The Internet

PHP works with the web

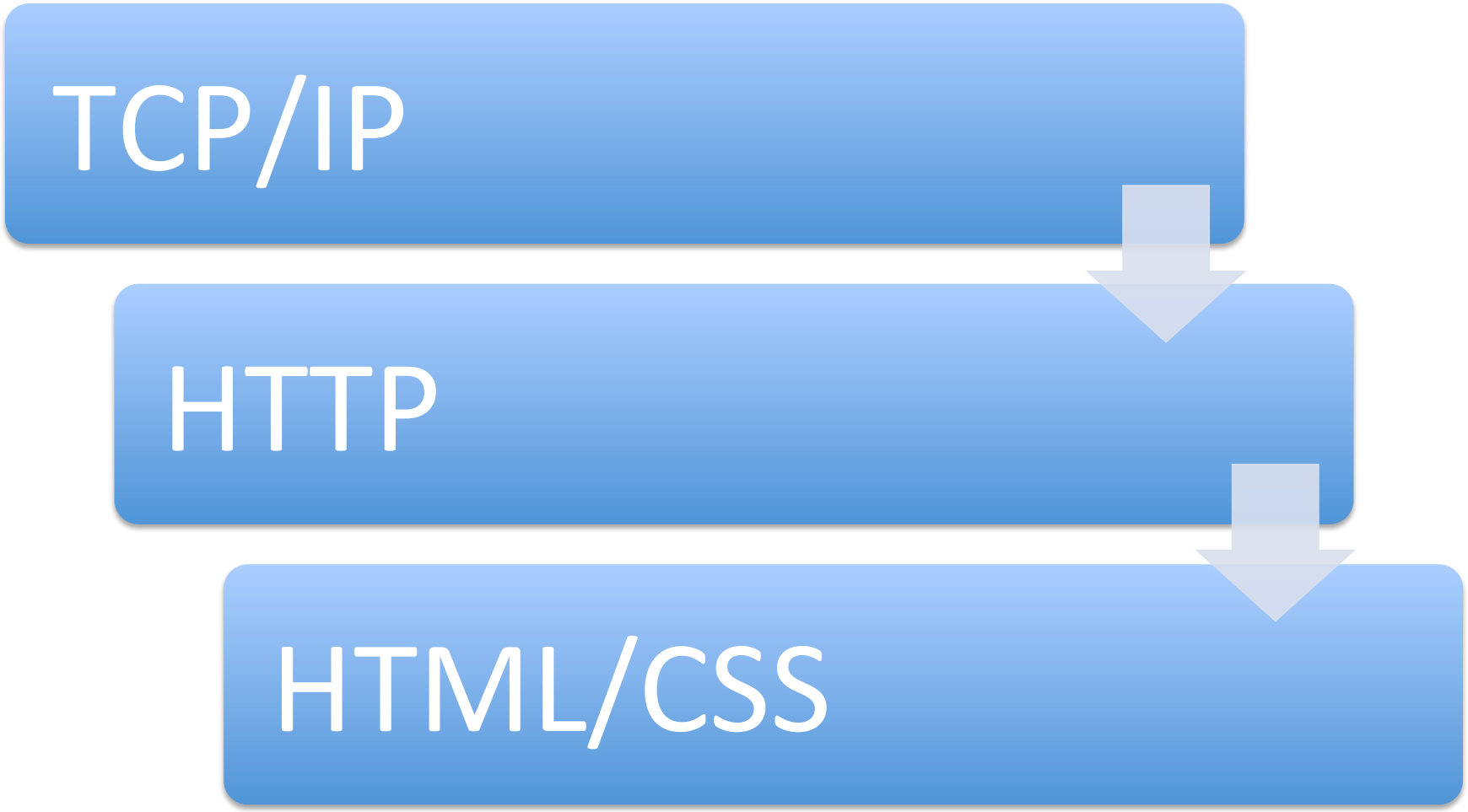
Lesson 1:

INTERNET BASICS

What is the Internet?

- The **Internet** is a global system of interconnected computer networks that use the standard **Internet** protocol suite (TCP/IP) to link several billion devices worldwide.

What runs the Internet



TCP/IP

- Transmission Control Protocol (**TCP**) and the Internet Protocol (**IP**)

IP

- IP address -> where we're going
- IP restricts size – 64K – but doesn't care how it gets chopped up
- IP cares nothing for order or when things arrive

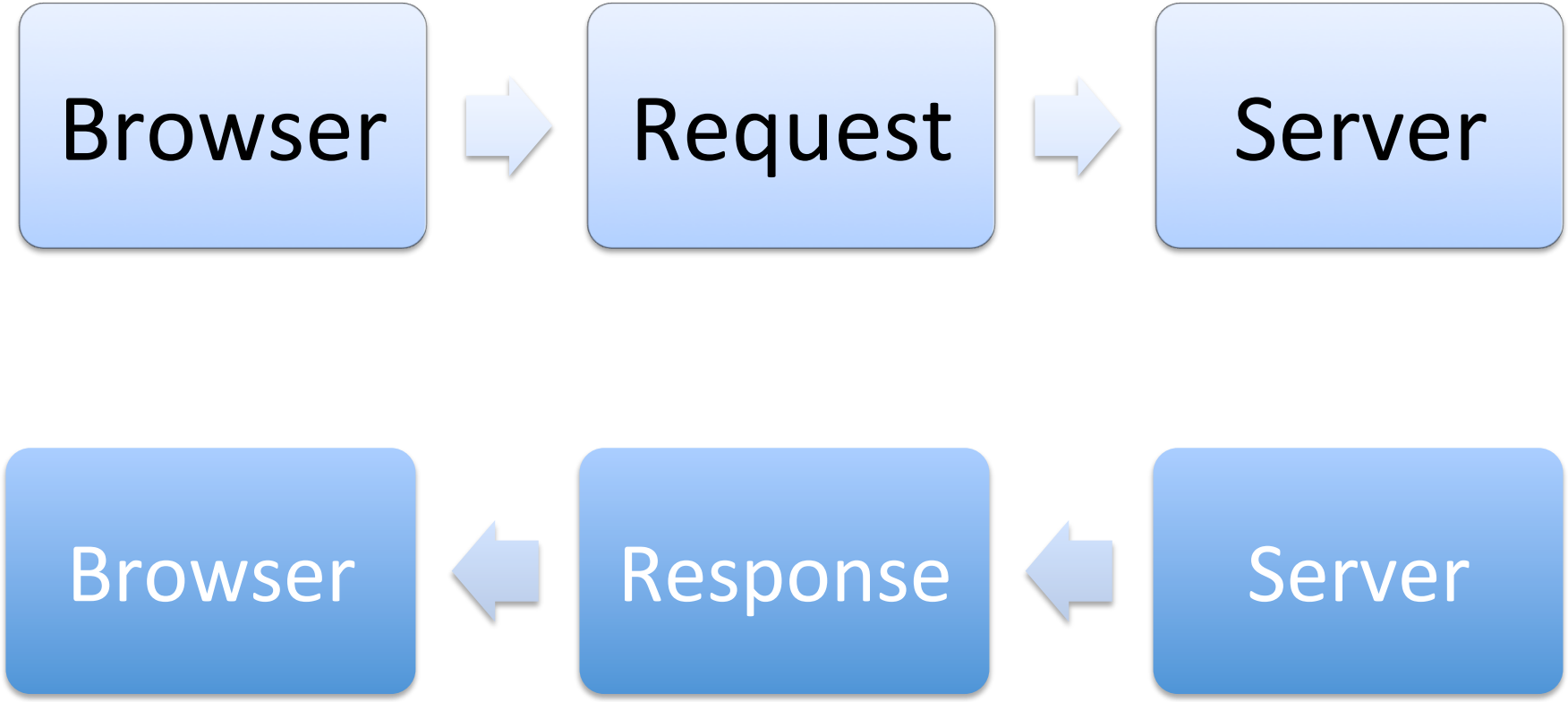
TCP

- TCP is responsible for chopping up data
- TCP is also responsible for putting it back together in the right order
- TCP sends receipts for each piece of data, so if a piece is lost it can be resent

HTTP

- The Hypertext Transfer Protocol (**HTTP**) is an application protocol for distributed, collaborative, hypermedia information systems. Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text.

How HTTP Works



Features of HTTP

- connectionless – make the request, then disconnect and wait for the server to talk back
- media independent – anything can be sent, as long as you attach the type of content being sent
- stateless – server and client only know about each other during the request, afterwards they forget it happened

HTTP format

Start Line

Headers

Body

An HTTP Request

```
GET /hello.htm
HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01;
Windows NT)
Host: example.com
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive
```

An HTTP Response

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body
<h1>Hello, World!</h1>
</body>
</html>
```

HTTP Verbs

- GET
- POST
- HEAD
- PUT
- DELETE
- CONNECT
- OPTIONS
- TRACE

HTTP RESPONSES

- **1xx: Informational**
  - It means the request has been received and the process is continuing.
- **2xx: Success**
  - It means the action was successfully received, understood, and accepted.
- **3xx: Redirection**
  - It means further action must be taken in order to complete the request.
- **4xx: Client Error**
  - It means the request contains incorrect syntax or cannot be fulfilled.
- **5xx: Server Error**
  - It means the server failed to fulfill an apparently valid request.

Clients

- Your browser
  - Safari
  - Chrome
  - Firefox
  - IE
- Tools
  - curl
- Programs
  - requesting via code

HTML and CSS (and JavaScript)

- **HTML** (the Hypertext Markup Language) and **CSS** (**Cascading Style Sheets**) are two of the core technologies for building Web pages. **HTML** provides the structure of the page, **CSS** the (visual and aural) layout, for a variety of devices
- JavaScript – browser embedded programming language

Remember - PHP is on the Server

- PHP creates text sent back as an HTTP response
- This is Server Side Processing
- To interact on the client, use JavaScript

Lesson 3:

REDIRECTS AND HEADERS

Header Redirect

```
<?php

header('Location: http://www.newpage.com');
exit;
```

Lesson 2:

BASIC HTML

HTML Page Parts

- <!DOCTYPE html>
- <html></html>
- <head></head>
- <body></body>

HTML Tags

- HTML uses opening and closing tags to start and stop sections:
  - Title: <title>Some text</title>
  - Div: Some text
  - Paragraph: Some text
  - Span: Some text

HTML Tags

- HTML uses opening and closing tags to start and stop formatting:
  - Italics (emphasis): Some text
  - Bold: Some text
  - Link: Link Text
  - Headings: <h1>Some text</h1>

HTML Tags

- Some HTML tags stand alone:
  - Line Break:
  - Special Characters, like "<": <

Exercise

- Create an HTML page with a title of "A Simple Form" that says "Welcome!" in an H1 heading and followed by "Please fill out the form below:" enclosed in paragraph tags.

Solution

```
<!DOCTYPE html>
<html>
<head>
  <title>A Simple Form</title>
</head>
<body>

<h1>Welcome!</h1>

<p>Please fill out the form below:</p>

</body>
</html>
```

Lesson 4:

FORMS

HTTP in PHP

- $_GET versus $_POST

Basic Form Tags

```
<form action="index.php" method="POST">

<label>
  First Name:<br>
  <input type="text" name="firstName" value="">
</label>

</form>
```

Input Types

```
<label>
  Textbox (1 line):
  <input type="text" name="firstName" value="">
</label>

<label>
  Textarea (multi-line):
  <textarea name="fieldName"></textarea>
</label>
```

Input Types

```
Radio Buttons (choose one):
<input type="radio" name="chooseOne" value="1"> 1
<input type="radio" name="chooseOne" value="2"> 2
<input type="radio" name="chooseOne" value="3"> 3

Check Boxes (choose multiple):
<input type="checkbox" name="choices[]" value="1"> 1
<input type="checkbox" name="choices[]" value="2"> 2
<input type="checkbox" name="choices[]" value="3"> 3
```

Input Types

```
Drop-down:
<select name="fieldName">
<option value="1"> First Choice</option>
<option value="2"> Second Choice</option>
<option value="3"> Third Choice</option>
</select>
```

Input Types

```
Hidden fields:
<input type="hidden" name="status" value="text">

Submit Button:
<input type="submit" name="submitButton" value="Go">

Reset Button:
<input type="reset" name="resetButton" value="Clear
the Form">
```

Exercise

- On your HTML page from the previous exercise, create a form using POST with a text field for first name and a text field for email address. Then, have a set of radio buttons for choosing favorite color (have at least three options). Next, have a set of checkboxes for choosing which pets you have (have at least three options). Finally, have a submit button that says "Submit my data".

Solution

```
<form action="index.php" method="post">

<p>First Name:<br>
<input type="text" name="firstName" value=""></p>
<p>Email Address:<br>
<input type="text" name="email" value=""></p>

<p>Favorite Color:<br>
<input type="radio name="favColor" value="Red">
Red<br>
<input type="radio name="favColor" value="Green">
Green<br>
<input type="radio name="favColor" value="Blue">
Blue</p>
```

Solution

```
<p>Pets:<br>
<input type="checkbox name="pets[]" value="Dog">
Dog<br>
<input type="checkbox name="pets[]" value="Cat">
Cat<br>
<input type="checkbox name="pets[]" value="Turtle">
Turtle</p>


<input type="submit" name="submitButton"
value="Submit my data">


</form>
```

Where does the data go?

- All form data comes into PHP as a string.
- $_GET or $_POST

```
<input type="text" name="firstName" value="">

$_POST['firstName']

<input type="checkbox" name="choice[]" value="Red">

$_POST['choice'][0]
```

Superglobal contents

- Contains all form data, including submit button and hidden fields

```
<input type="submit" name="submitButton"
value="Submit my data">

echo $_POST['submitButton'];

//Submit my data
```

Exercise

- Update your form script so that it checks the value of the submit button. If the form has not been submitted, display the form. If the form has been submitted, display the values of the form.

Solution

```
if ($_POST['submit'] == "Submit my data") {
   echo "Name: {$_POST['firstName']}<br>
   Email: {$_POST['email']}<br>
   Favorite Color: {$_POST['favColor']}<br>
   Pets: " . implode(',', $_POST['pets']) . "<br>";
}
else {
   //form code
}
```

php.net

Downloads Documentation Get Involved Help

PHP is a popular general-purpose scripting language that is especially suited to web development.

Fast, flexible and pragmatic, PHP powers everything from your blog to the most popular websites in the world.

PHP 5.4.25 Released! » 2014-02-06

The PHP development team announces the immediate availability of PHP 5.4.25. 5

str_getcsv

FUNCTIONS 119

str_getcsv
Parse a CSV string into an array

str_ireplace
Case-insensitive version of

str_pad
Pad a string to a certain length

str_repeat
Repeat a string

CLASSES 1

streamWrapper
The streamWrapper class

» Search php.net for str

Or, go directly to: http://www.php.net/strlen

Function Pages

strlen

Change language: English

(PHP 4, PHP 5)

Edit Report a Bug

strlen — Get string length

Description

```
int strlen ( string $string )
```

Returns the length of the given `string`.

Parameters

`string`

The string being measured for length.

Return Values

The length of the `string` on success, and `0` if the `string` is empty.

Changelog

Version	Description
5.3.0	Prior versions treated arrays as the string *Array*, thus returning a string length of *5* and emitting an `E_NOTICE` level error.

Examples

Example #1 A strlen() example

```
<?php
$str = 'abcdef';
echo strlen($str); // 6

$str = ' ab cd ';
echo strlen($str); // 7
?>
```

Notes

Note:
strlen() returns the number of bytes rather than the number of characters in a string.

See Also

- count() - Count all elements in an array, or something in an object
- mb_strlen() - Get string length

User Contributed Notes 10 notes

add a note

22 tux at tax dot tox 1 year ago

```
Attention with utf8:
$foo = "bär";
strlen($foo) will return 4 and not 3 as expected..
```

Validate Input, Escape Output

- Nothing from an external source (like $_POST or $_GET) can be trusted.
- Focus on validating when bringing data into your form.
- Always escape the data whenever it is leaving your script.

Simple Validation

- Character Type Checking Functions: http://php.net/ctype

```
//Checks for digits only
if (ctype_digit($_POST['myVar'])) {
   echo "Yes, this contains only digits";
}
```

Simple Validation

- Filter variables
  http://php.net/filter_var

```
//Checks for a valid URL
if (filter_var($_POST['url']), FILTER_VALIDATE_URL) {
   echo "Yes, this is a URL";
}
```

Simple Escaping

- Convert all applicable characters to their HTML entities http://php.net/htmlentities

```
$myVar = "<b>text</b>";
echo htmlentities($myVar);
//&lt;b&gt;text&lt;b&gt;
```

Simple Escaping

- Remove HTML and PHP tags
  http://php.net/strip_tags

```
$myVar = "<b>text</b>";
echo strip_tags($myVar);
//text
```

Exercise

- Take a look at: http://php.net/manual/filter.filters and validate the data you are bringing in through your form. If it validates, then display the data.
- Escape the data before displaying it on the page.

Solution

```
if (ctype_alpha($_POST['firstName'])) {
  echo "Name: " . htmlentities($_POST['firstName']) . "<br>";
}
if (filter_var($_POST['email']), FILTER_VALIDATE_EMAIL) {
  echo "Email: " . htmlentities($_POST['email']) . "<br>";
}
if (($_POST['favColor'] == "Red") || ($_POST['favColor'] ==
"Green") || ($_POST['favColor'] == "Blue")) {
  echo "Favorite Color: " . htmlentities($_POST['favColor'])
  . "<br>";
}
$theirPets = implode(',', $_POST['pets']);
if (ctype_alpha(str_replace(',', '', $theirPets)) {
  echo "Pets: " . htmlentities($theirPets) . "<br>";
}
```

Using Forms

When a validation test fails, make it easy for your user to fix it (Check for malicious submissions, but always treat your users as though it were an accident).

Solution

```
$checkVar = "Bad";
if ($_POST['submit'] == "Submit my data") {
   //Check your validation
   if ($var == "good") {
      $checkVar = "Good";
   }
   else {
      $checkVar = "Bad";
   }
}

if ($checkVar == "Good") {
   //display submitted data
}
else {
   //form code
}
```

Refill the Form

Don't make your users practice their typing skills. Always refill non-malicious data.

Personal Habit:

When I validate my data, I assign it to a local variable, so I know I'm using my validated data.

Example

```
<form action="index.php" method="post">


<p>First Name:<br>
<input type="text" name="firstName"
value="$firstName"></p>
<p>Email Address:<br>
<input type="text" name="email" value="$email"></p>
```

Example

```
<p>Favorite Color:<br>

<input type="radio name="favColor" value="Red"<?php
if($favColor == "Red") { echo " checked"; } ?>>
Red<br>

Echo '<input type="radio name="favColor"
value="Green"';
if ($favColor == "Green") {
  echo " checked";
}
echo '> Green<br>';
```

Example

```
echo '<p>Pets:<br>
<input type="checkbox name="pets[]" value="Dog"';

if (in_array("Dog", $pets) {
   echo " checked";
}

echo '> Dog<br>
<input type="checkbox name="pets[]" value="Cat">
Cat<br>
<input type="checkbox name="pets[]" value="Turtle">
Turtle</p>';
```

Lesson 5:

SESSIONS AND COOKIES

Accessing the Data

Sessions:

- Server-side
- Less picky on header timing (but still picky)

Cookies:

- Client-side
- Must occur before headers are sent

Both:

- Allow data to be stored by one script and accessed by another
- Accessible via superglobal array

Using Sessions

Place this at the very top of your page:

```
session_start();
```

This must occur before headers are sent. Things that will send the headers:

- the HTML declarations
- Whitespace
- echo'ing anything

Example

```
session_start();
$_SESSION['custName'] = $_POST['firstName'];
```

Example

```
session_start();
echo "Hello, {$_SESSION['custName']}. Welcome back!";
```

Example

```
setcookie("custName", $_POST['firstName']);


header("Set-Cookie: custName=$_POST['firstName'];
custEmail=$_POST['email']");
```

Example

```
echo "Hello, {$_COOKIE['custName']}. Welcome back!";
```