Web Programmering 3

Lesson 3: PHP

Henrik Lange

Agenda

- Questions from last time
- HTML & CSS recap
- Last weeks exercises
- Assignment 1
- Client Server Relationships
- HTTP Protocol
- Session & Cookies
- PHP
- Mixing PHP and HTML
- Variables
- If-then-else
- Loops
- Functions
- Super Globals
- Filtering
- JavaScript AJAX

Install PHP

Mac (this should be the same on Linux, but I only tried it on Mac)

- Open the terminal (press cmd +space to open spotlight, write terminal, press enter)

- Write everything after the colon in a single line and press enter: /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

  - This will install "Homebrew" on your computer along with XCode command line tools. It can take a good chunk of time, so be patient.

  - You might need to press enter an extra time and put in your system-password.

  - If something fails, just try again

- To install PHP via Homebrew write the following in the terminal: brew install php73

- Keep Homebrew, as you will use it to install MariaDB later on

Install PHP

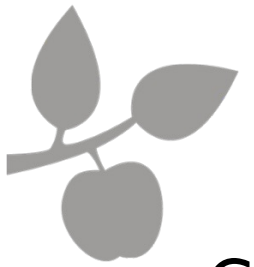
Windows

- Go to the following site: https://windows.php.net/download#php-7.3
- Select the link called "zip" in the second grey box for 64-bits, or in the 4th box for 32-bit windows
- Put the content of that zip file in a folder directly on your C drive, such as C:/php
- Inside the folder, rename the file php.ini-development to php.ini
- Open php.ini in a text editor and change the line ;extension_dir = "ext" to extension_dir = "C:/php/ext"
    - yes, the leading semicolon was removed.
- Open the control panel and search for "Edit the system environment variables"
- A new windows opens, and in the bottom of it, select Environment Variables
- In the SECOND list (not the first!), select the element called "Path" and click the "Edit" button
- Click "new" and enter the php folder path: C:/php
- Click OK buttons until everything is closed and reboot your computer

Test PHP installation

- Create a text file with the following content:
- <?php echo 'hello world';
- Save the text file as index.php in any folder on your computer
- Open the folder that contains the .php file inside your Terminal (Mac/Linux/Unix) or CMD (Windows)
- Use the following command to start the PHP server inside the folder: php -S localhost:8080
- Open your browser and type in this URL: localhost:8080
- The text Hello World should appear as the only thing in the browser window

Finally, write down the location of your php.ini file, as we will be working with it later on



Questions from last time

Can you import a stylesheet from a stylesheet?

- @import “mystyle.css”;
- @import url(“mystyle.css”);
- Both of these are valid ways of including a .css file from css itself
- Works inside .css files or the <style> HTML element

Questions from last time

RegEx "AND"

- Regular Expressions are sequences
- \w\s means "word character", **then** "whitespace".
- A string can not contain two characters in the same spot
- If either is okay, I can use OR:
  - \w|\s

Questions from last time

Why does my regex.test() return true when the pattern does not match?

- RegEx looks for **a** match!
- Example: \w{4} matches "hello" & "hell" but not "hel"
- "hello" **contains** a match!
- ^ is the beginning of a string, and $ is the end
- ^\w{4}$ will match "hell", but not "hello" or "hel"

Questions from last time

Why is my JavaScript not registering?

- JavaScript should no longer be in the HTML head!
- Should be placed as the last element in the HTML body

Questions from last time

What about the HTML 5 **pattern** attribute?

```
3    <input type="text" pattern="^\+45\d{8}$"
4           title="Must be +45 followed by 8 digits"
5           required/>
```

Questions from last time

What about HTML5 input type="date" ?

- Let's take a look at browser implementation
  - Chrome, IE, Edge, Firefox

- Check compatibility
  - Caniuse.com

Plan

- Week 1: HTML & CSS
- Week 2: JavaScript
- Week 3: HTTP, PHP & AJAX
- Week 4: MariaDB / MySQL
- Week 5: Recap, Internet History, Project help
- Week 6: Project Hand-In
- Week 6: OOP PHP, MVC, Routers, Reflection
- Week 6: Midway Evaluation
- Week 7: Project Presentation
- Week 7: Assignment 2 specification
- Week 8: API's and Frameworks

HTML & CSS Recap

Let's see if it sticks!

- http://fuzzball.nu/Projects/simplevote

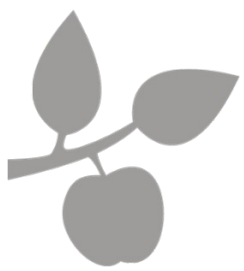
Exercises from last week (s)

Let's take a look at the sample solutions

Javascript – Cat & Mouse

Javascript – Regex

CSS – Basic setup



Assignment 1 – Content

- AJAX call **ideas**:
  - Page could load more images on scroll
  - You could make a search bar with suggestions for
    - Other users
    - Pictures
  - Make a "like" button
  - Let users comment on each others pictures
  - …or whatever you can come up with

- Note that none of these examples are required – just that you use an AJAX call in your application.

Note on portfolio project

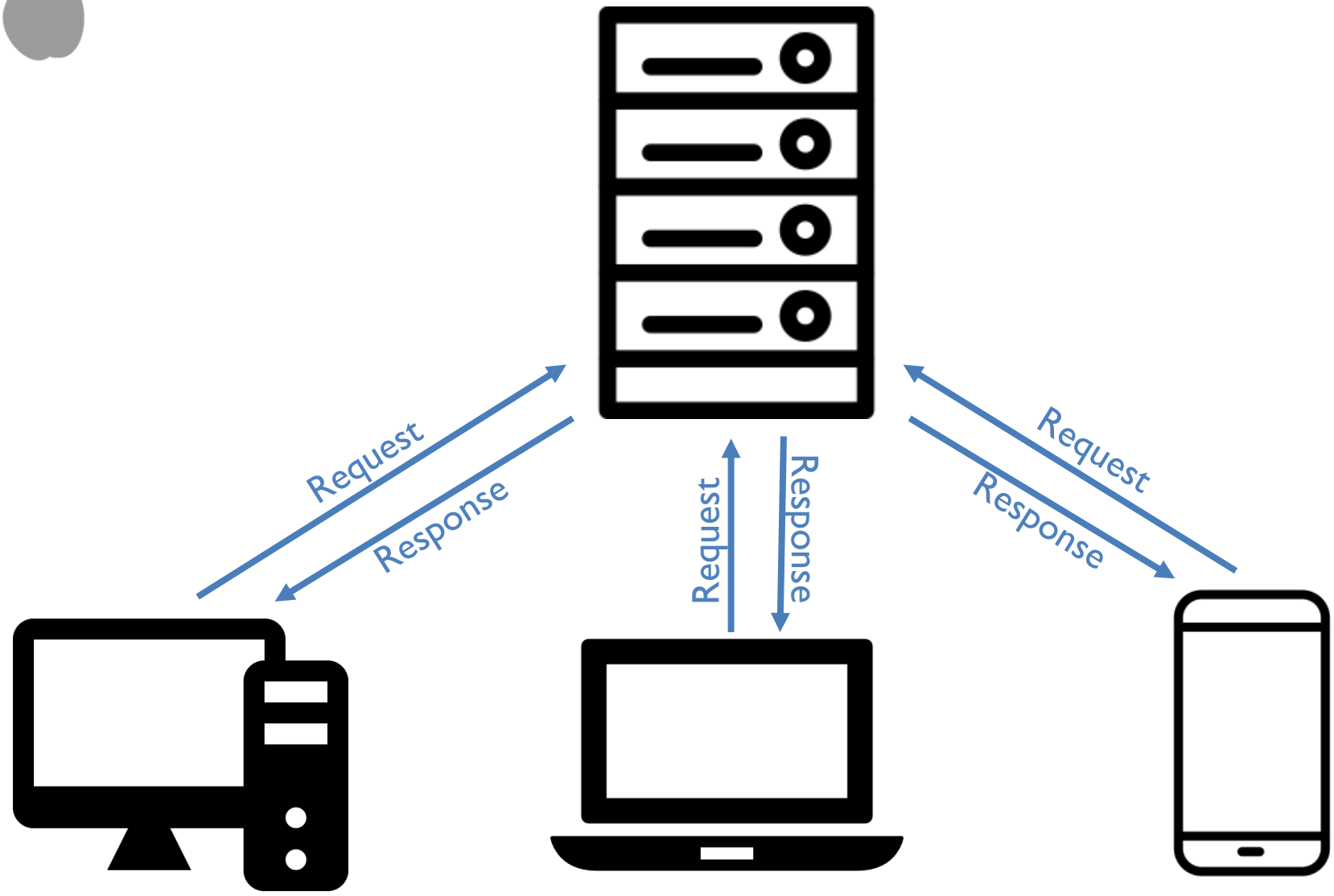
- For those who build ahead of the lectures:
  - We use PDO to connect PHP to the Database
  - The reason is security – and the security lectures later in the course are based on PDO

- For those of you who work along with the lectures, this will be be apparent when we get to it

GitHub

- Jeg har optaget 25 minutters video om Git og GitHub
- Indholdet er en del af pensum
- I kan finde videoen her:
  - https://www.youtube.com/watch?v=v51vmQSwW80

Client Server Relationship



Client Server Relationship

- Servers can communicate directly with each other
- Clients can communicate directly with each other
- The most common setup is where clients only communicate with the server

Client Server Relationship

- All online devices have an IP address per online network driver
- Multiple servers on a single IP address are reached through ports
  - Port forwarding
  - Multiple servers on one device
- URL's are IP address shortcuts
  - DNS (Domain Name Server)

- Let's try our local network settings

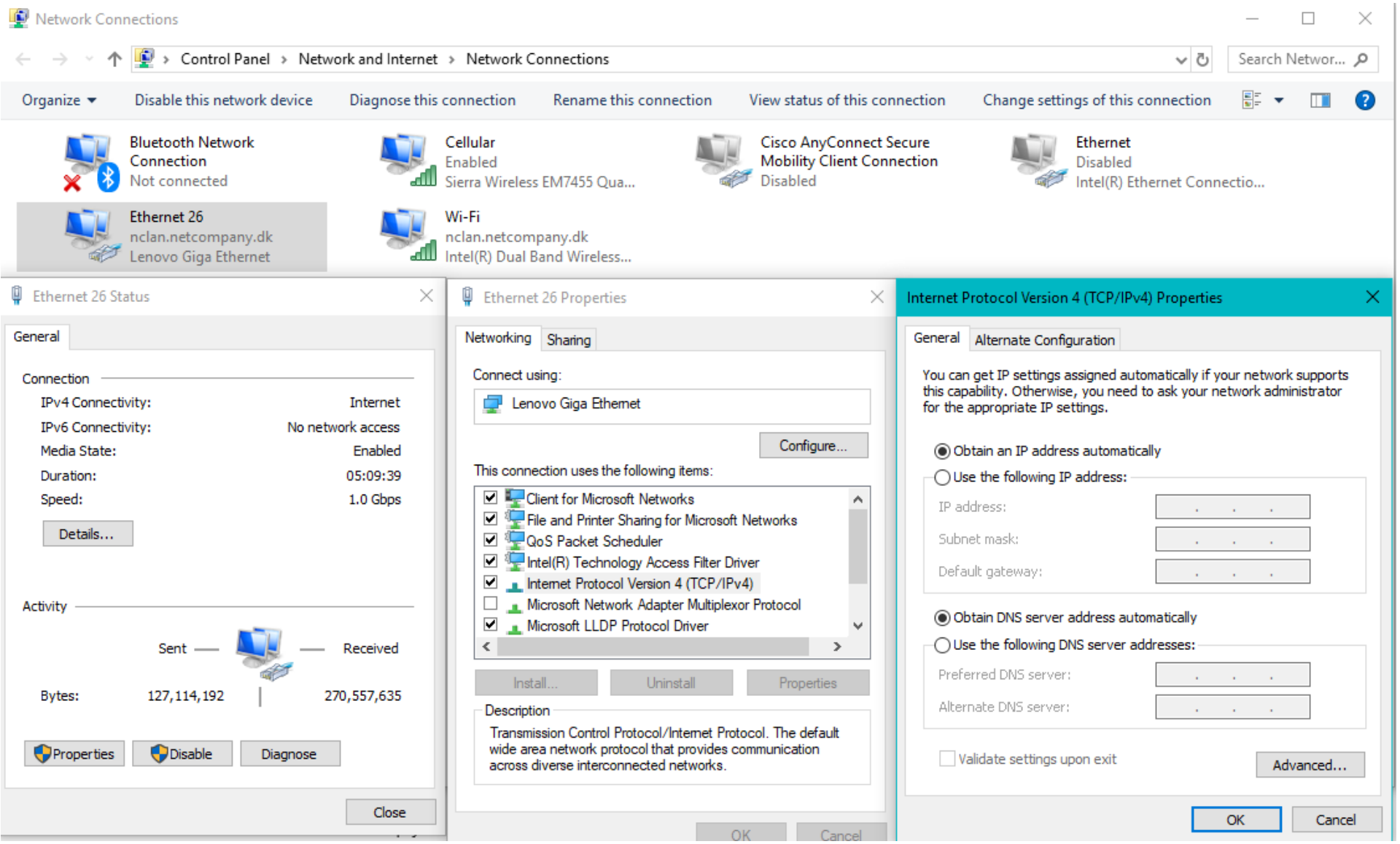
Press Enter

- Address is put into browser
- Application >> LAN already explained
- LAN is connected by router
  - Router connected to ISP (internet service provider)
  - ISP has DNS (domain name server)
  - DNS translates domain name to IP address
  - Browser saves IP for further communication
  - Routers throughout the world communicate to find the requested IP address
  - Server responds to request

The Hardware

- Ethernet
  - A type of port with a RJ45 plug
- Hub
    - Lets multiple devices connect
    - Communication is possible, but requires manual setup
- Switch
    - A powered Hub
- Router
    - A switch with DHCP
      - Dynamic Host Configuration Protocol
    - Distributes IP addresses across the network
    - Can communicate with other routers
- Modem
    - Connects home network to ISP

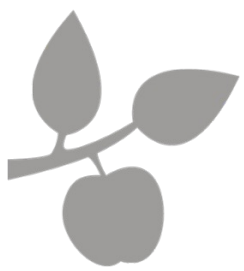
Network Settings



HTTP Stack

- Other alternatives exist

Layer	Example	Alternative
Application	Browser	App
Application Layer	HTTP	HTTPS
Transport layer	TCP	UDP
Network layer	IP	IP
Link layer	Ethernet Driver	WiFi Driver
Hardware layer	Ethernet	WiFi chip



IP

- IP is the address protocol
  - IPv4 is the most commonly used protocol today
    - IPv4 uses 32-bit addresses (2^{32})
    - Defined by 4 sets of 3 digits, divided by dots
    - Ports are supplied after a colon
    - Path is defined after a slash
      - 123.321.12.3:80/folder
    - The subnet mask defines the local networking range
    - 127.0.0.0 is usually your localhost
    - 192.168 and 10.0 addresses are usually used for LAN
  - IPv6 is the latest version of the protocol
    - Uses 128-bit addresses (2^{128})
    - Defined by 8 sets of 4 hex digits, where 0000 can be omitted
      - 12fa:32ff::::0291::9de1

TCP

- Handles connection handshakes
- Handles data packages

HTTP

Contents:

- Verb
- Resource
- Version
- Headers
- Status
- Body

Request:

GET /address/ HTTP/1.1

Accept: text/html

Response:

HTTP/1.1 200 OK

Content-type: text-html

<!DOCTYPE html>

<html lang="en">

…

HTTP Verbs

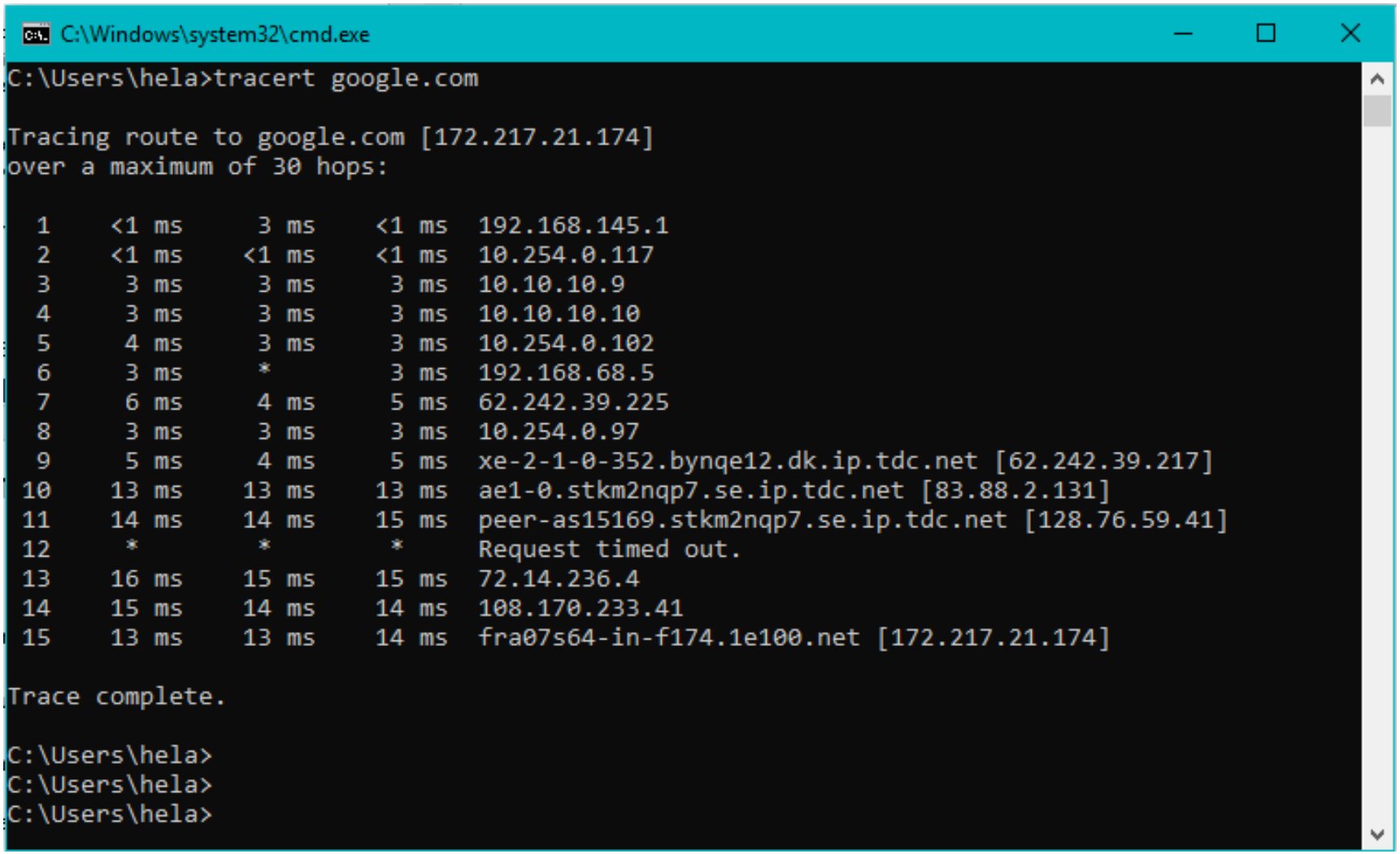
- GET
  - Retrieve data (from server to client)
  - Should have no side effects
  - Parameters in resource (URI)
- POST
  - Send data (from client to server)
  - Parameters in body
- PUT
  - Replace data (sent from client to server)
- DELETE
  - Removes data

- GET, PUT & DELETE are idempotent

HTTP Verbs

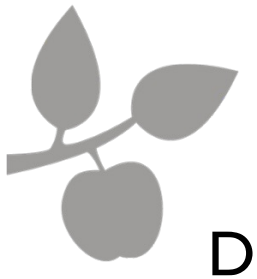
- HEAD
  - Same as GET, but transfers status and header only
- CONNECT
  - Establishes a tunnel identified by URI
- OPTIONS
  - Describes communication options for target
- TRACE
  - Performs a message loop-back test along the path to the target
  - Let's try CMD tracert

Trace


C:\Windows\system32\cmd.exe
C:\Users\hela>tracert google.com
Tracing route to google.com [172.217.21.174]
over a maximum of 30 hops:
1 <1 ms 3 ms <1 ms 192.168.145.1
2 <1 ms <1 ms <1 ms 10.254.0.117
3 3 ms 3 ms 3 ms 10.10.10.9
4 3 ms 3 ms 3 ms 10.10.10.10
5 4 ms 3 ms 3 ms 10.254.0.102
6 3 ms * 3 ms 192.168.68.5
7 6 ms 4 ms 5 ms 62.242.39.225
8 3 ms 3 ms 3 ms 10.254.0.97
9 5 ms 4 ms 5 ms xe-2-1-0-352.bynqe12.dk.ip.tdc.net [62.242.39.217]
10 13 ms 13 ms 13 ms ae1-0.stkm2nqp7.se.ip.tdc.net [83.88.2.131]
11 14 ms 14 ms 15 ms peer-as15169.stkm2nqp7.se.ip.tdc.net [128.76.59.41]
12 * * * Request timed out.
13 16 ms 15 ms 15 ms 72.14.236.4
14 15 ms 14 ms 14 ms 108.170.233.41
15 13 ms 13 ms 14 ms fra07s64-in-f174.1e100.net [172.217.21.174]
Trace complete.
C:\Users\hela>
C:\Users\hela>
C:\Users\hela>

HTTP Resource

- Regular
  - gp?product=1234&comment=4321
    - url?key1=value1&key2=value2&key3=value3
- RESTful
  - gp/product/1234/comment/4321



HTTP Version

Different versions have different behavior:

- 1991: HTTP 0.9
  - Single line protocol with no headers.
  - Only GET supported: GET index.html
- 1996: HTTP 1.0
  - Added headers, and a version string
- 1999: HTTP 1.1
  - Connection keep-alive by default
  - Additional caching mechanisms
  - **Primary HTTP version used today**
- 2015: HTTP 2.0
  - Binary framing
  - Header compression
  - General optimizations

HTTP Headers

- Request specific format:

```
Accept: text/html
```

- Request secondary format if first is not available

```
Accept: application/json,application/xml
```

- Request preferred encoding

```
Accept-Encoding: bzip2,gzip
```

- Request preferred language

```
Accept-Language: dk, es, en-US
```

- User-Agent (usually set by browser)

```
User-Agent: Mozilla/5.0 (Windows NT 10.0;Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/71.0.3578.98 Safari/537.36
```

HTTP Status Codes

1xx Informational

100 Continue
101 Switching Protocols
102 Processing (WebDAV)

2xx Success

★ 200 OK
★ 201 Created
202 Accepted
203 Non-Authoritative Information
★ 204 No Content
205 Reset Content
206 Partial Content
207 Multi-Status (WebDAV)
208 Already Reported (WebDAV)
226 IM Used

3xx Redirection

300 Multiple Choices
301 Moved Permanently
302 Found
303 See Other
★ 304 Not Modified
305 Use Proxy
306 (Unused)
307 Temporary Redirect
308 Permanent Redirect (experimental)

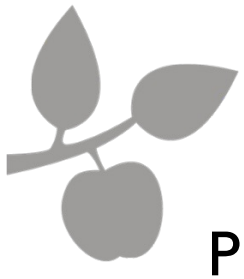
4xx Client Error

★ 400 Bad Request
★ 401 Unauthorized
402 Payment Required
★ 403 Forbidden
★ 404 Not Found
405 Method Not Allowed
406 Not Acceptable
407 Proxy Authentication Required
408 Request Timeout
★ 409 Conflict
410 Gone
411 Length Required
412 Precondition Failed
413 Request Entity Too Large
414 Request-URI Too Long
415 Unsupported Media Type
416 Requested Range Not Satisfiable
417 Expectation Failed
418 I'm a teapot (RFC 2324)
420 Enhance Your Calm (Twitter)
422 Unprocessable Entity (WebDAV)
423 Locked (WebDAV)
424 Failed Dependency (WebDAV)
425 Reserved for WebDAV
426 Upgrade Required
428 Precondition Required
429 Too Many Requests
431 Request Header Fields Too Large
444 No Response (Nginx)
449 Retry With (Microsoft)
450 Blocked by Windows Parental Controls (Microsoft)
451 Unavailable For Legal Reasons
499 Client Closed Request (Nginx)

5xx Server Error

★ 500 Internal Server Error
501 Not Implemented
502 Bad Gateway
503 Service Unavailable
504 Gateway Timeout
505 HTTP Version Not Supported
506 Variant Also Negotiates (Experimental)
507 Insufficient Storage (WebDAV)
508 Loop Detected (WebDAV)
509 Bandwidth Limit Exceeded (Apache)
510 Not Extended
511 Network Authentication Required
598 Network read timeout error
599 Network connect timeout error

Source: https://www.restapitutorial.com/httpstatuscodes.html



HTTP Request Body

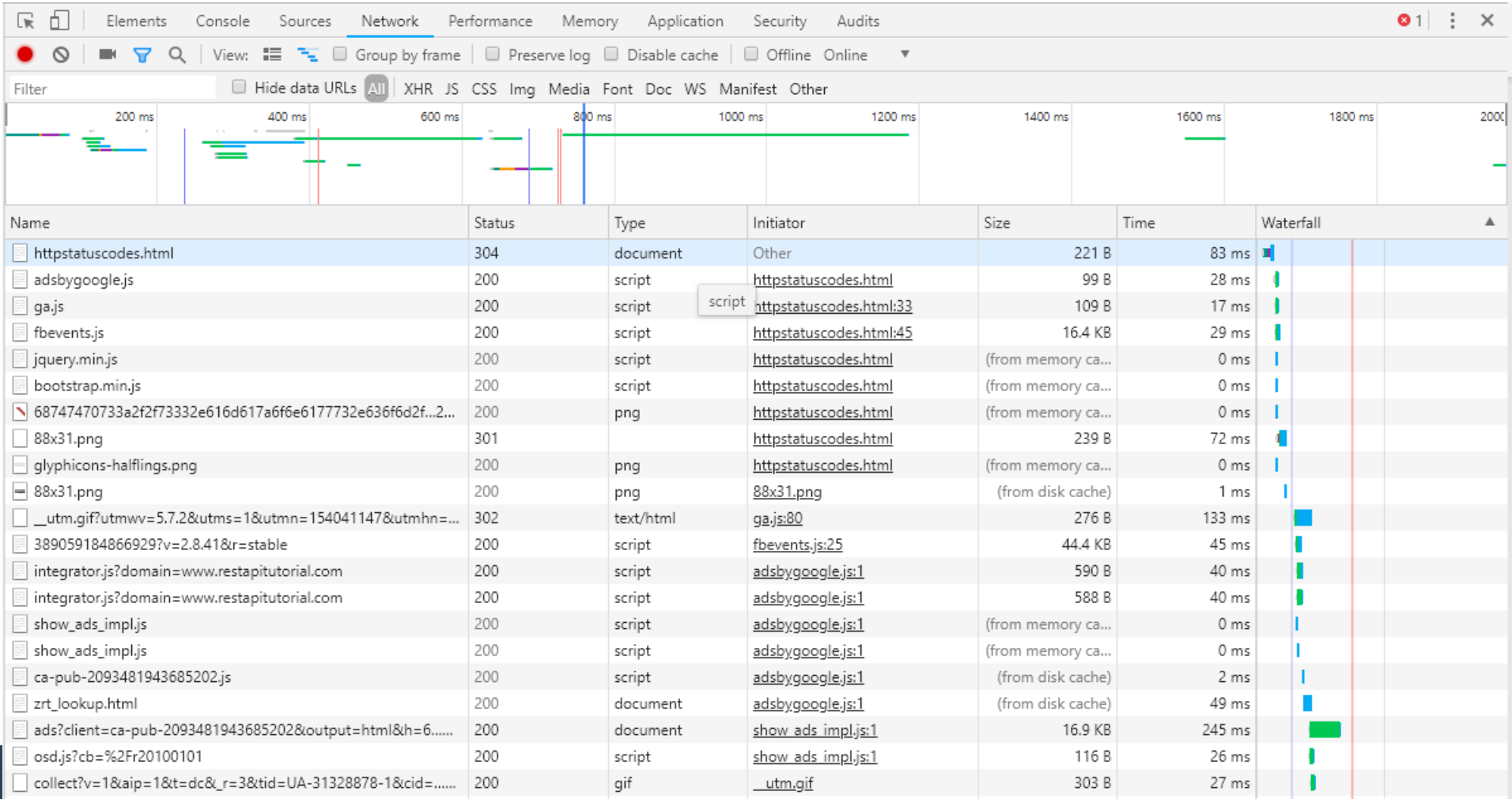
PUT & POST request typically have a body. HTML forms usually results in a POST request with a x-www-form-urlencoded type

```
POST /post HTTP/1.1
Host: url.org
User-Agent: Mozilla/5.0 (Windows NT 10.0;Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
Accept: */*
Content-Length: 18
Content-Type: application/x-www-form-urlencoded

user=bo&comment=Hi
```

HTTP in action

Let's try Chromes "network" tab



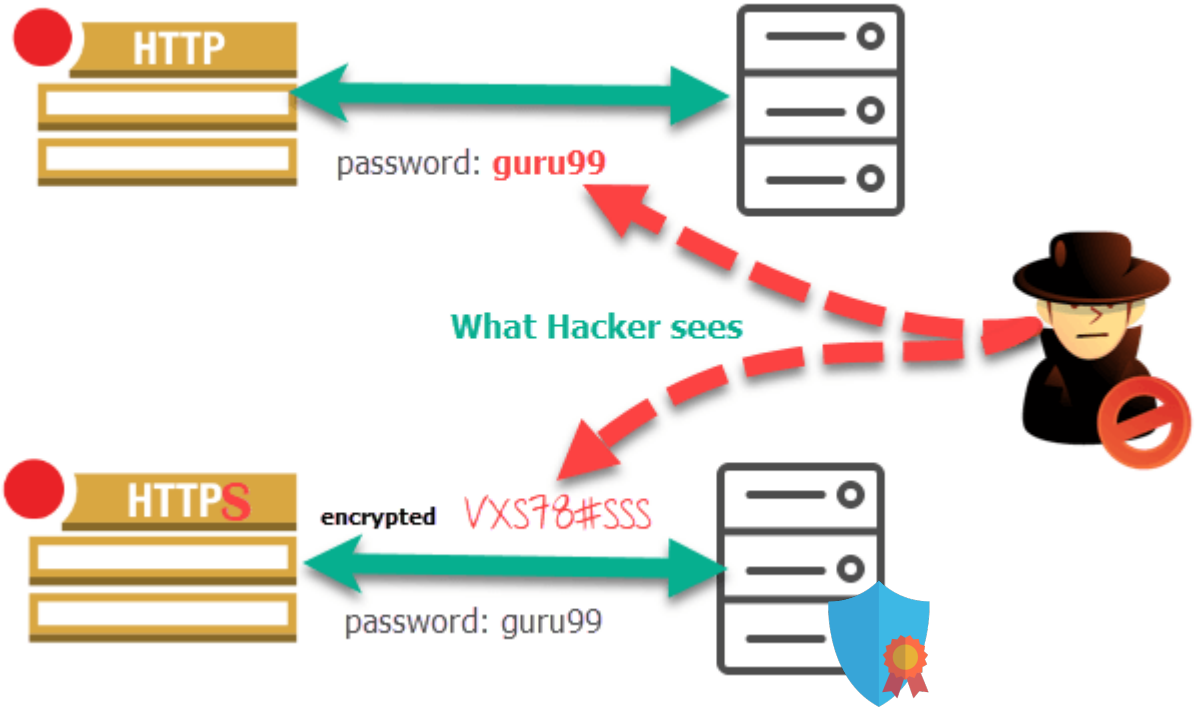
HTTP vs HTTPS

HyperText Transfer Protocol Secure:

- Means trusted SSL certificate is in place
  - Anyone can get one
  - TLS is an updated version, but we still call it SSL
  - Stands for Secure Socket Layer, Transport Layer Security
- Requests and responses are scrambled and encrypted
  - Stored data is not
- Uses port 443 instead of port 80
- Increases SEO ranking
- Browsers warn users not to use HTTP

  - https://www.guru99.com/difference-http-vs-https.html

HTTP vs HTTPS



Source: https://www.guru99.com/difference-http-vs-https.html

Other Protocols

- **FTP**
  - File transfer protocol
  - Default way of working with files on the server
- **SMTP**
  - Simple Mail Transfer Protocol
  - Default way of sending and receiving e-mails

The URL

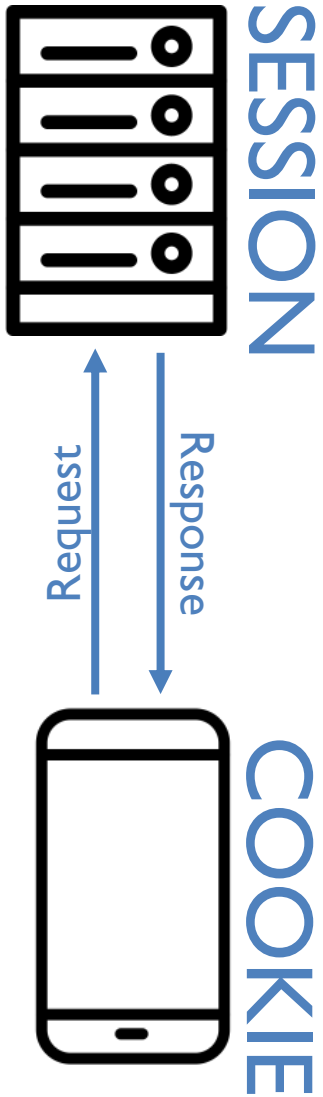
http://www.fuzzball.nu/Project/simplevote/index.html

- HTTP protocol
- World-wide-web (indicates website)
- Domain name
- Top-level domain
- Mapping on server
- Specific file
  - If this address is a file path, the index.html file will be loaded automatically if it exists
  - This can easily be disabled

Sessions & Cookies

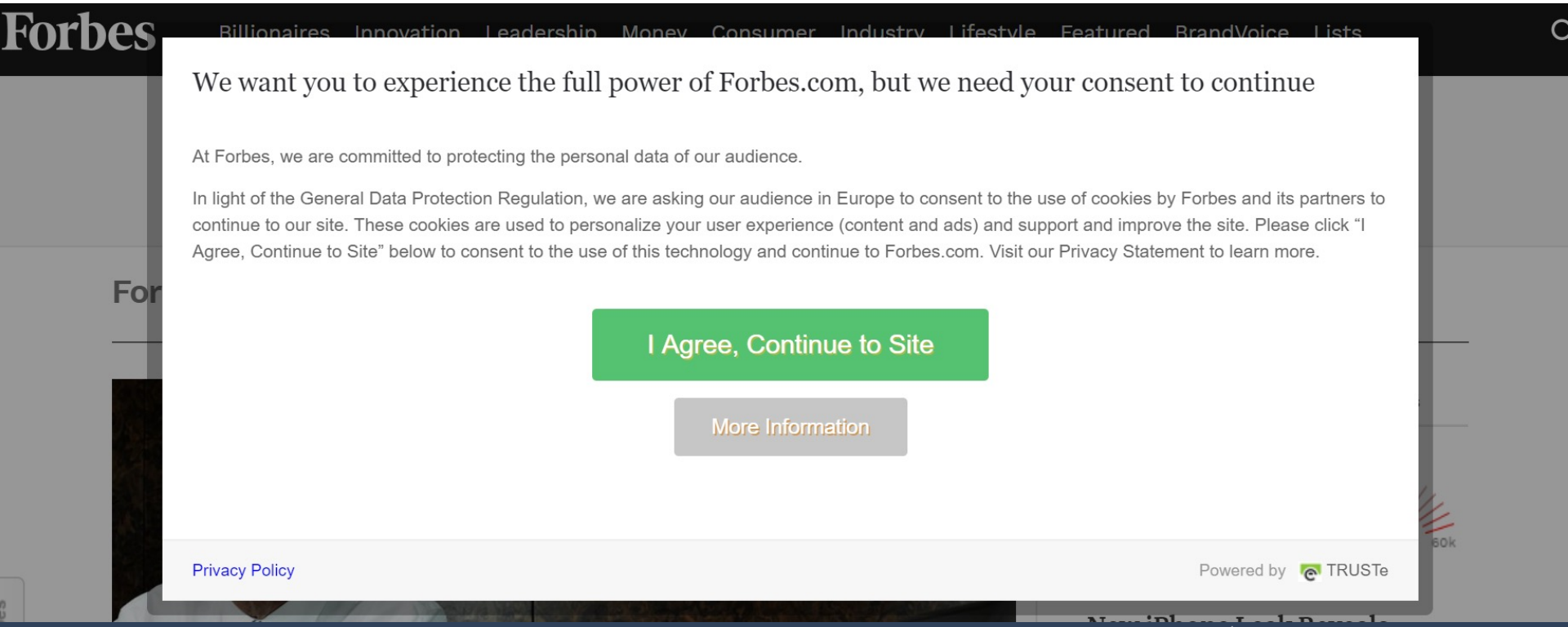
- Sessions are stored on the server
  - invisible to the user
    - Reliable
- Cookies are stores on the client and thus
  - Visible and editable by
    - User
    - Every other website
    - Commonly used by spyware

- Sessions expire when the browser is closed
- Session timeouts can be set
- Session ID's are stored in cookies



Sessions & Cookies

- Sites must warn you about Cookies
- Why?
- Why not Sessions?



Break!

And afterwards, we will continue with:

PHP

PHP

- Personal Home Page Hypertext Preprocessor
- Backend language – runs server side
- Defines the content that is sent to the Client

How do we run it?

- Open the folder that contains your project in Terminal or CMD
- Use the following command:
  - Php –S localhost:8080
    - Localhost can be

PHP

- A PHP file has the extension .php
- .php files are interpreted by an Apache server
- index.php is loaded by default
- .php files can be seen as HTML files with a special PHP element <?php ?>

```
<?php
/* This is a PHP comment */
// This is also a PHP comment
# And so is this
?>
```

PHP

- The PHP is functional and will not be present in the generated HTML
- To write something that should be included in the HTML
  - Use echo
    - echo 'Hello World!';
  - Or exit the PHP (?>)

PHP

- Combined with HTML example:

```
<h1><?php echo 'Hello World'; ?></h1>
```

PHP

- Although PHP variable names are case-sensitive, function names and keywords are not.
- PHP has Integer, Double, String (single or double quotes) and Boolean primitive types.
- Inside double quotes a variable is replaced by its value (this is called *variable interpolation*).
- PHP arrays can be viewed either as "ordinary" arrays with integer indexes, or as a list of key/value pairs.

PHP Variables

- PHP variables are noted with a dollar sign: $
- PHP variables are similar to JavaScript variables
  - The type is not set
  - The type can be changed by new assignment
- But there is a difference
  - PHP variables have no initializer

JavaScript:

```
var number = "2";
number = parseInt(number);
>> 2
```

PHP:

```
$number = "2";
$number = intval($number);
>> 2
```

Strings

- Strings can be enclosed in single or double brackets
- Double quotes are considered regular characters in a single-quote string and vice versa

```
echo ‘He said “Hello”.’;
```

```
echo “Don’t do it!“;
```

- This is also true for JavaScript!
- Escape characters are also as you might expect
  - \” \’ \n \r \t \$ \\

Strings and variables

- Strings can be concatenated with + or .
  - + is generally mathematical
  - . Is always concatenation

```
$a = 4;
$b = 5;
echo $a + $b;   >> 9
echo $a . $b;    >> 45
```

- . is considered best practice for concatenation

```
echo 'Hello, ' . $name;
```

- A variable can be resolved directly inside a double-quoted string – not a single-quotes one
  - This is called interpolation
- To avoid confusion, the variable can be wrapped in curly brackets

```
$name = "Manny";
echo 'We have a $name';       >> We have a $name
echo "We have a $name";       >> We have a Manny
echo "We have 2 $names";      >> ERROR
echo "We have 2 {$name}s";    >> We have 2 Mannys
```

Strings and variables

- Echo's and non-PHP is treated equally
- There is a convenience method for entering PHP to echo a variable

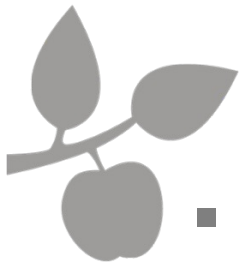
```
<?php echo "Hello <b>$name</b>";          >> Hello Manny
Hello <b><?php echo $name; ?></b>         >> Hello Manny
Hello <b><?=$name?></b>                   >> Hello Manny
```

Strings functions

- PHP has a lot of built-in functions and they are not tied to classes as you might expect.
- Complete list:
  - https://www.w3schools.com/php/php_ref_string.asp
- substr
- strlen
- substr
- substr_replace
- strtolower
- strtoupper
- htmlentities
- explode / str_split
- implode / join
- ltrim
- md5
  - Hashing method
- preg_match ($pattern, $string)
  - Regex matcher

Math functions

- Complete list:
  - https://www.w3schools.com/php/php_ref_math.asp
- abs: absolute value
- cos / acos
- sin / asin
- tan / atan
- sqrt : square root
- pow : x to the power of y
- rand : random
- floor : rounds float down to int
- round : rounds float to int
- pi



PHP Arrays

- Arrays in PHP can have key / value pairs

```
$array = array("key" => "value", "foo" => 3, "x");
```

- Values can be accessed via key or index:

```
$array[2]; >> "x"
```

```
$array["key"]; >> "value"
```

- To add an item to the array

```
$array[] = "new item";
```

- To add new key/value set

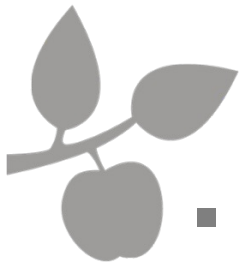
```
$array["new key"] = "new value";
```

- To remove a key/value set

```
unset($array["new key"]);
```

- To check if a value is set

```
isset($array["new key"]);
```



PHP Arrays

- To see all contents of a PHP array

```
print_r($array)
```

- Values can be accessed via key or index:

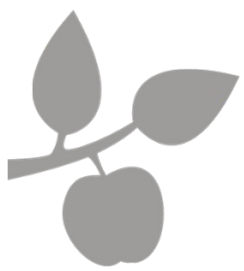
Functions

- Create and call a function
- As with JavaScript, we do not specify type

```
function sum($a, $b) {
  return $a + $b;
}

echo sum(2, 2);          >> 4
echo sum("2", "2")     >> 22
```

- Note that built-in functions are called directly
  - strlen($myString);



Expressions and evaluators

- Mathematical expressions are the same as Java:
  - + - * / %
- Usual comparisons are as expected, but we there are more:
  - == != < <= > >=
  - !== === match value and type
  - <> not equal, value only
  - <=> returns -1 0 or 1, depending on whether the left argument is lower than, equal to or higher than the right argument

If-Then-Else

- Can be written in two different ways, to match your HTML style:

```
if($name === "Manny") {
    echo 'Hi Mr. M';
} elseif ($name === "Jackie") {
    echo 'Hi Mrs. J';
} else if ($name === "Bo") {
    echo "Hi Mr. B";
} else {
    echo "Hi $name";
}
```

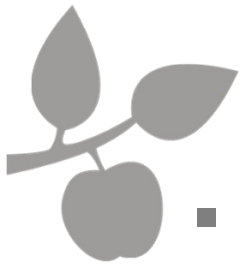
```
if($name === "Manny") : ?>
    Hi Mr. M
<?php elseif ($name === "Jackie") : ?>
    Hi Mrs. J
<?php elseif ($name === "Bo") : ?>
    Hi Mr. B
<?php else : ?>
    Hi <?=$name?>
<?php endif; ?>
```

Switch

- Can be written in two different ways, to match your HTML style:

```
switch($name) {
    case "Manny":
    echo "Hello Mr. M";
    break;
    case "Jackie":
    echo "Hello Mrs. J";
    break;
    default:
    echo "Hello $name";
}
```

```
switch($name) :
    case "Manny":
    echo "Hello Mr. M";
    break;
    case "Jackie":
    echo "Hello Mrs. J";
    break;
    default:
    echo "Hello $name";
endswitch;
```



Loops

- Can be written in two different ways, to match your HTML style:

```
for($i = 0; $i < sizeOf($names); $i++) {
  echo $names[$i] . ", ";
}
```

```
for($i = 0; $i < sizeOf($names); $i++) :
  echo $names[$i] . ", ";
endfor;
```

```
foreach ( $names as $key => $value ) {
  echo $key . " " . $value . ", ";
}
```

```
foreach ( $names as $key => $value ) :
  echo $key . ", " . $value;
endforeach;
```

```
foreach ( $names as $value ) {
  echo $value . ", ";
}
```

```
foreach ( $names as $value ) :
  echo $value;
endforeach;
```

Loops

- Can be written in two different ways, to match your HTML style:

```
$i = 0;
while($i < sizeof($names)) {
    echo $names[$i] . ", ";
    $i++;
}
```

```
$i = 0;
while($i < sizeof($names)) :
    echo $names[$i] . ", "
    $i++;
endwhile;
```

```
$i = 0;
do {
    echo $names[$i] . ", ";
    $i++;
} while ($i < sizeof($names));
```

PHP Super Globals

- PHP has a set of "Super Global" Variables
- They are accessible from anywhere, regardless of scope
- Remember that print_r() can show an arrays content!

- $GLOBALS
- $_SERVER
- $_REQUEST
- $_POST
- $_GET
- $_FILES
- $_COOKIE
- $_SESSION

$GLOBALS

- Contains all global variables from anywhere:

```php
<?php
$x = 75;
$y = 25;

function addition() {
    $GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];
}

addition();
echo $z;
```

>> 100

$_SERVER

- Holds information about headers, paths, and script locations
- We will use this to manipulate URL's later on

$_SERVER['PHP_SELF']	Returns the filename of the currently executing script
$_SERVER['GATEWAY_INTERFACE']	Returns the version of the Common Gateway Interface (CGI) the server is using
$_SERVER['SERVER_ADDR']	Returns the IP address of the host server
$_SERVER['SERVER_NAME']	Returns the name of the host server (such as www.w3schools.com)
$_SERVER['SERVER_SOFTWARE']	Returns the server identification string (such as Apache/2.2.24)
$_SERVER['SERVER_PROTOCOL']	Returns the name and revision of the information protocol (such as HTTP/1.1)
$_SERVER['REQUEST_METHOD']	Returns the request method used to access the page (such as POST)
$_SERVER['REQUEST_TIME']	Returns the timestamp of the start of the request (such as 1377687496)
$_SERVER['QUERY_STRING']	Returns the query string if the page is accessed via a query string
$_SERVER['HTTP_ACCEPT']	Returns the Accept header from the current request
$_SERVER['HTTP_ACCEPT_CHARSET']	Returns the Accept_Charset header from the current request (such as utf-8,ISO-8859-1)
$_SERVER['HTTP_HOST']	Returns the Host header from the current request
$_SERVER['HTTP_REFERER']	Returns the complete URL of the page from which the current page was called

$_SERVER

- Holds information about headers, paths, and script locations
- We will use this to manipulate URL's later on

$_SERVER['HTTPS']	Is the script queried through a secure HTTP protocol
$_SERVER['REMOTE_ADDR']	Returns the IP address from where the user is viewing the current page
$_SERVER['REMOTE_HOST']	Returns the Host name from where the user is viewing the current page
$_SERVER['REMOTE_PORT']	Returns the port being used on the user's machine to communicate with the web server
$_SERVER['SCRIPT_FILENAME']	Returns the absolute pathname of the currently executing script
$_SERVER['SERVER_ADMIN']	Returns the value given to the SERVER_ADMIN directive in the web server configuration file (if your script runs on a virtual host, it will be the value defined for that virtual host) (such as someone@w3schools.com)
$_SERVER['SERVER_PORT']	Returns the port on the server machine being used by the web server for communication (such as 80)
$_SERVER['SERVER_SIGNATURE']	Returns the server version and virtual host name which are added to server-generated pages
$_SERVER['PATH_TRANSLATED']	Returns the file system based path to the current script
$_SERVER['SCRIPT_NAME']	Returns the path of the current script
$_SERVER['SCRIPT_URI']	Returns the URI of the current page

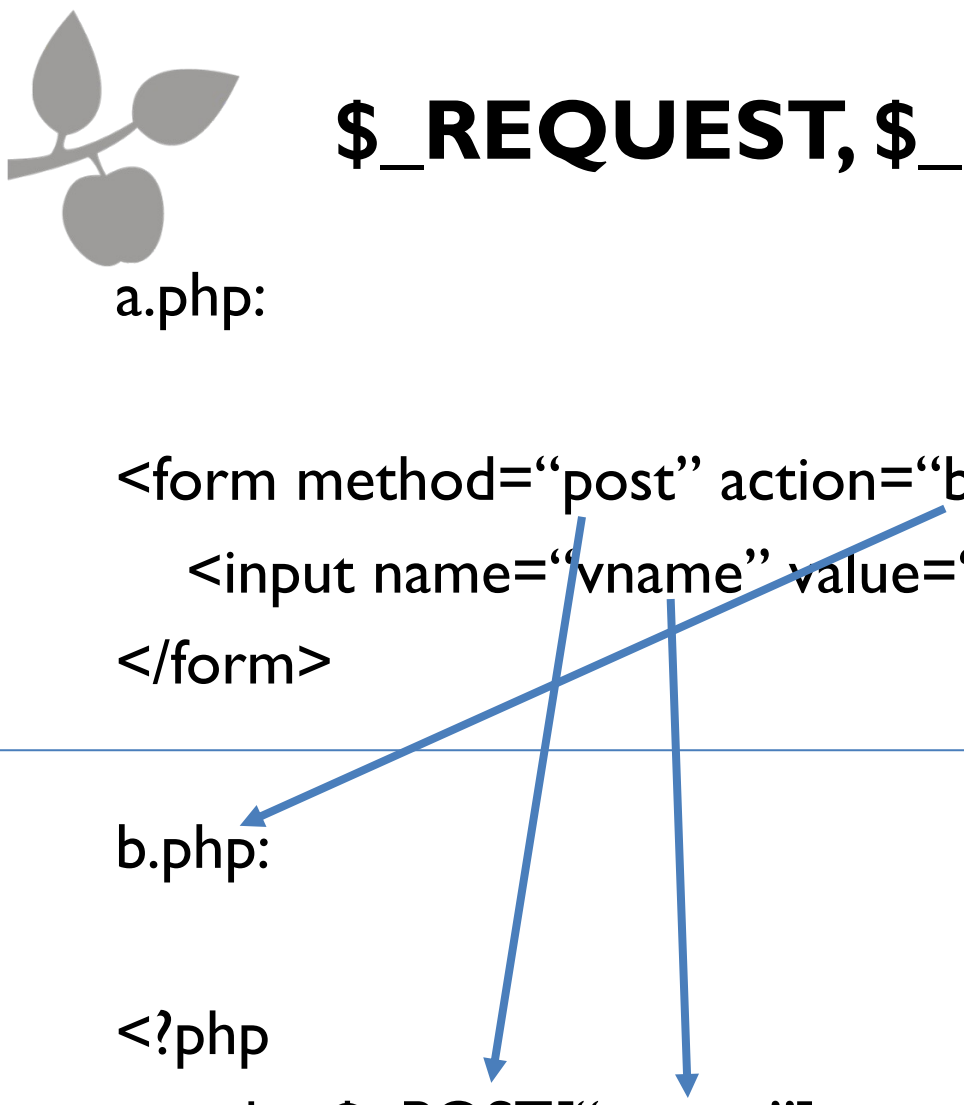
$_REQUEST, $_GET & $_POST

- Used to collect data after HTML form is submitted
- $_GET contains the content of a request when the HTTP request type is GET
- $_POST contains the content of a request when the HTTP request type is POST
- $_REQUEST contains the content of a request no matter which HTTP method was used
- Use $_SERVER[“REQUEST_METHOD”] to find out which method was used

$_REQUEST, $_GET & $_POST

a.php:

```
<form method="post" action="b.php">
    <input name="vname" value="hello"/>
</form>
```



b.php:

```
<?php
    echo $_POST["vname"];          >> hello
    echo $_REQUEST["vname"];       >> hello
    echo $_GET["vname"];           >>
```

$_FILES

- $_FILES contains files and file metadata from a request
  - Full example: https://www.w3schools.com/php/php_file_upload.asp
- You can tailor this example for your projects
  - I expect more than just a copy
  - We will discuss that part further next week with databases
- Check files size, file extension, mime type

- Let's take a look!

Filesystem

- PHP is the functional gateway between the clients and the server
  - Filesystem access
  - Database access

  - Create, rename and delete files and folders
  - PHP filesystem functions:
    - https://www.w3schools.com/php/php_ref_filesystem.asp

  - Let's have a look

- More next week!

$_COOKIE

- Used to store information in a cookie on the client machine
- Remember that anything inside a cookie can be altered by a user!

$_SESSION

- Contains information specific to a user
- A cookie shares an ID, so the session can identify the user and select the right session
- The data in a session is stored on the server
- The user does not have direct access to the data in a session
- When you create your login systems, you can set something like
  - $_SESSION[“logged_in”] = true;
- And check it with something like
  - If(isset($_SESSION[“logged_in”] && $_SESSION[“logged_in”])
- We will get back to logging in when we talk about databases next week

$_SESSION

- A session needs to be initialized

```
if (session_status() == PHP_SESSION_NONE) {
    session_start();
}
```

Filter Sanitize

- You can sanitize your request data to make sure it is the format you expect and not malicious
  - filter_input(INPUT_GET, "email", FILTER_VALIDATE_EMAIL)
- Complete list of sanitize filters here
  - http://php.net/manual/en/filter.filters.sanitize.php
- I expect all user input to be sanitized
- Let's have a look at that list

HTML Entities

- Sometimes you may want to accept codes, but in most cases, you will not
- To change a request string into using HTML Entities, use the method htmlentities
  - htmlentities($_POST[“username”])
  - specialchars($_POST[“username”])
- When you both sanitize and use htmlentities, you sanitize first!
- This is used to avoid XSS (Cross Site Scripting)
- Let’s try it!

include and require

- The purpose of PHP is to build the HTML the server should send to the client
- Sometimes a chunk of code should be repeated
- include creates warning if it fails, require throws fatal error
- include_once and require_once only includes if content has not already been included

- include ‘myFile.php’;
- include_once ‘myFile.php’;
- require ‘myFile.php’;
- require_once ‘myFile.php’;

Debugging and development

- During development, put this PHP function call at the beginning of your code to receive error reports.
  - `error_reporting(E_ALL)`
- When you are happy with the script, turn off error reporting
  - `error_reporting(0).`

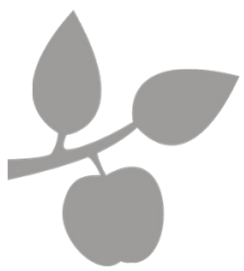
Exercise

- You can use this exercise in your first assignment
- Create an HTML form that can POST a username and a password
- For now, just check the inputs with hardcoded values
  - if($_POST[“username”] == “john”)
- When you are logged in, there should be an option to log out
- When you are not logged in, the HTML form should be included

AJAX

- We use JavaScript to change a page after it has loaded
- This might be done by retrieving more data from the server
- That is done with AJAX
  - Full example: https://www.w3schools.com/php/php_ajax_php.asp

```
var xmlhttp = new XMLHttpRequest();
xmlhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
        document.getElementById("txtHint").innerHTML = this.responseText;
    }
};
xmlhttp.open("GET", "gethint.php?q=" + str, true);
xmlhttp.send();
```



Exercise

- Make an AJAX call that gets a random number from the server every second and puts it on the page

- Javascript hint:
  - setInterval

- PHP hint:
  - rand()

Next time we talk PHP

- Objects
- JSON
- XML
- .htaccess
- Tokens
- API's
  - Creation
  - Consumption

TRY IT OUT!

Next time, we will continue with:

MySQL

Please install MariaDB if you have not already