The Internet

PHP works with the web

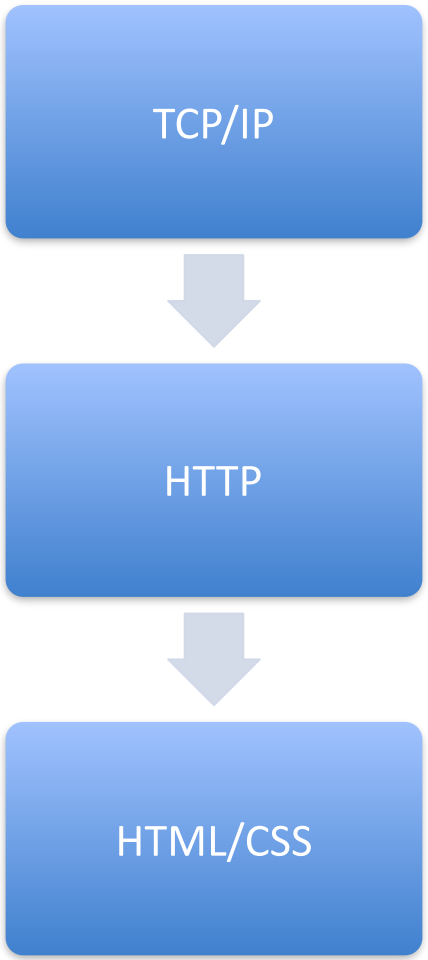
Lesson 1:

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

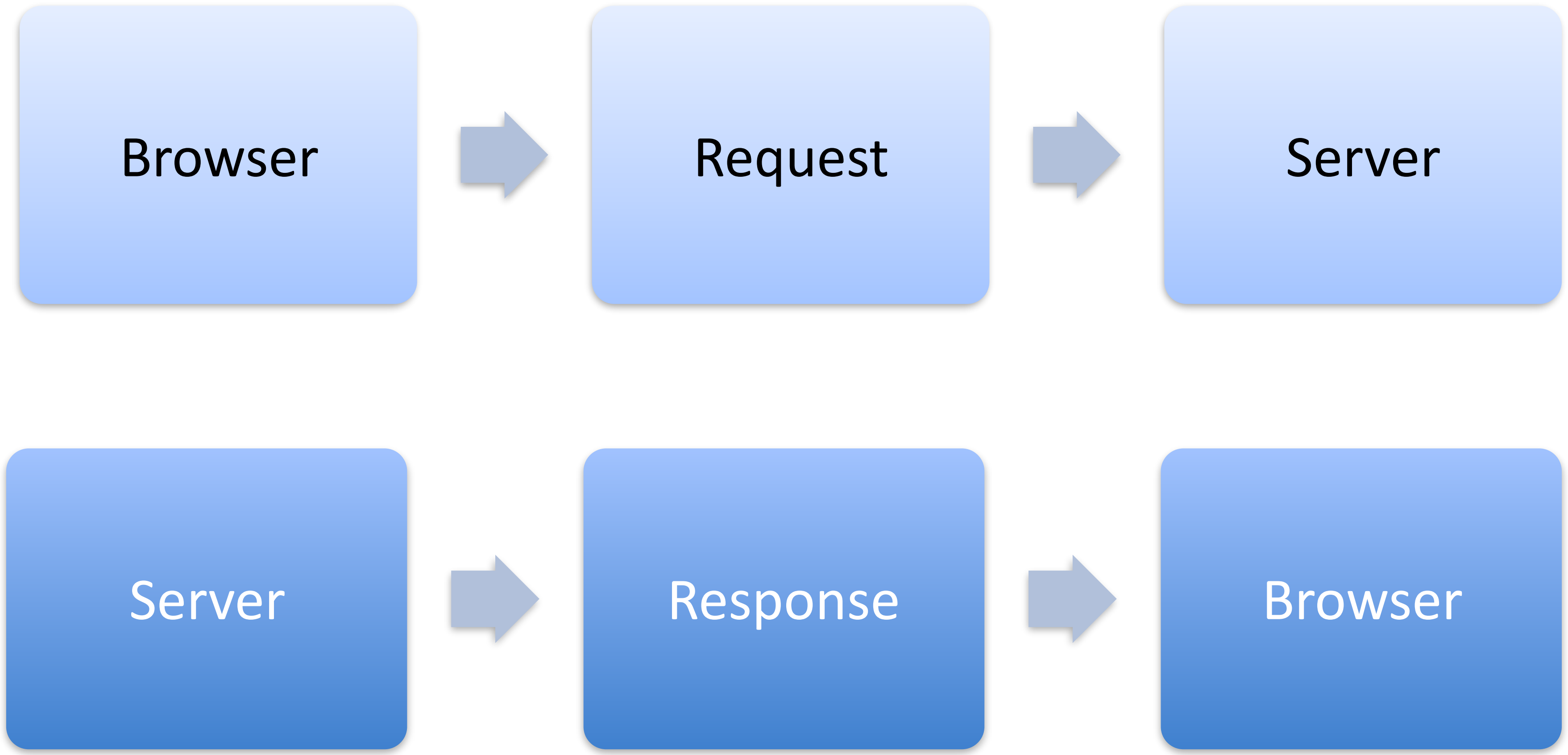
TCP

- 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.
- Defined in RFCs 7230-7235

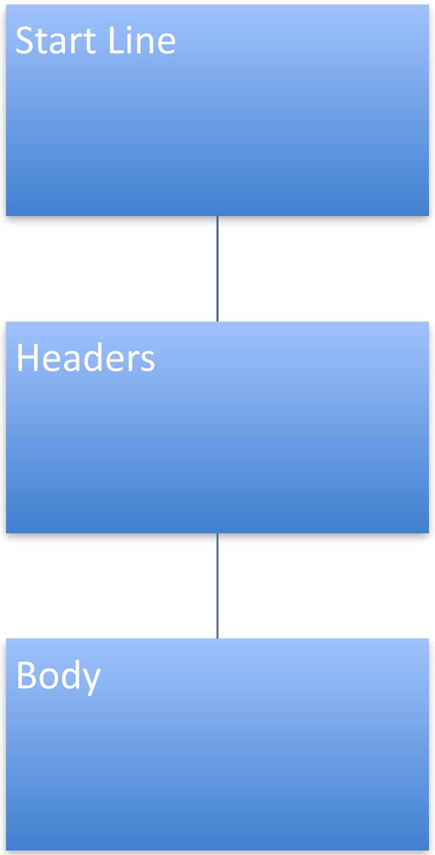
How HTTP Works



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



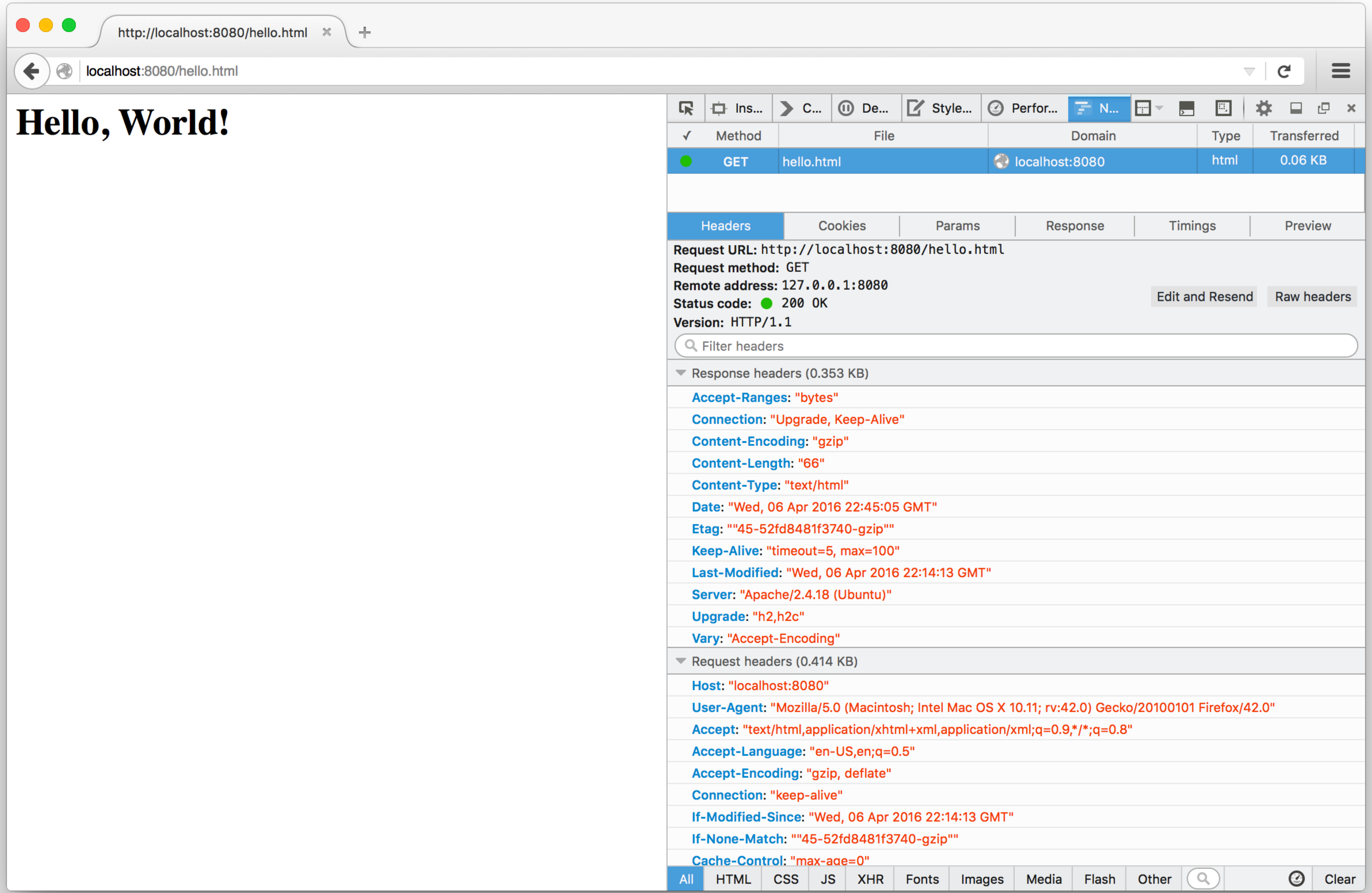
An HTTP Request

```
GET /hello.html HTTP/1.1
Accept: text/html,application/xhtml+xml,*/*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.5
Cache-Control: max-age=0
Connection: keep-alive
Host: example.com
User-Agent: Mozilla/5.0 Gecko/20100101 Firefox/42.0
```

An HTTP Response

```
HTTP/1.1 200 OK
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 76
Content-Type: text/html
Date: Wed, 06 Apr 2016 22:15:39 GMT
ETag: "45-52fd8481f3740-gzip"
Last-Modified: Wed, 06 Apr 2016 22:14:13 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding

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



Hello, World!

✓	Method	File	Domain	Type	Transferred
●	GET	hello.html	localhost:8080	html	0.06 KB

Request URL: `http://localhost:8080/hello.html`
Request method: GET
Remote address: 127.0.0.1:8080
Status code: ● 200 OK
Version: HTTP/1.1

Response headers (0.353 KB)

```
Accept-Ranges: "bytes"
Connection: "Upgrade, Keep-Alive"
Content-Encoding: "gzip"
Content-Length: "66"
Content-Type: "text/html"
Date: "Wed, 06 Apr 2016 22:45:05 GMT"
Etag: ""45-52fd8481f3740-gzip""
Keep-Alive: "timeout=5, max=100"
Last-Modified: "Wed, 06 Apr 2016 22:14:13 GMT"
Server: "Apache/2.4.18 (Ubuntu)"
Upgrade: "h2,h2c"
Vary: "Accept-Encoding"
```

Request headers (0.414 KB)

```
Host: "localhost:8080"
User-Agent: "Mozilla/5.0 (Macintosh; Intel Mac OS X 10.11; rv:42.0) Gecko/20100101 Firefox/42.0"
Accept: "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
Accept-Language: "en-US,en;q=0.5"
Accept-Encoding: "gzip, deflate"
Connection: "keep-alive"
If-Modified-Since: "Wed, 06 Apr 2016 22:14:13 GMT"
If-None-Match: ""45-52fd8481f3740-gzip""
Cache-Control: "max-age=0"
```

Browser Network Inspector

- Firefox
  - Mac: Tools > Web Developer > Network
  - Windows: ☰ > Developer > Network
- Chrome
  - Mac: View > Developer > Developer Tools (Network tab)
  - Windows: ☰ > More tools > Developer Tools (Network tab)

Browser Network Inspector

- Internet Explorer
  - ⚙ > F12 Developer Tools (Network tab)
- Microsoft Edge
  - … > F12 Developer Tools (Network tab)

Browser Network Inspector

- Safari
  - First, go to Safari > Preferences (Check "Show Develop menu in menu bar" on Advanced tab)
  - Then it’s under Develop > Show Web Inspector

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
  - Your browser’s Web Developer toolbar
  - Fiddler web debugging proxy - www.telerik.com/fiddler
  - httpie - httpie.org
  - curl - curl.haxx.se/docs/manpage.html
- Programs
  - requesting via code (i.e. file_get_contents() in PHP, etc.)

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 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: <div>Some text</div>
  - Paragraph: <p>Some text</p>
  - Span: <span>Some text</span>

HTML Tags

- HTML uses opening and closing tags to start and stop formatting:
  - Italics (emphasis): <em>Some text</em>
  - Bold: <strong>Some text</strong>
  - Link: <a href="http://url.com">Link Text</a>
  - Headings: <h1>Some text</h1>

HTML Tags

- Some HTML tags stand alone:
  - Line Break: <br>
  - Special Characters, like "<": &lt;

Exercise: Basic HTML

- Run the source using the built-in PHP web server and load it in your browser
- Create a basic HTML page
- Integrate Bootstrap for a prettier layout
  - http://getbootstrap.com/
- Edit the HTML to add new paragraphs or other data, reload in your browser

Exercise source: Section_2_Lesson_2_Basic_HTML

Lesson 3:

REDIRECTS AND HEADERS

Header Redirect

```
<?php

header('Content-Type: image/png');

// echo image content
```

Header Redirect

```php
<?php

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

Exercise: Redirects & Headers

- Use a header to display a plain text page
  - Try editing the plain text and reloading the page in the browser to see what happens
- Use a header to redirect to another page
  - Change the redirection location to redirect to a page of your choice

Exercise source: Section_2_Lesson_3_Redirects_Headers

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: Forms

- Create a form for a journal entry:
    - create.php
    - Fields
        - Title
        - Article
        - Submit button
- Add a different form field of your choice

Exercise source: Section_2_Lesson_4_Forms

Where does the data go?

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

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

$_POST['title']

<textarea name="article"></textarea>

$_POST['article']
```

Superglobal contents

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

```
<input type="submit" name="submit" value="Create">

echo $_POST['submit'];

//Create
```

Superglobal contents

- You can also check if the $_POST superglobal contains anything

```
if (isset($_POST) && count($_POST) > 0) {

    // do something with the data

}
```

Exercise: Forms

- Update your form script so that it checks whether $_POST contains any values. If it does, use var_dump() to dump the value of $_POST.

Exercise source: Section_2_Lesson_4_Forms

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.
- Add an "email" form field and validate that it contains a valid email address.

Exercise source: Section_2_Lesson_4_Forms

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).

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.

Exercise: Forms

- Update the form fields so that they display the values entered by the user after submission.
- Be sure to escape the output properly.
- Use Bootstrap field validation states to show fields with errors.

Exercise source: Section_2_Lesson_4_Forms

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

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

Exercise: Sessions

- Create a login page to log in a user and create a session
- Store some data in the session when “logging in” the user
- Display the session data on the index.php page, if it exists
- Create a logout page that removes the session data

Exercise source: Section_2_Lesson_5_Sessions