howest
hogeschool

Server-Side Scripting Intro

Web, Mobile and Security

Frédéric Vlummens

Agenda

- What is server-side scripting?
- HTTP
  - Request – Response
  - Request methods
  - Headers
  - Status codes
- Network ports
- Webservers
- Introduction to PHP

Let’s start with an exercise

- We want to write a Facebook wall-like application…
- Let’s try using what we’ve learned so far

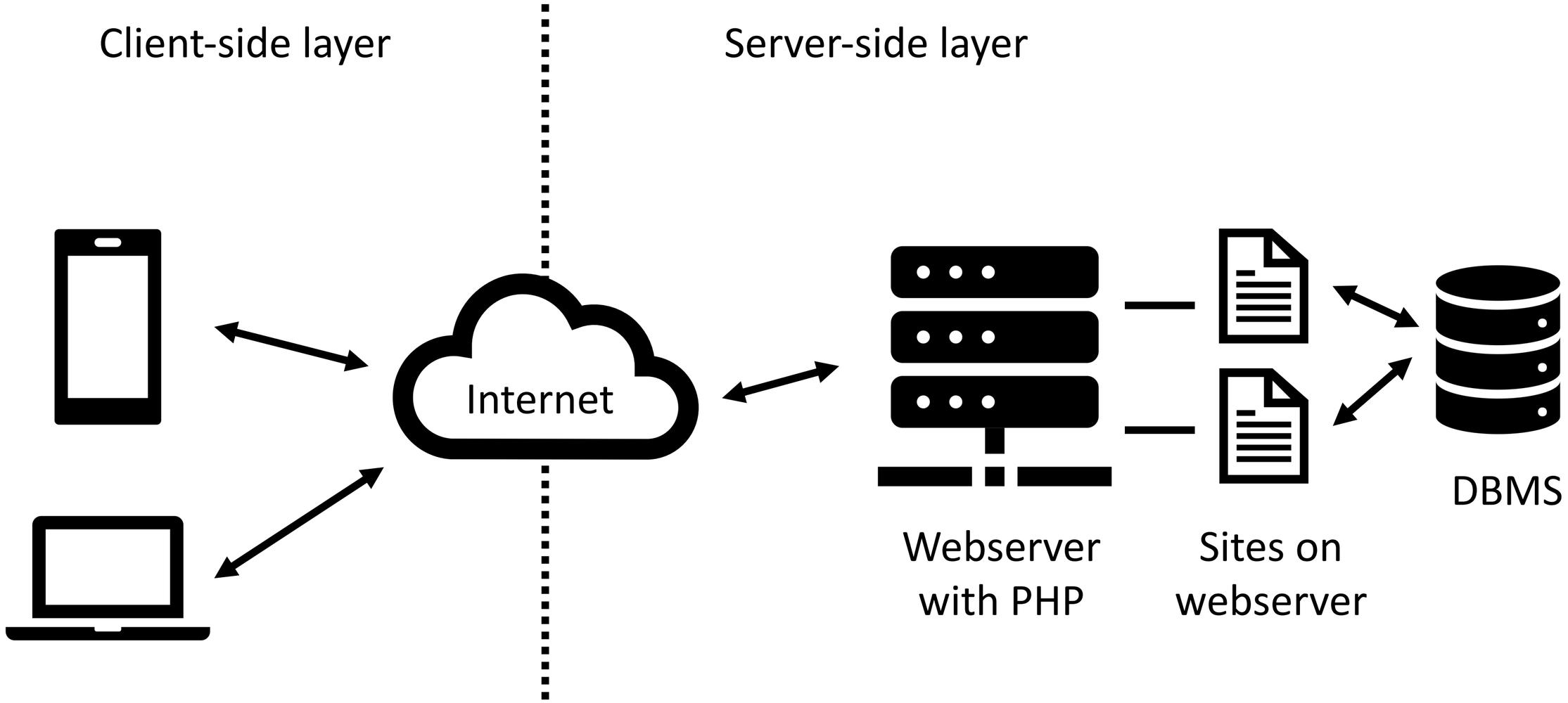
Facebook wall: solution

- Build a website using HTML, JavaScript and Local Storage
- Problem: the site is confined to one computer and one browser
- We need something that is available on all computers connected to the internet and all browsers
- Solution: server-side scripting

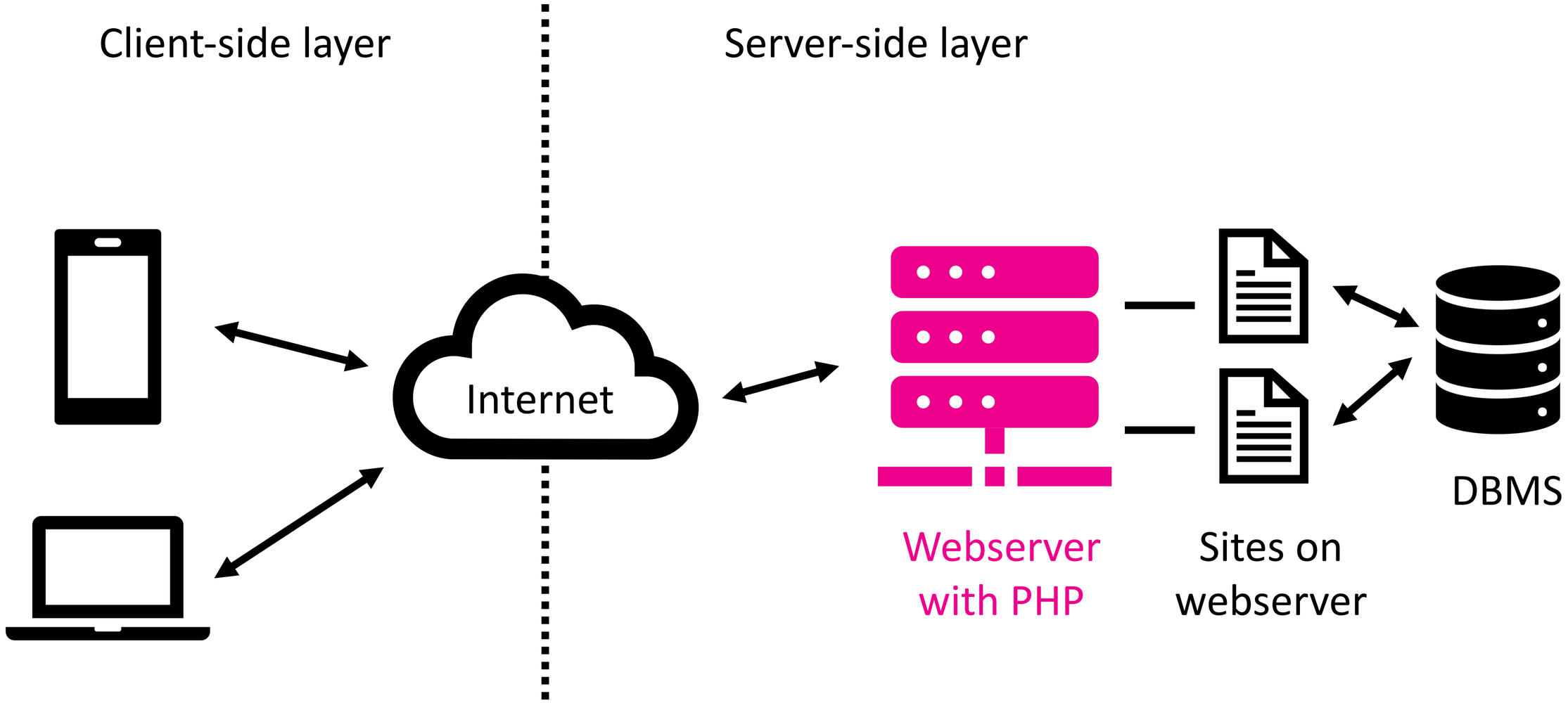
What is server-side scripting?

- Common technique used in web development
- Scripts (programs) run on the web server
- The scripts generate custom HTML (or even other files, such as CSS or images) that is sent to the browser
- The browser interprets the HTML it receives as if it were a static page
- This raises quite a few questions:
  - How does this transmission of data work?
  - What is a web server?
  - …

Server-side scripting



Server-side scripting



Web server

- What is it? A machine? Software?
- Answer: both
- Machine
  - Hosts the files on its hard disks
  - Is connected to the internet
- Software
  - HTTP server
  - Processes incoming request
  - Sends response to client
- In this course: web server = software

Overview of popular web server software

- Several popular products:
  - IIS
  - NGINX
  - Apache
  - Tomcat
  - Glassfish
  - …
- Some are general purpose web servers, others more specific
- In this course: Apache running on Debian Linux
- A virtual machine with Debian Linux, Apache and PHP (incl. Laravel) has been prepared
- Will be installed and used in various labs for this module

HTTP

HTTP

=

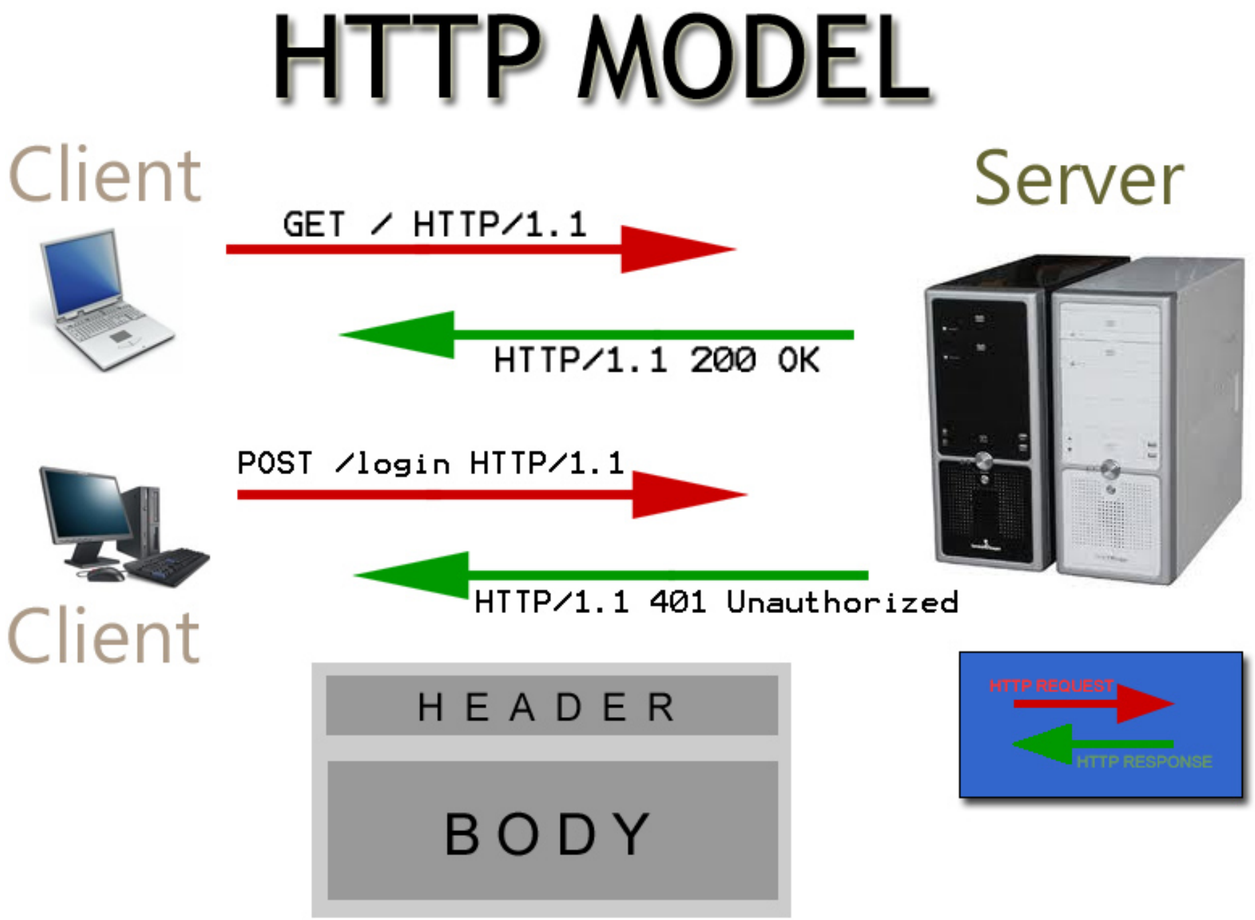
HyperText Transfer Protocol

howest hogeschool

HTTP

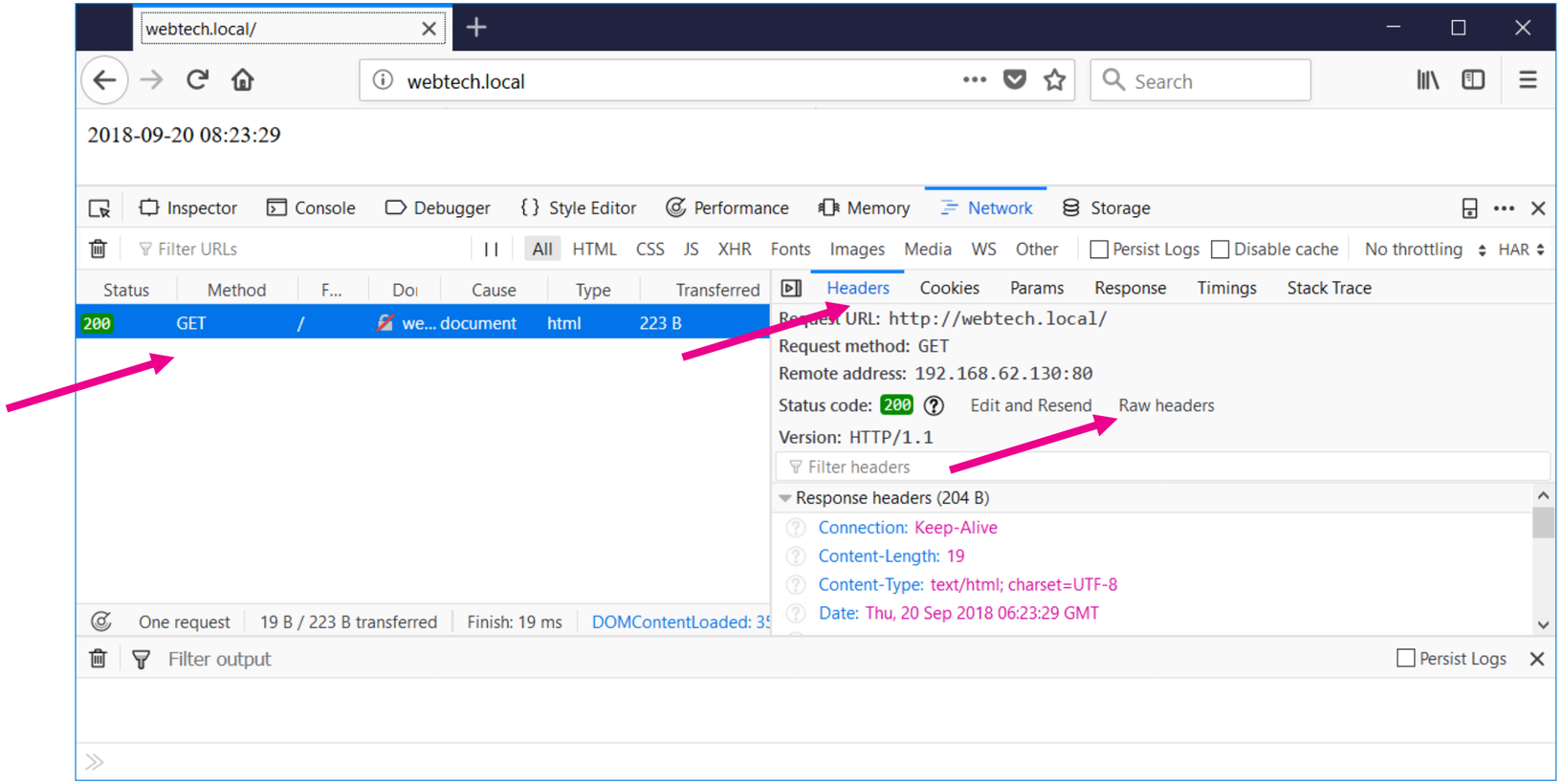
- HyperText Transfer Protocol
- Hypertext: structured text using hyperlinks between nodes
- HTTP is an application protocol that transfers said text
- It is the standard used to access the web

Request – Response



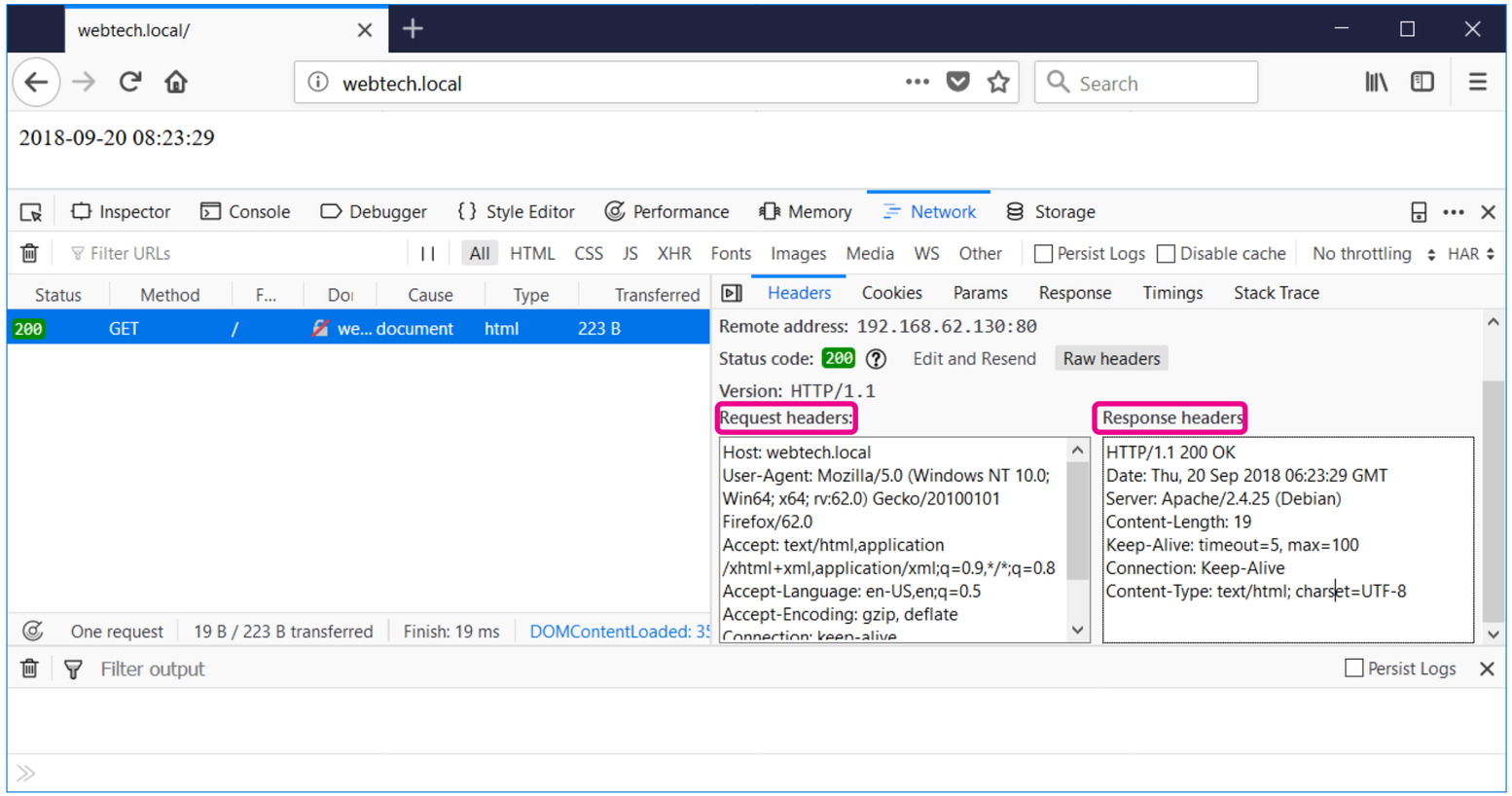
Viewing the Request and Response

- Using Mozilla Firefox Developer Tools
- Network tab

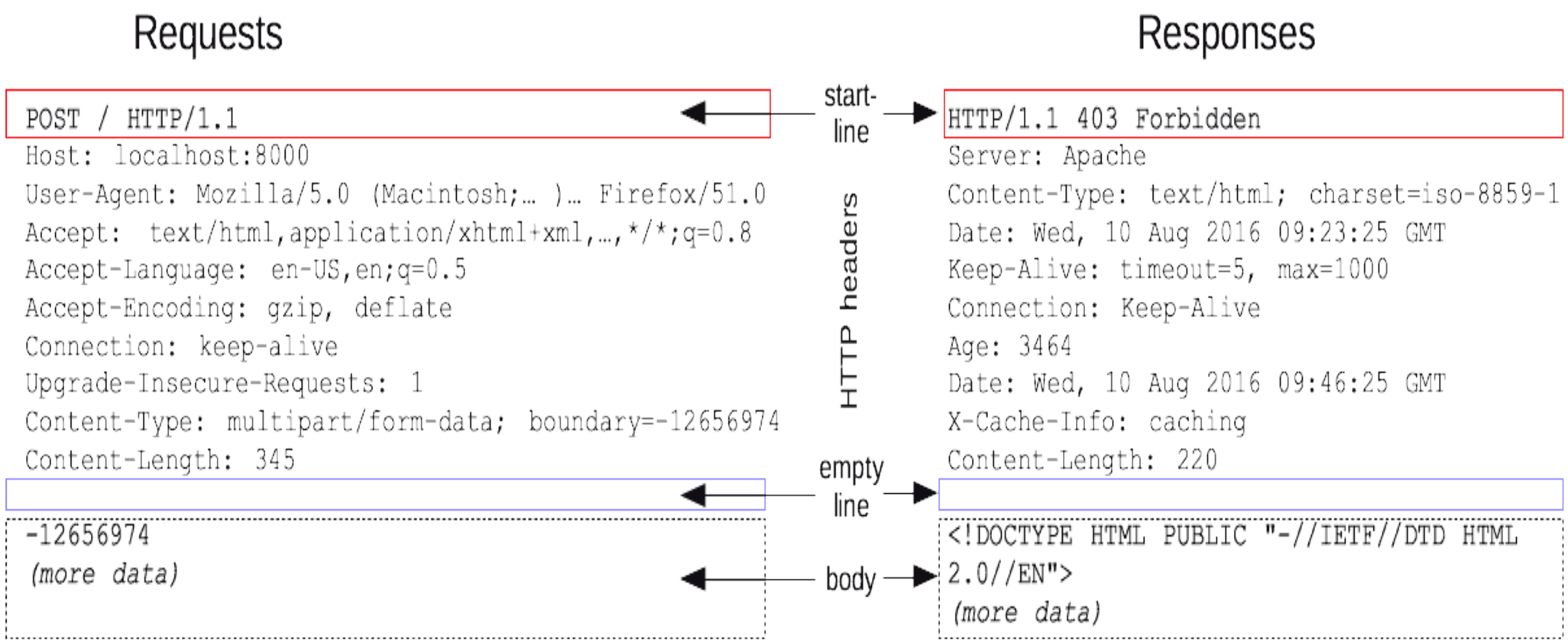


Viewing the Request and Response

- Using Mozilla Firefox Developer Tools
- Network tab



Request and Response



Using PuTTY or telnet to issue a request

webtech.local - PuTTY

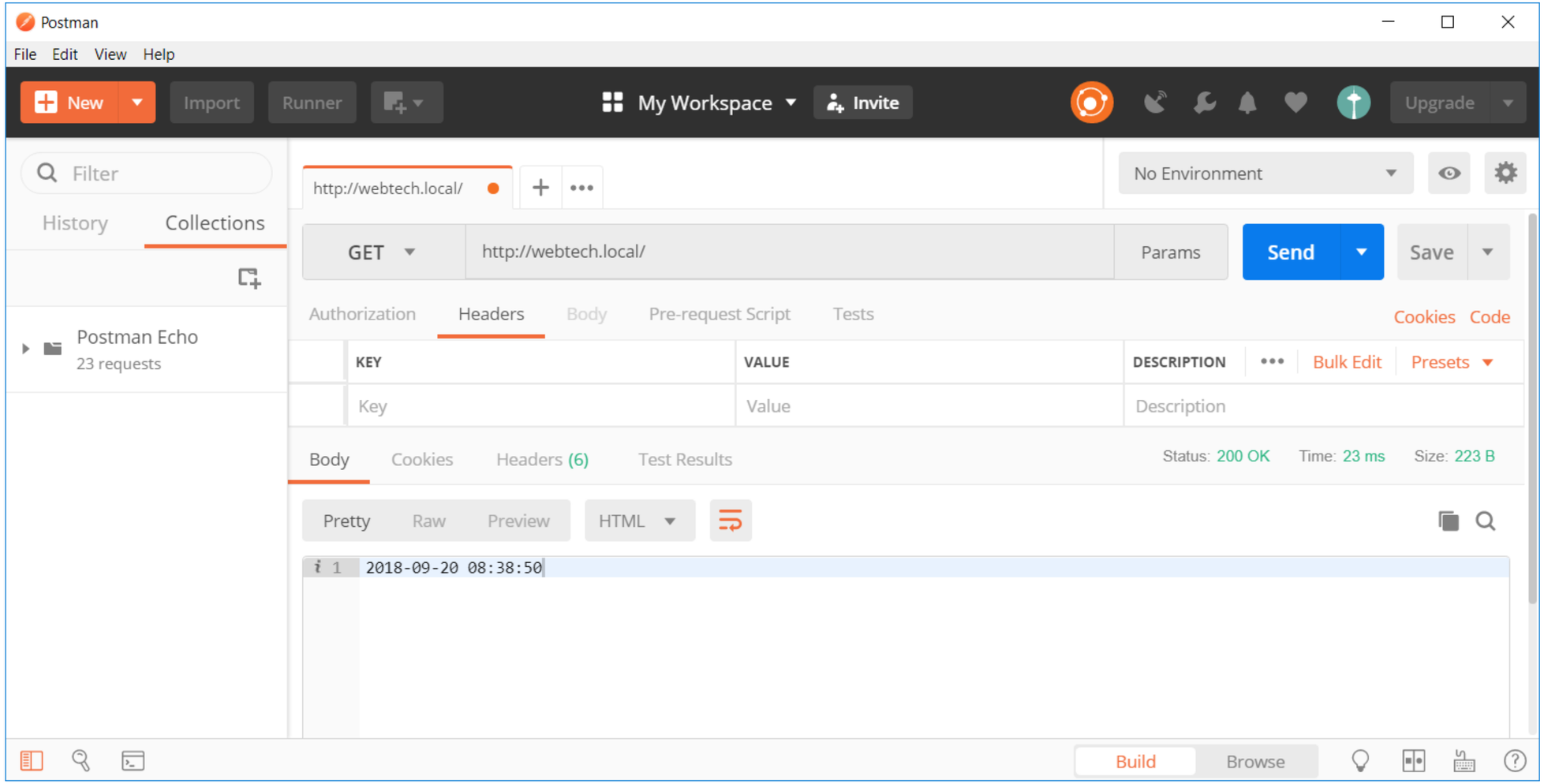
```
GET /index.php HTTP/1.1
Host: webtech.local
User-Agent: PuTTY

HTTP/1.1 200 OK
Date: Thu, 20 Sep 2018 06:54:20 GMT
Server: Apache/2.4.25 (Debian)
Content-Length: 19
Content-Type: text/html; charset=UTF-8

2018-09-20 08:54:27
```


howest hogeschool

Using Postman to issue a request



Request method

- Indicates the desired action for a given resource
- Made up of one noun or verb
- Also called *HTTP Verbs*
- Non-exhaustive list
  - GET
  - POST
  - PUT
  - DELETE
  - HEAD
  - OPTIONS

GET

- The HTTP GET method requests a representation of the specified resource.
- Requests using GET should only retrieve data.

```
GET /index.php HTTP/1.1
Host: webtech.local
User-Agent: Mozilla Firefox
```

POST

- The HTTP POST method sends data to the server.
- The type of the request body is indicated by header Content-Type.
- Usually used to submit form data.

```
POST /process.php HTTP/1.1
Host: webtech.local
Content-Type: application/x-www-form-urlencoded
Content-Length: 21

name=Frederic&team=TI
```

PUT

- PUT creates a new resource or replaces a representation of target resource with request payload.
- PUT is **idempotent**: calling it multiple times has no side effect (such as an order being submitted twice, which can happen with POST)

```
PUT /aboutme.html HTTP/1.1
Host: webtech.local
Content-type: text/html
Content-length: 17

<h1>About me</h1>
```

DELETE

- Deletes the specified resource

```
DELETE /old.html HTTP/1.1
```

HEAD

- Used to request headers returned should resource be requested using GET.
- Example use case: request the content length (size) and then decide whether to go ahead with download or not.

HEAD /largefile.zip HTTP/1.1

HEAD ▾ http://webtech.local/largefile.zip

Body | Cookies | Headers (9) | Test Results

Accept-Ranges → bytes

Connection → Keep-Alive

Content-Length → 200000000

Content-Type → application/zip

Date → Thu, 20 Sep 2018 13:13:53 GMT

ETag → "bebc200-5764d4a886a2d"

Keep-Alive → timeout=5, max=100

Last-Modified → Thu, 20 Sep 2018 13:13:38 GMT

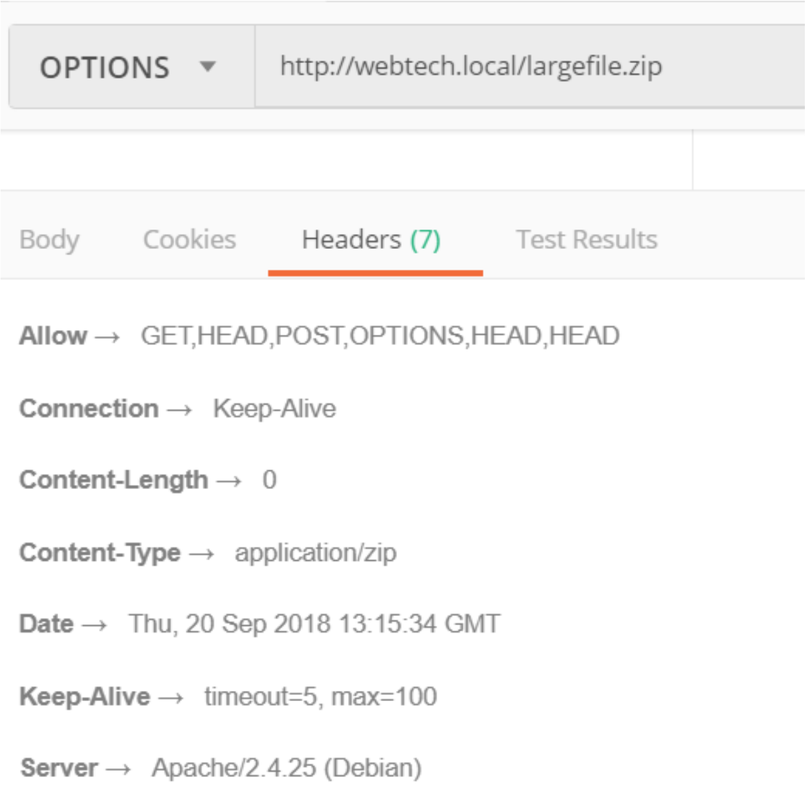
Server → Apache/2.4.25 (Debian)

howest hogeschool

OPTIONS

- Used to determine which request methods are supported by server or specific resource.

OPTIONS /largefile.zip HTTP/1.1



HTTP headers

- Allow client and server to pass additional information with request or response.
- Request header format: **header-name: value** (case insensitive)
- Example:

```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept: text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: multipart/form-data; boundary=-12656974
Content-Length: 345

-12656974
(more data)
```

Request headers

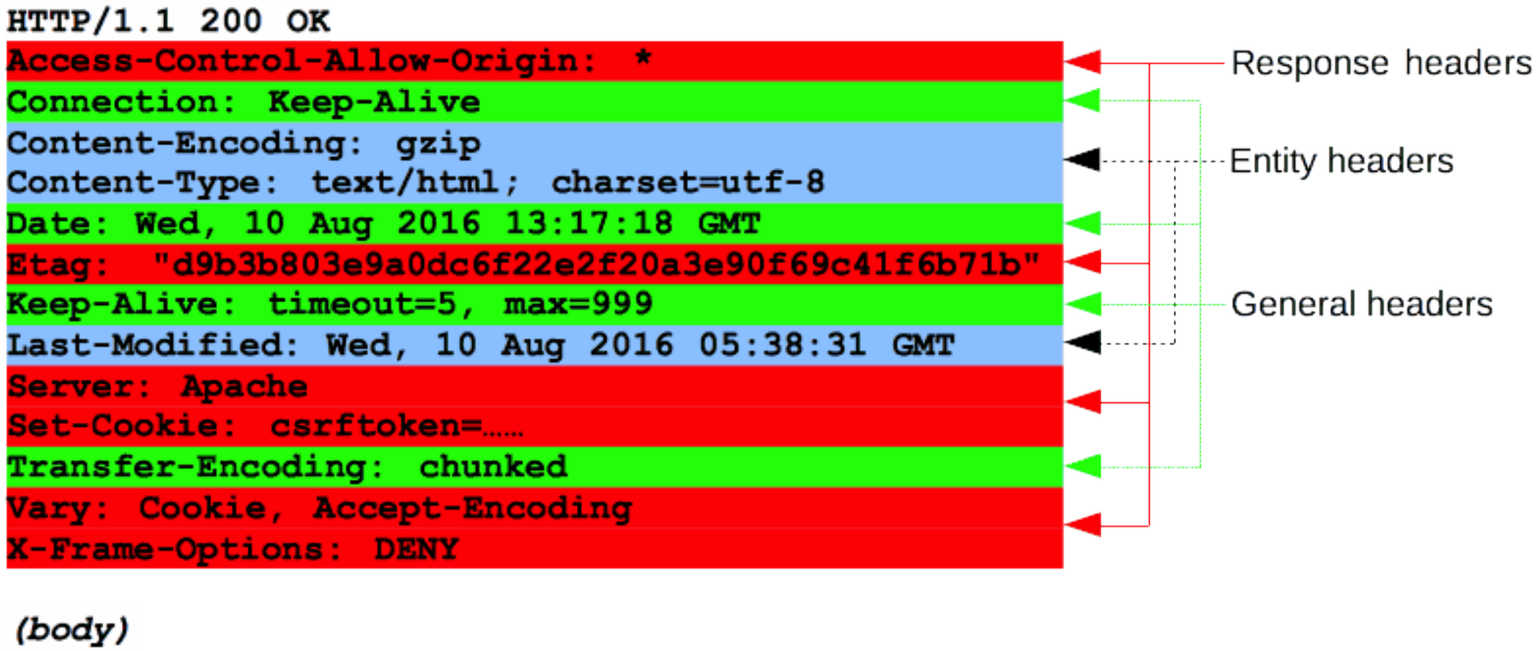
General headers

Entity headers

howest hogeschool

HTTP headers

- Allow client and server to pass additional information with request or response.
- Response header format: **header-name: value** (case insensitive)
- Example:



HTTP status codes

- Sent back to client
- Indicates whether request was completed successfully or not
- 3 digit codes
- Divided into 5 categories:
  - Informational (1##)
  - Successful (2##)
  - Redirects (3##)
  - Client errors (4##)
  - Server errors (5##)

Which status codes do you know?

Common HTTP status codes

- 200 OK
- 201 CREATED
- 400 BAD REQUEST
- 401 UNAUTHORIZED
- 403 FORBIDDEN
- 404 NOT FOUND
- 500 INTERNAL SERVER ERROR

- For more information, see https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

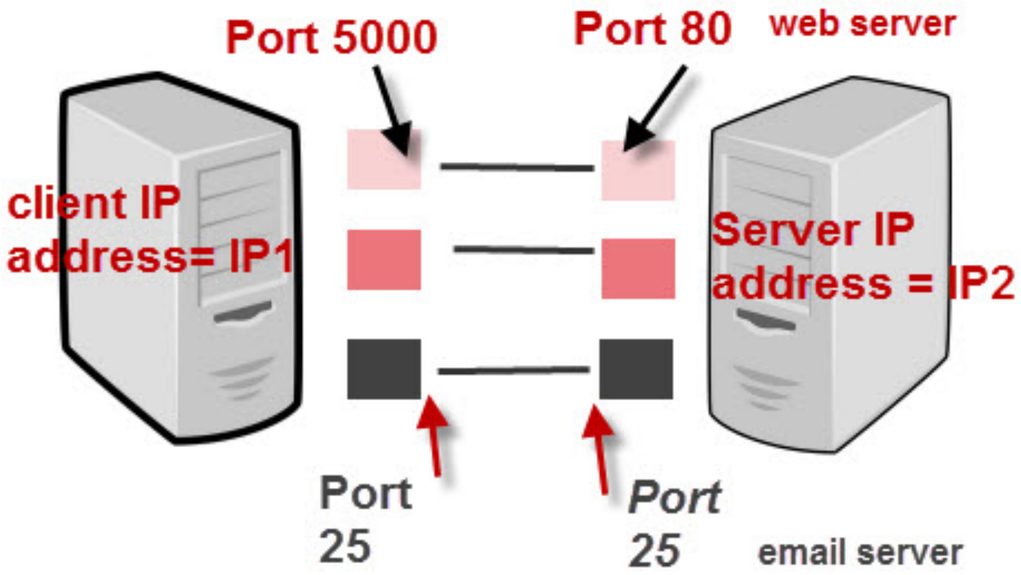
HTTP status code

```
GET /index.php HTTP/1.1
Host: webtech.local
User-Agent: MyBrowser
```

```
HTTP/1.1 200 OK
Date: Thu, 20 Sep 2018 13:34:01 GMT
Server: Apache/2.4.25 (Debian)
Content-Length: 19
Keep-Alive: timeout=5, max=98
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8
```

Network ports

- An **IP address** identifies a **computer**
- A **network port** identifies the **application/service** running on the computer.

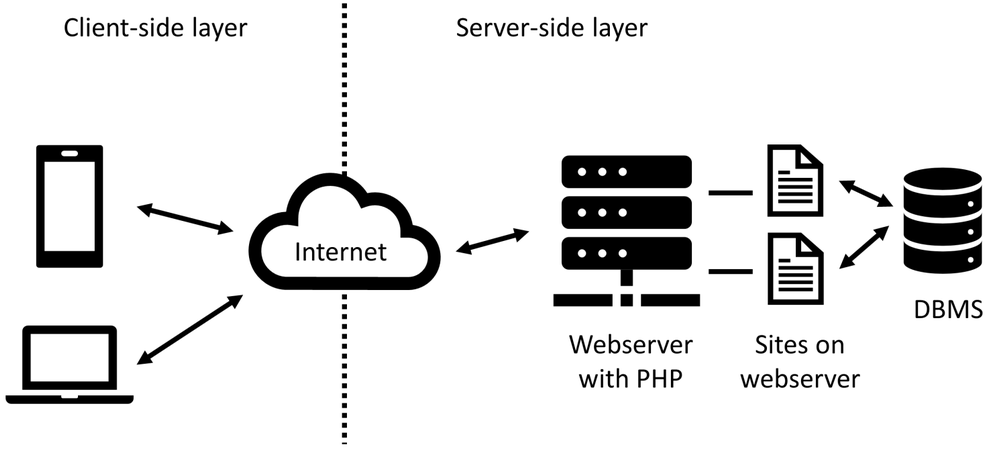


Some common network ports

Port	Description
21	File Transfer Protocol (FTP)
22	Secure Shell (SSH)
25	Simple Mail Transfer Protocol (SMTP)
80	HyperText Transfer Protocol (HTTP)
110	Post Office Protocol (POP3)
443	HTTP Secure (HTTPS)

Back to server-side scripting…

- Now we know what a web server is and how a browser and server communicate.
- Server side scripting = common technique used in web development.
- Scripts (programs) run on the web server.
- The scripts generate custom HTML (or even other files, such as CSS or images) that gets sent to the browser.
- The browser interprets the received HTML as if it where a static HTML page.



Introducing PHP

- PHP = PHP HyperText Processor (recursive acronym)
- Development started in 1994
- Various versions have since been introduced
- Today we are at PHP 7.3
- Interested in the history? https://en.wikipedia.org/wiki/PHP#History

A first look at a plain PHP script: index.php

File Edit Selection View Go Debug Terminal Help • index.php - Visual Studio Code

index.php

C: > xampp > htdocs > index.php

```php
<?php
    $a = 5;
    $b = 3;

    $sum = $a + $b;
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Hello world</title>
</head>
<body>

    <h1>Hello world!</h1>

    <p>The sum is: <?php echo $sum; ?></p>

</body>
</html>
```

PHP is embedded in HTML file using <?php and ?> delimiters

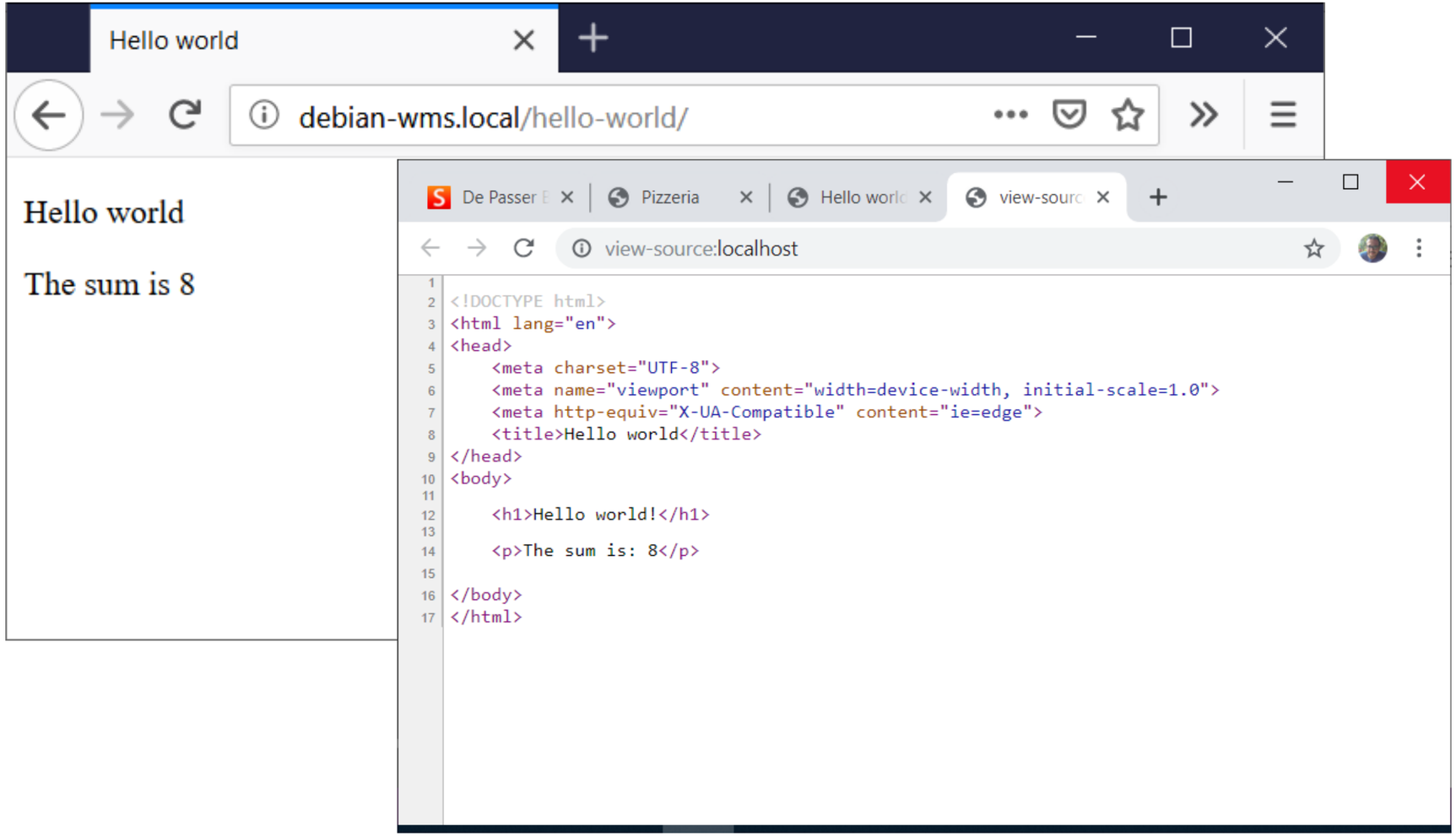
The top block is logic code

This smaller block is display code

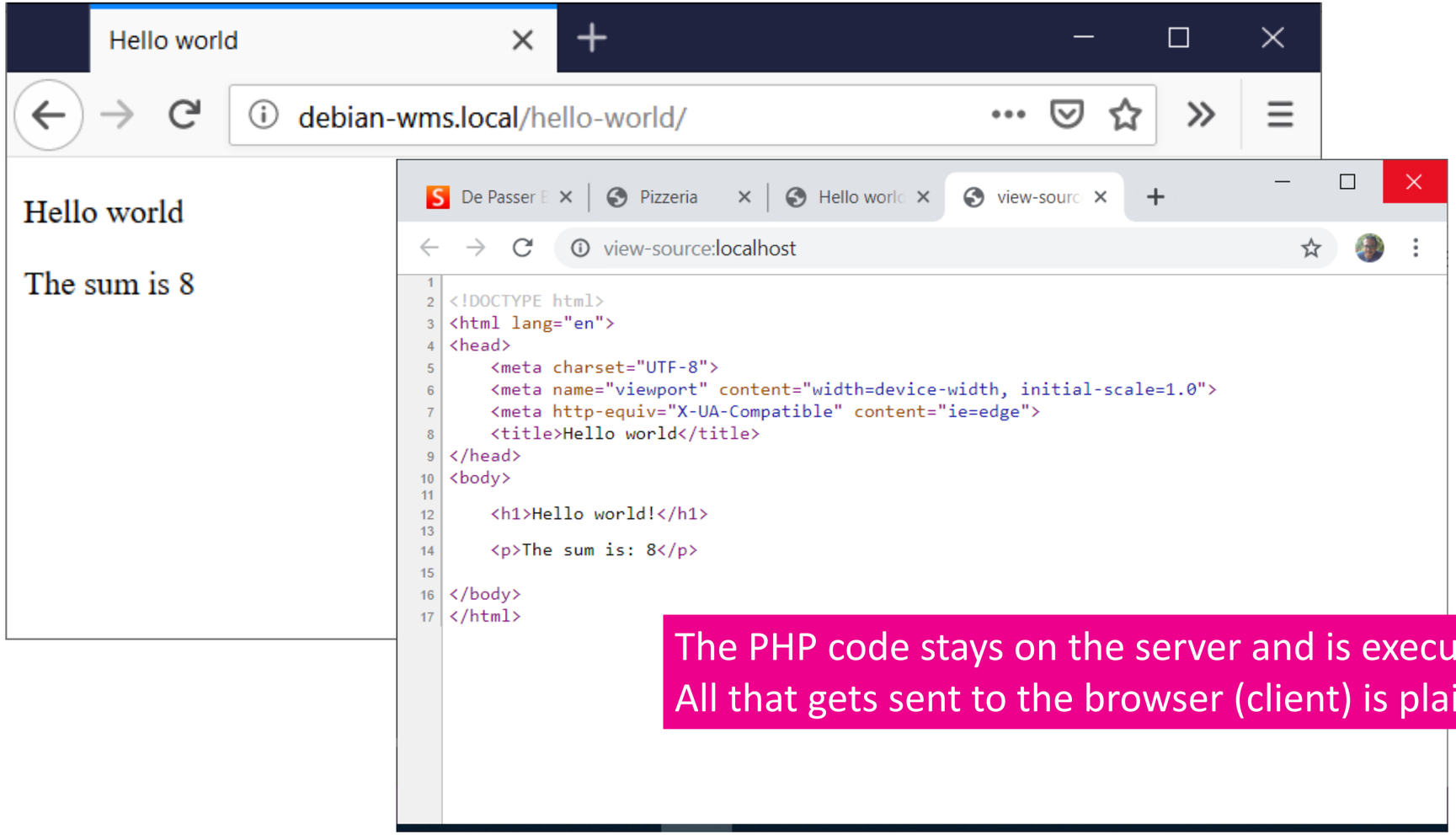
0 0 Ln 27, Col 1 Spaces: 4 UTF-8 CRLF PHP

howest hogeschool

Requesting the page and executing the script

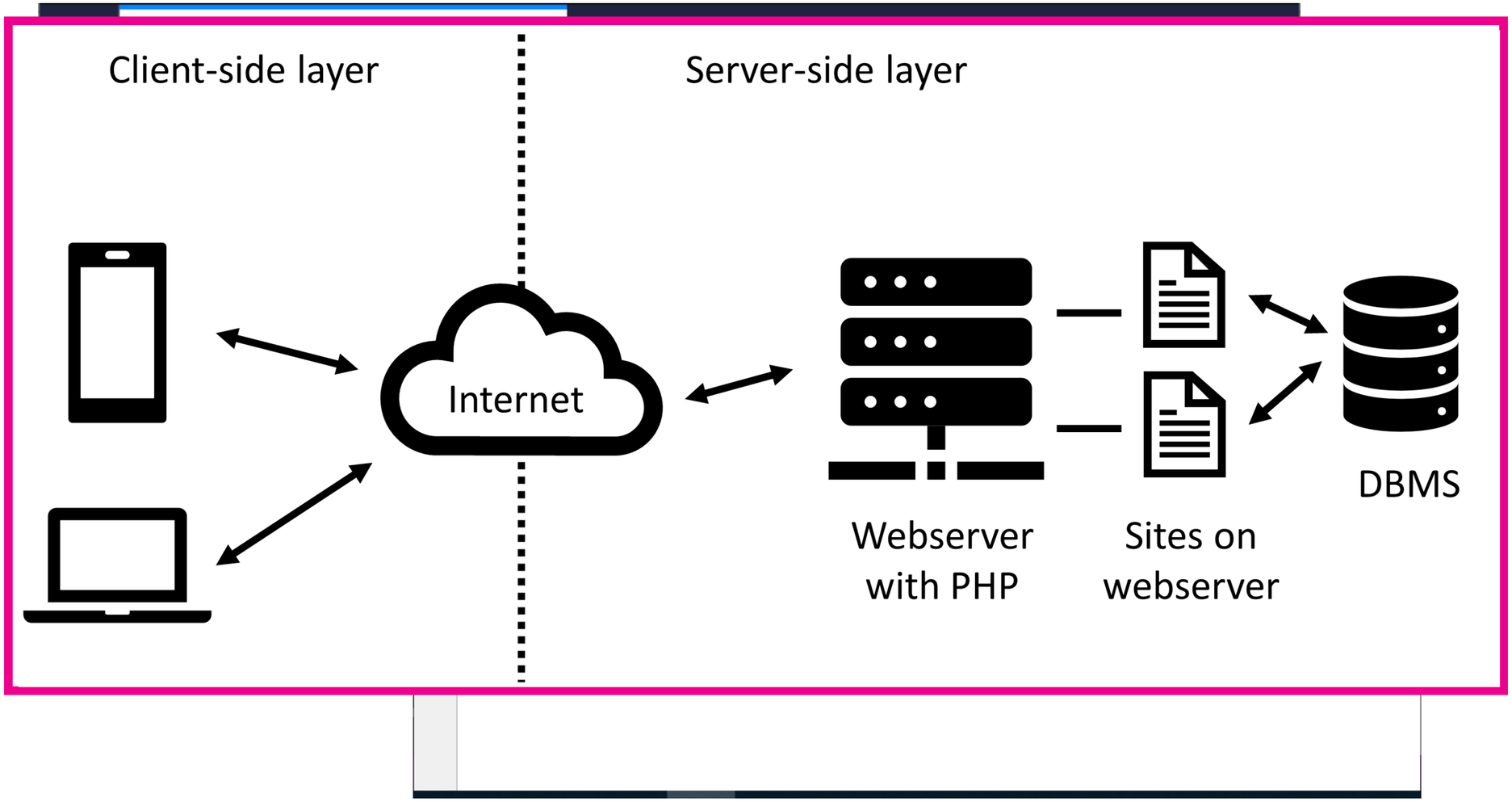


Requesting the page and executing the script



The PHP code stays on the server and is executed there.
All that gets sent to the browser (client) is plain HTML.

Requesting the page and executing the script



About PHP syntax

- Different from Python, more or less comparable to Java and JavaScript
- Excellent reference material: http://php.net/manual/en/langref.php
- Basis control structures:

  if – else – elseif
  for
  while
  …

  http://php.net/manual/en/language.control-structures.php

Key aspects of PHP syntax

- Variables start with a $ sign ⇔ Java, JavaScript and Python
- Functions are defined using the **function** keyword (idem as JavaScript)

```
<?php
    function calc_avg($a, $b) {
        return ($a + $b) / 2;
    }

    $x = 3;
    $y = 4;
?>

<p>The average of <?php echo $x; ?> and <?php echo $y;
?> is: <?php echo calc_avg($x, $y); ?></p>
```


howest hogeschool

Key aspects of PHP syntax

- Arrays are initialized using square brackets []
- You can loop over an array using **foreach**
- Here, array_push is an example of one of the many built-in functions of PHP

```
<?php
    $fruits = [];

    array_push($fruits, "apple");
    array_push($fruits, "pear");
    array_push($fruits, "banana");
?>

<ul>
   <?php foreach ($fruits as $fruit) { ?>
      <li><?php echo $fruit; ?></li>
   <?php } ?>
</ul>
```

Questions?