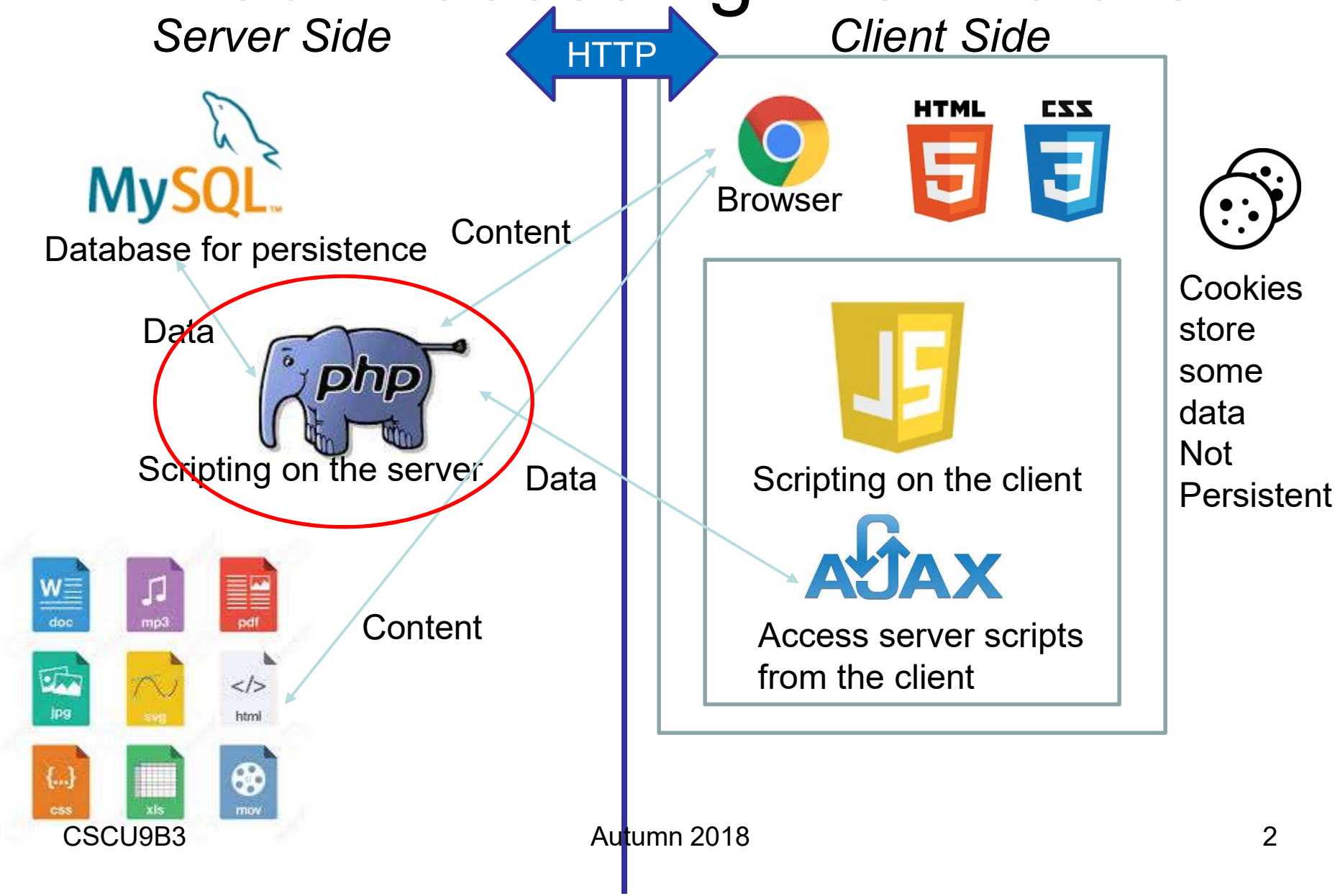
CSCU9B3
An Introduction to PHP

Web Processing in a Picture



Introduction to PHP

- PHP is a scripting language
- PHP is interpreted by the server at run time, so doesn’t need a compiler
- Doesn’t need a development environment like Eclipse
- You can use a simple text editor like Textpad
  - Create files: e.g. mydemo.php

Running a PHP Program

- In a client (e.g. browser)/ server model, the PHP program is run on the server, so no code is downloaded to the client
- Only the output from the PHP program is sent to the client
- The easiest way to run a PHP program and see its output is to put it on a (web) server and open it in a browser

Starting a PHP Program

```
<?php


?>
```

- Anything inside these delimiters is interpreted as php
- Anything outside them is sent straight to the client, usually the browser
  - Typically HTML / CSS / Javascript code

Example 1

```
<HTML>
<head></head>
<body>
<?php
print("This is the output from
  the php program");
?>
</body>
</HTML>
```

PHP Program Structure

- In common with many programming languages, PHP:
  - Ends each line with a semi-colon `;`
  - Uses `{ }` braces to delimit code blocks
- PHP can be used in either a procedural or object-oriented way
- PHP denotes *variables* with the dollar sign: `$a`, for example

Variables

- You assign variables like this:

```
$a=5;
```

- And compare them like this:

```
if($a==5)…
```

- You can use them in strings like this:

```
print("$a is the value of variable a");
```

Variable Types

- PHP is pretty relaxed about variable types
- You do not need to declare variables as being of a certain type, just start using them:

```
$a=5;
$a="Hello";
```

Checking Variables

- You do not have to declare a variable before you use it, but you should
- PHP will issue a warning (but still run) if you access a variable that is undeclared:

```
$a=$b;     // $b never declared
```

- Use `isset($var)` to check if a variable has been set

Arrays

- PHP supports two kinds of array:
  - Numeric indexed, where the index is a number
  - Associative, where the index is a string
- Syntax:
  - `$a[5]=“Hello”;`
  - `$i=5;`
  - `$a[$i]=“Hello”;`
  - `$a[“name”]=“Bob”;`

Objects

- PHP supports objects / classes
- Syntax:

```
class Car {
    function Car() {
        $this->model = "VW";
    }
}
```

Objects (2)

- Creation and usage:

```
// create an object
$herbie = new Car();

// show object properties
echo $herbie->model;
```

Initialise an Array

- It is a good idea to initialise arrays before you use them:

```
$a=array(“cat”,”dog”,”mouse”);
```

or (associative array)

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

Functions

- You declare functions in PHP like this:

```
function myfunc($param1, $param2)
{
   return $param1+$param2;
}
```

- And call them like this:

```
$res = myfunc(1,2);
```

- Parameters are set according to their position
- Return values are optional

Scope and Functions

- Global variables exist outside functions
- Inside functions, variables are local and do not persist after the function is run
- To access global variables in a function:

```
function myfunc()
{
   global $a, $b;
   $a++;
}
```

- Note that this changes the value of $a in the main body of the program, not just the function. Remove the global declaration and $a outside the function would be left untouched.

Pass by Value

- By default, parameters passed to a function are copied ("pass by value"), so

```
$a=5;
myfunc($a);
print($a);
```

- Would display 5 no matter what happened to the parameter $a in the function

Pass by Reference

- If you want a function to alter the value of the variable sent to it in a parameter, you send it by reference:

```
myfunc(&$a)
```

- The `&` tells PHP to send the actual variable, not a copy

Default Parameters

- You can set default values for parameters in function declarations:

```
function myfunc($param1, $param2=0)
```

- Calling like this would result in $param2 being set to 0:

```
myfunc(5);
```

Loop Structures

- for(set;compare;update)

```
for($i=0;$i<10;$i++)
```

- while(condition)

```
$i=0;
while($i<10)
{
   $i++;
}
```

Loop Structures (2)

- do {} while(condition);

```
$i=0;
do
{
   $i++;
}
while($i<10)
```

Iterating an Array

- Values from an array

```
foreach($array as $val)
{
   print("$val<br>");
}
```

- Values and Keys

```
foreach($array as $key => $val)
{
   print("$key is the index to $val<br>");
}
```

Printing Output

- We have already seen

```
print($string);
```

- What ever is printed is sent to the browser. This could be:
  - Raw text
  - HTML
  - JavaScript
  - CSS etc.

Building Strings

- Strings are concatenated with a dot .

```
$a="Hello";
$b=" There";
$c=$a.$b;
```

- `$c` becomes `"Hello There"`
- You can also use `$a.="something";` to tag a string onto the end of `$a`

String Manipulation

- Useful PHP functions for string manipulation include:

```
substr ($string, $start[, $length ])
str_replace ($search, $replace,
$subject[, &$count])
strpos ($haystack, $needle[, int
$offset=0] )
```

System Variables

- There are a number of system variables in PHP that can be very useful
- The two that are of particular interest are:

```
$_POST[]
$_GET[]
```

- These are arrays, indexed by name, of data sent from the browser, usually via a form

Example

- HTML file, in browser

```
<form action=‘go.php’ method=‘post’>
<input name=‘firstname’ type=‘text’>
</form>
```

- PHP script, go.php on server

```
<?php
$n=$_POST[‘firstname’];
print(“Hello $n”);
?>
```

Sessions

- Once a PHP script is run, it is forgotten
- It does not keep running on the server, waiting for interaction from a browser
- To keep an interaction going (e.g. maintain a logged in state), use sessions
- These are a PHP abstraction that stores data in a global _SESSION variable on the server while the session lasts

Use

```
<?php
session_start();
$_SESSION['user']="Fred";
?>
```

Later, perhaps a different PHP script …

```
<?php
session_start();
$name = $_SESSION['user'];
?>
```

Read and Do More

- There are many online PHP resources:
  - php.net, w3schools
- Try it yourself:
  - Put some simple code in a file called test.php in your www wamp0 folder: \\wamp0.cs.stir.ac.uk\www\xxx
  - Point a browser at: http://wamp0.cs.stir.ac.uk/xxx/test.php
  - xxx is your username (as for MySQL)