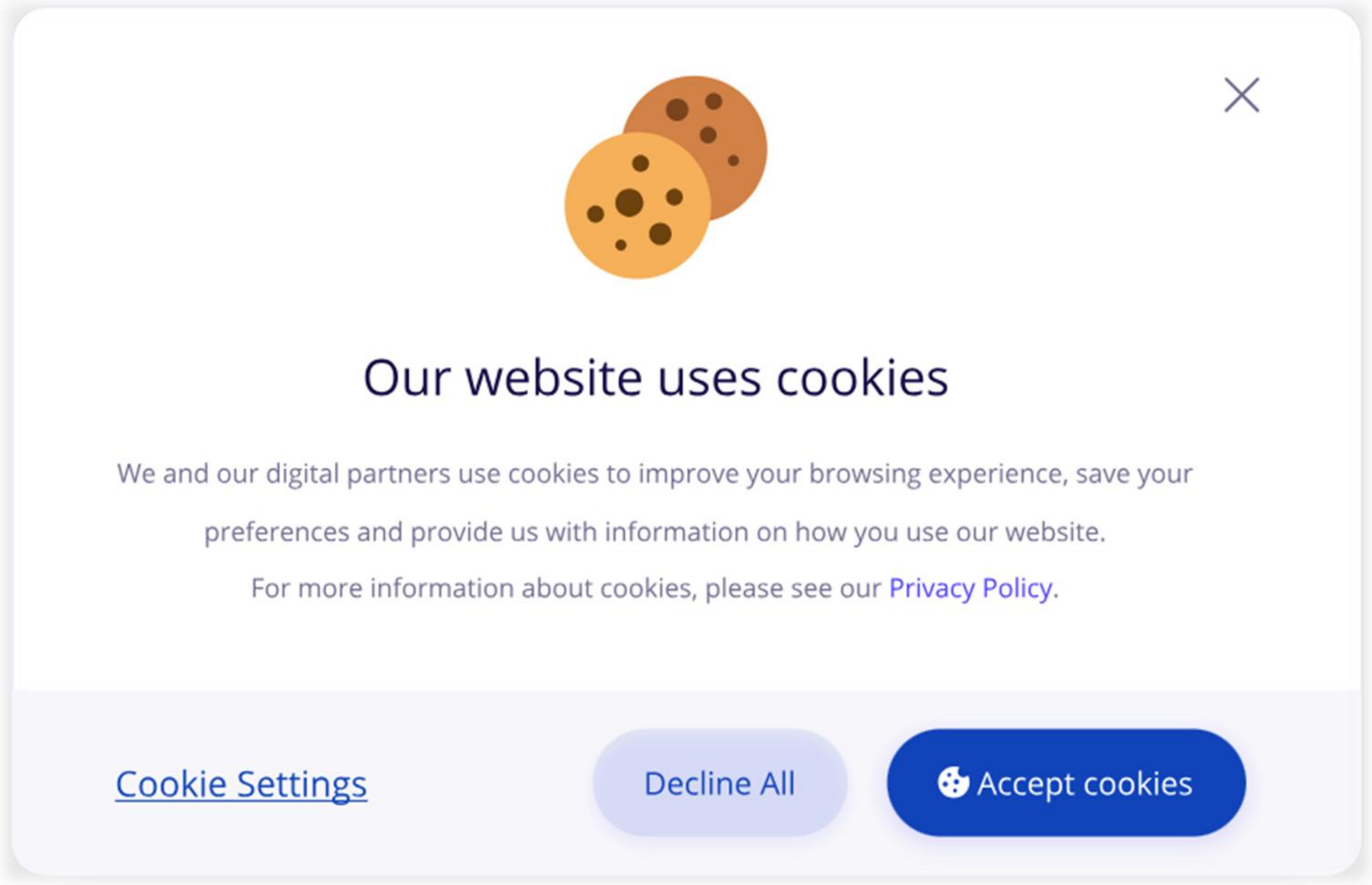
Although the browser tracks the pages you visit, allowing you to use the Back button to return to previously visited pages and indicating visited links in a different color, the server does not follow what individual users see and do.

What Are Cookies

Example:

When you log in to a website, a cookie helps the website remember that you're logged in so you don’t have to sign in again on every page.

What Are Cookies


Our website uses cookies

We and our digital partners use cookies to improve your browsing experience, save your preferences and provide us with information on how you use our website.
For more information about cookies, please see our Privacy Policy.

What Are Cookies

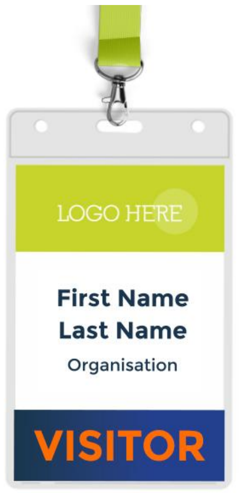
The Importance of Tracking Users

If the server can’t track a user, there can be no shopping carts for making purchases online. If cookies didn’t exist (or if they’re disabled in the browser), people wouldn’t be able to use popular sites that require user registration. In short, **without cookies, there would be no Amazon or Facebook or any of the other most popular or useful sites**.

What Are Cookies

Cookies are simply a way for a server to store information on the user's computer. By doing so, the server can remember the user over the course of a visit or through several visits.

Think of a cookie as a name tag: You tell the server your name, and it gives you a name tag. Then it can know who you are by referring to the name tag.

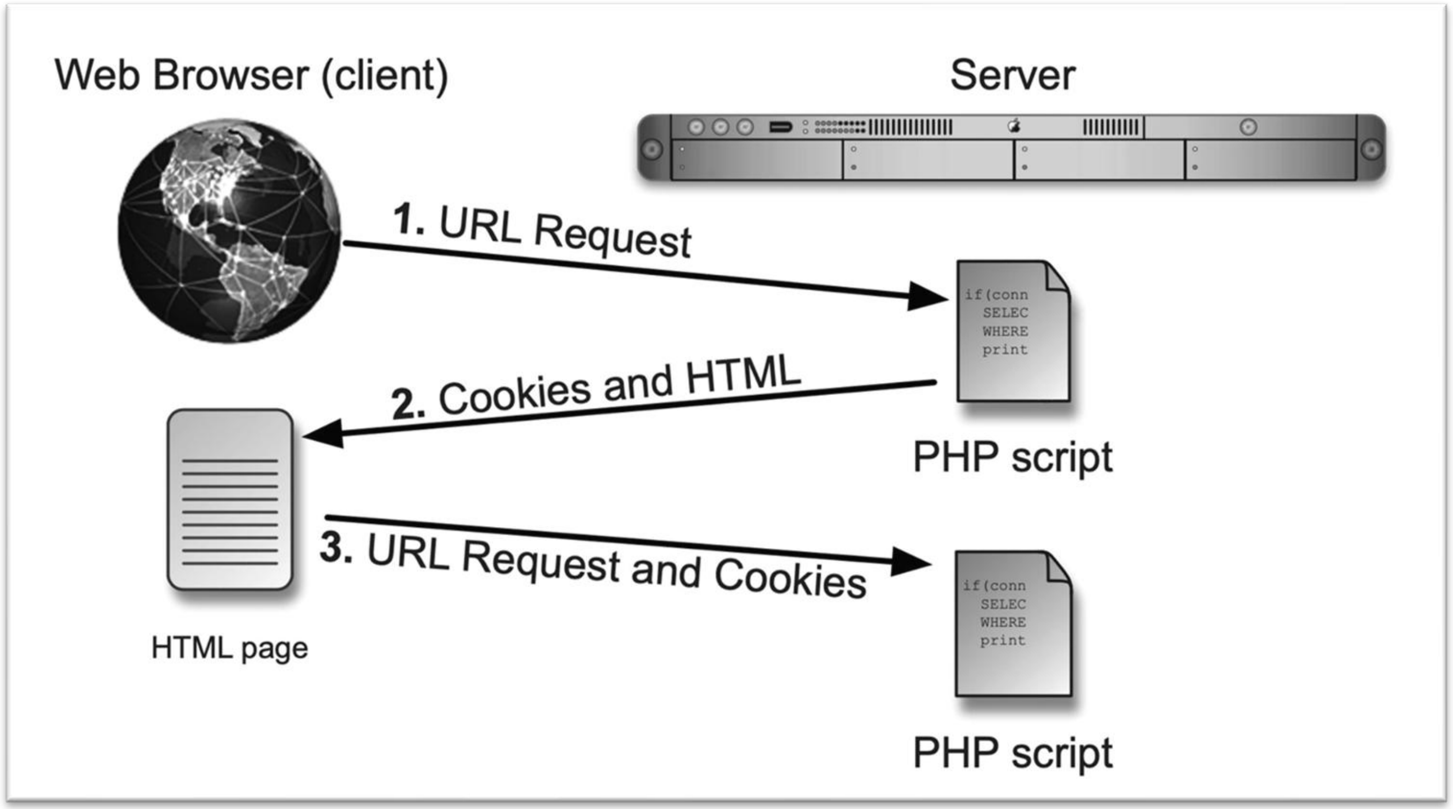


What Are Cookies

- Cookies are stored by the browser, so technically the browser has access to them.
- Only the website (domain) that set the cookie can read it automatically when you visit that site again. Other websites cannot access cookies set by a different site. This is called the same-origin policy.
- So, your browser doesn’t send your personal info to other sites, only to the site that created the cookie.

What Are Cookies

- How cookies are sent back and forth between the server and the client.



Setting Cookies

Cookies are created with the **setcookie()** function.

```
<?php
setcookie(name, value);
?>
```

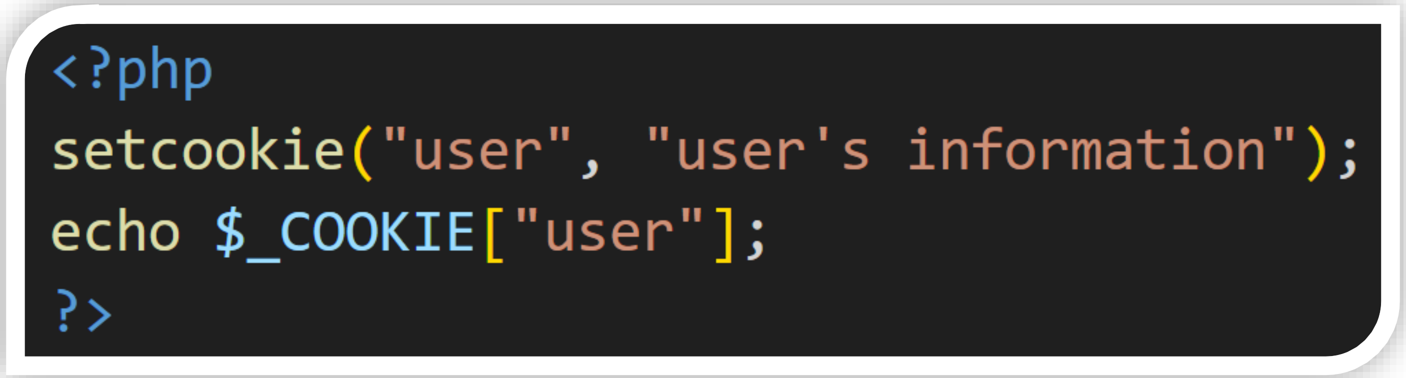
```
<?php
setcookie('Cookie Name', 'This is a cookie value');
?>
```

That line of code sends to the browser a cookie with the name **CookieName** and the value **This is the cookie value**.

Reading From Cookies

Just as form data is stored in the $_POST array and values passed to a script in the URL are stored in the $_GET array, the setcookie() function places cookie data in the $_COOKIE array

To retrieve a value from a cookie, you only need to refer to the cookie name as the index of this array,



```php
<?php
setcookie("user", "user's information");
echo $_COOKIE["user"];
?>
```

Reading From Cookies

Passing just the **name** and **value** to the setcookie() function is sufficient for most uses, but the function also accepts **up to five additional parameters** that control the cookie’s behavior.

```
<?php
setcookie(name, value, expire, path, domain, secure, httponly);
?>
```

Explanation of Parameters

name → Name of the cookie

value → Value of the cookie

expire → Expiration timestamp

path → Path where the cookie is available

domain → Domain where the cookie is available

secure → Only send over HTTPS

httponly → Not accessible via JavaScript

Deleting a Cookie

- The final thing to know about cookies is how to delete them. Although a cookie automatically expires when the expiration date/time is met.
- Sending a cookie with the same name and an expired timestamp will delete the existing cookie.

Deleting a Cookie

For example, to create the cookie username, you write the following code:

```
<?php   setcookie('username', 'user’s info');   ?>
```

To delete the username cookie, you write the following code:

```
<?php   setcookie('username', '', time() - 3600); ?>
```

To delete a cookie, always set its expiration in the past.

What Are Sessions

A session, like a cookie, provides a way to track data for a user over a series of pages. The key difference is that a **cookie stores data on the client** (in the browser), whereas **session data is stored on the server**.

What Are Sessions

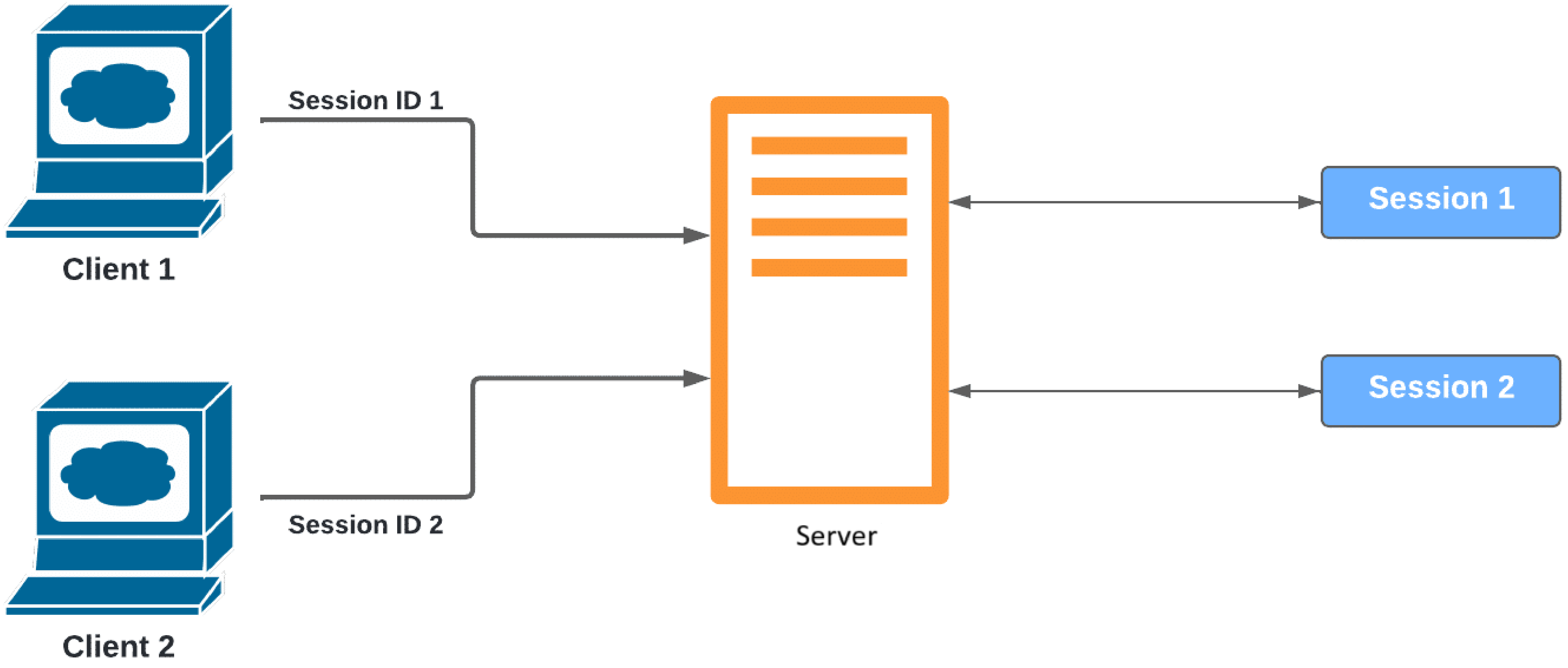
Because of that difference, **sessions** have numerous benefits over cookies:

- Sessions are generally more secure, because the data isn’t transmitted back and forth between the client and server repeatedly.
- Sessions allow you to store more information than you can in a cookie.
- Sessions can still work if the user does not accept cookies, by passing the session ID in the URL.
- You can more easily store other types of data in sessions, such as arrays and Booleans.

What Are Sessions

- When a session starts, PHP creates a unique session ID.
- That session ID identifies a file on the server that stores session data.
- Each user gets their own session ID.
- To keep the session consistent across pages, PHP needs to know the session ID every time.
- By default, PHP sends the session ID to the browser using a cookie, so the browser can send it back on each request.

What Are Sessions



Creating a Session

- Creating, accessing, or deleting a session begins with the session_start() function.
- This function will attempt to send a cookie the first time a session is started, so it must be called prior to any HTML or whitespace being sent to the browser. Therefore, it should always come before any output.

Creating a Session

- On pages that use sessions, you should call the session_start() function as one of the very first lines in your script:

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

- Once the session has been started, you can record data to it by assigning values to the $_SESSION array:

```
<?php
            $_SESSION['first_name'] = 'Aso' ;
            $_SESSION['age'] = 21 ;
?>
```

Creating a Session

- Unlike other arrays you might use in PHP, you should always treat this array as an **associative array**. In other words, you should explicitly use strings for the keys, such as first_name and age.

```php
<?php
            $_SESSION['first_name'] = 'Aso' ;
            $_SESSION['age'] = 21 ;
?>
```

- Each time a value is assigned to the $_SESSION array, PHP writes that data to a temporary file stored on the server.

Accessing Session Variables

- The first step is to invoke the session_start( ) function. This is necessary on every page that will make use of sessions, whether it's creating a new session or accessing an existing one.
- To access the stored user's sessions, refer to $_SESSION[] array.

```
<?php
        echo  $_SESSION['first_name'] ;
?>
```

Deleting a Session

- It’s important to know how to delete a session, just as it’s important to know how to delete a cookie: Eventually you’ll want to get rid of the data you’ve stored.
- During script execution, session data exists in two places: the $_SESSION array in memory and the session file on the server.
- You can clear the session variables by resetting the $_SESSION array.

```
<?php    $_SESSION = [ ] ;    ?>
```

Deleting a Session

- Finally, remove the session data from the server (where it’s stored in temporary files). To do this use:

```
<?php    session_destroy( );    ?>
```

Comparison

- Comparing cookies and sessions in PHP

Feature	Cookies	Sessions
Storage location	Browser	Server
Lifetime	Can persist across browser sessions	Usually until the browser closes
Security	Less secure	More secure
Size	Limited (~4KB per cookie)	Can store larger amounts of data (depends on server)
Accessibility	Client & Server	Server only
Use	Preferences, tracking	Sensitive data, login status

Lab Assesments and Next Session’s Topic

Lab Assessments

- Lab Exercises.
- Quiz 3 (Practical) from Lecture Note 5.

Next Session’s Topic

- Database Connection & Table Creation in PHP

References

- Tatroe, K., & MacIntyre, P. (2020). Programming PHP: Creating dynamic web pages (4th ed.). O’Reilly Media.
- Ullman, L. (2016). PHP for the web: Visual QuickStart guide (5th ed.). Peachpit Press.
- PHP Documentation. (n.d.). PHP.net. Retrieved November 8, 2025, from https://www.php.net/docs.php

Thank You!