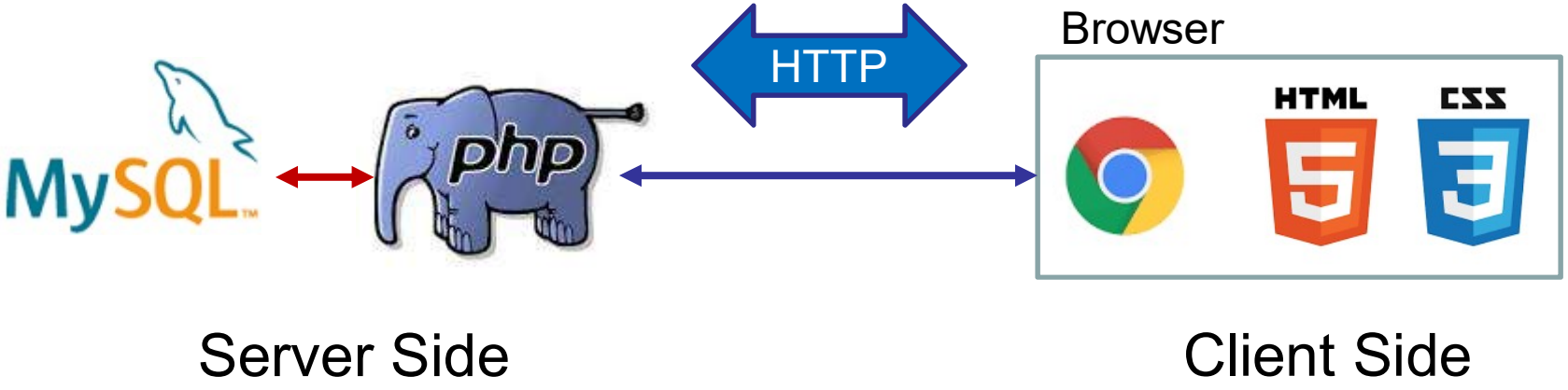
CSCU9B3
MySQL and PHP

Programming Databases

- Scripts in PHP to allow data to be manipulated on the server
- This allows dynamic content for web pages
- What if the data is in a database?
- Your PHP script can access the data in the SQL database and use it to create content in a page



MySQL and PHP

- Many database driven web sites use PHP and MySQL
- It is simple, quite easy, and universally supported by web hosting companies
- All the software involved is free too
- You can install it all on your own computer
  - WAMP for windows and MAMP for Apple
  - www.wampserver.com

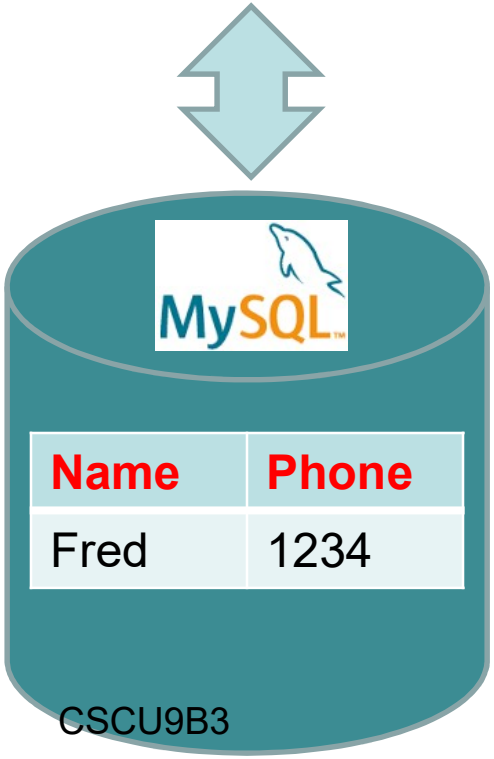
Server

Client

Gets form data
Reads from database
Creates results page
Sends HTML to browser

Form Contents

HTML Page



Name	Phone
Fred	1234

Search Results

Name	Phone
Fred	1234

Concepts: Server / Client

- The database and any programs that access it are on a **server**
- People who use the database do so via a **client** – that is local software that interacts with the server to make queries
- With web based applications, the client is usually the **web browser**

Client / Server in PHP

- The MySQL database and the PHP code both sit on the server
- The PHP runs, makes queries from the database, and creates HTML
- The HTML is sent to the browser and displayed
- The PHP code is not readable by people using the browser – important security.

Consequences

- Interaction with the database is ‘web page based’
- To run a query, the user requests a new web page either by clicking a link or submitting a form
- (Ajax overcomes this by running Javascript in the client browser to request less than a new page)

Three Step Process

1. Data sent from browser to server in the form of a request to run a .php page
2. The .php program executes. Part of this program makes queries from the database
3. The output (usually HTML) from the program is displayed in the browser

1. Data Sent from Browser

- Web pages send data to the server in a number of ways
  - Data from a form filled in by the user
  - Data stored in a cookie on the user's computer
  - Session data
  - Data appended to the URL: go.php?user=fred
  - The name of the .php file to run

Data from a Form

- You may be familiar with HTML forms:

```
<form action='go.php' method='post'>
<input name='username' type='text'>
</form>
```

Data from a Form

- When the form is submitted, data is sent to the PHP program using the method specified:
  - '**get**' appends the data to the URL: www.example.com/go.php?username=fred
  - '**post**' sends the data in the HTTP request
- Data is accessed in the PHP program using arrays $_GET and $_POST

Data from a URL

- You can mimic the GET method by using links in your HTML with variables set:

```
<a href='go.php?action=Login'>Login</a>

<a href='go.php?action=Join'>Join</a>
```

2. Run PHP

- PHP program connects to database and sends queries using one of the methods shown next
- Results are read from the database by the PHP program
- These results are used to decide what to show in the web browser

3. Produce Output

- The most usual output is for the PHP program to produce HTML to be sent to the browser
- It can also write to the database
- Or carry out other actions based on the results of its processing:
  - Send emails
  - Process orders or money etc.

PHP and MySQL

- There are three main methods of connecting to MySQL from PHP:

1. mysqli_ procedural set of commands
2. mysqli_ object-oriented set of commands
3. Use PHP Data Objects (PDO)

- Let's look at each in turn
  - Though we will use mysqli_

mysqli_

- Connecting and selecting databases
- Running queries
- Stepping through the results of queries
- Example object:

```
$conn = new mysqli($servername,
$username, $password);
```

- Example function:

```
$conn = mysqli_connect($servername,
$username, $password)
```

PHP Data Objects (PDO)

- PDO is a database abstraction layer
- That is to say, it doesn't matter what database you want to connect to (MySQL, or some other), the code stays the same (mostly)
- Advantage: Switch databases with minimal fuss
- Disadvantage: Loses some MySQL specific functionality

Security

- Regardless of how a web page sends data to the PHP program, a malicious hacker could write their own code to send whatever they want to your script
- You can’t prevent this
- All you can do is be very careful at the PHP stage to check what was sent from the form before sending anything to the database

A Simple Worked Example

1. Build a web page to ask for a user's first and second name
2. Run a PHP script to access that data from HTTP Post
3. Use the data to build a MySQL statement to search a table for that name and return the phone number that goes with it
4. Display the phone number in HTML

The Web Form

```
<form method="post" target="go.php">
Enter your first name:
<input type="text" name="firstn">
Enter your second name:
<input type="text" name="secondn">
<input type="submit" name="go"
  value="Go">
</form>
```

Connecting to the Database

- Connect to a data base using *mysqli_connect*
- Need server name, user name, password, and name of database to connect to

```
$d = mysqli_connect($server, $user, $password, $db)
if (!$d) {die("Failed: " . mysqli_connect_error());}
```

- Returns a link to the database to pass to other *mysqli_* calls

Getting the Values in PHP

```
$firstn=$_POST['firstn'];
$secndn=$_POST['secondn'];
```

// Escape quotes e.g. ` to \`:

```
$firstn=mysqli_real_escape_string($d,$firstn);
$secndn=mysqli_real_escape_string($d,$secndn);
```

// Remove malicious HTML tags:

```
$firstn=strip_tags($firstn);
$secndn=strip_tags($secndn);
```

mysqli_ commands

```
$sql = "SELECT phone FROM people
WHERE fname='$firstn'
AND sname='$secondn'";

$result = mysqli_query($d, $sql);
```

Getting the Results

```
if (!$result)
{
   print("$sql produced an error: " .mysql_error());
}
else
{
   $row=mysqli_fetch_row($result);
   if($row!==FALSE)
   {
       $phonenum=$row[0];
       print("Number for $firstn $secondn is $phonenum<br>");
   }
   else
       print("That name cannot be found");
}
```

The Whole Script

```
<?php
$d = mysqli_connect($server, $user, $password, $db);
if (!$d) {die("Failed: " . mysqli_connect_error());}
$firstn=$_POST['firstn'];
$secndn=$_POST['secondn']; // don't forget to sanitise!
$sql = "SELECT phone FROM people WHERE fname='$firstn' AND
   sname='$secondn'";
$result = mysqli_query($d, $sql);
if (!$result) {
   print("$sql produced an error: " .mysql_error());}
else {
   $row=mysqli_fetch_row($result);
   if($row!==FALSE){
       $phonenum=$row[0];
       print("Number for $firstn $secondn is $phonenum<br>");}
   else print("That name cannot be found");
}
mysqli_close($d);
?>
```

Object-oriented Script

```
<?php
$d = new mysqli($server, $user, $password, $db);
if ($d->connect_error) {die("Failed: ".$d->connect_error);}
$firstn=$_POST['firstn'];
$secndn=$_POST['secondn']; // don't forget to sanitise!
$sql = "SELECT phone FROM people WHERE fname='$firstn' AND
   sname='$secondn'";
$result = $d->query($sql);
if ($result->num_rows > 0) {
   $row=mysqli_fetch_row($result);
   while($row = $result->fetch_assoc()){
        $phonenum=$row[0];
        print("Number for $firstn $secondn is $phonenum<br>");
} else
   print("That name cannot be found");
}
$d->close();
?>
```

Inserting Data

- How do we do things other than searching?

```
$sql = "INSERT INTO people (fname, sname, phone)
VALUES (‘Fred', ‘Bloggs', ‘123456')";

if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

OR

```
if (mysqli_query($conn, $sql)) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . mysqli_error($conn);
}
```

Prepared Statements

- Under some circumstances it may be useful to work in three stages
  - prepare a statement using placeholders
  - fill in the blanks
  - execute the statement
- If the SQL is executed repeatedly, it may be more efficient as the preparation can be done outside the loop, once
  - (some DBMSs may do nothing)

Prepared Statements

(example from w3schools)…

```
<?php
// Create connection
$conn = new mysqli($servername, $username, $password,
$dbname);
if ($conn->connect_error) {die($conn->connect_error);}

// prepare and bind
$stmt = $conn->prepare("INSERT INTO MyGuests (firstname,
lastname, email) VALUES (?, ?, ?)");

$stmt->bind_param("sss", $firstname, $lastname, $email);
// set parameters and execute
```

Prepared Statements

```
// set parameters and execute
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();

$stmt->close();
$conn->close();
?>
```

More …

- Find out more about the mysqli functions in PHP here:

  uk.php.net/manual/en/set.mysqlinfo.php

  https://www.w3schools.com/php/php_ref_mysqli.asp