

http://



HTTP

- HTTP is the **protocol** that makes it possible for the web to work
- Protocol is the **rule of communication** between two or more entities
- HTTP works in a **request-response** manner
- An **HTTP server** serves its users by request made by the users

HTTP request



HTTP response

An **HTTP request
& response**
is made for each
resource



1 JS
2 CSS
4 images

1 Page

= **8** HTTP
requests & response

HTTP

- We can monitor HTTP requests and responses made and received by a browser using **developer tools** (press F12 or Ctrl+Shift+I on your browser!)



UNIVERSITAS BRAWIJAYA

Indonesia

WebMail

Prasetya

Dire

Home

About UB

Academic

Admission

Campus Life

Research

Work Units

search


**AKREDITASI A
BAN-PT**

NOMOR : 367/SK/BAN-PT/AK-SURV/PT/IX/2014

FEB UB


**THE ALLIANCE
ON BUSINESS EDUCATION
AND SCHOLARSHIP
FOR TOMORROW,,
a 21st century organization.**

FIA UB


**Asian Association
of Schools
of Business
International**

FTP UB

feeding the minds
that feed the world

Penguin

Elements Network Sources Timeline Profiles Resources Audits Console NetBeans

☐ Preserve log ☐ Disable cache

Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline	10.00 s	15.0
__utm.gif?utmwv=5.6.3&utms=1&utmz=383... www.google-analytics.com/r	GET	(failed) net::ERR_BL...		ub.ac.id/:607 Parser	0 B 0 B	4.64 s -			
ajax-loader.gif static.ub.ac.id/ub/asset/images	GET	200 OK	image/gif	ub.ac.id/:607 Parser	9.5 KB 9.2 KB	3.17 s 3.14 s			
akreditasi-2014_20141008102143.jpg ws.ub.ac.id/offweb/files/header_image	GET	200 OK	image/jpeg	ub.ac.id/:208 Parser	57.1 KB 56.9 KB	925 ms 207 ms			
akreditasi-feb.jpg static.ub.ac.id/static_files	GET	200 OK	image/jpeg	ub.ac.id/:189 Parser	15.6 KB 15.2 KB	1.02 s 947 ms			
akreditasi-fia.jpg static.ub.ac.id/static_files	GET	200 OK	image/jpeg	ub.ac.id/:194 Parser	10.5 KB 10.2 KB	1.73 s 1.68 s			
akreditasi-ftp.jpg static.ub.ac.id/static_files	GET	200 OK	image/jpeg	ub.ac.id/:199 Parser	10.5 KB 10.2 KB	4.34 s 4.30 s			
akreditasi-ub.jpg static.ub.ac.id/static_files	GET	200 OK	image/jpeg	ub.ac.id/:184 Parser	13.2 KB 12.9 KB	2.22 s 2.15 s			
arus_20141031161939.jpg ws.ub.ac.id/offweb/files/header_image	GET	200 OK	image/jpeg	ub.ac.id/:208 Parser	98.0 KB 97.7 KB	3.19 s 817 ms			
bannerweb_20150212132925.jpg	GET	200	image/jpeg	ub.ac.id/:208	217 KB	20.13 s			

38 requests | 1.4 MB transferred | 35.71 s (load: 35.73 s, DOMContentLoaded: 13.22 s)

Console Search Emulation Rendering



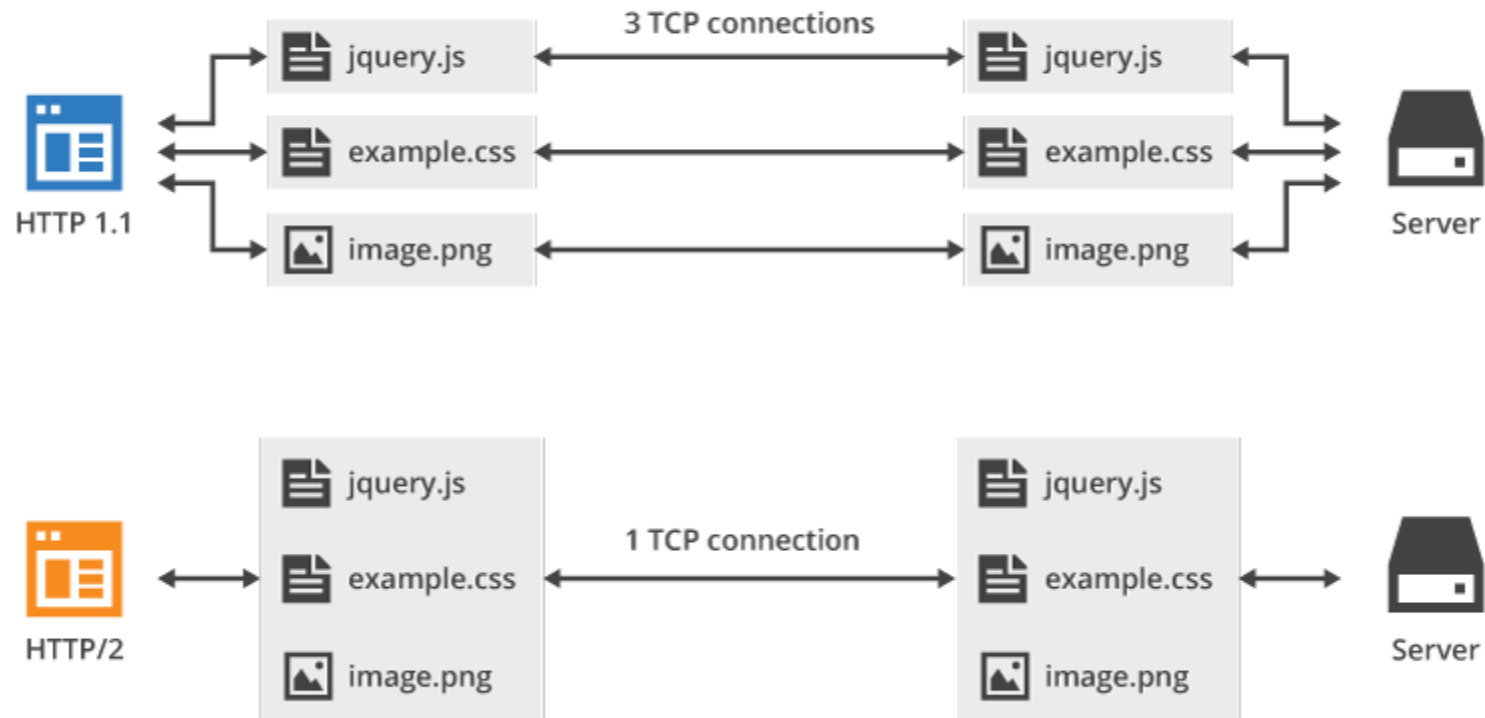
1990 – HTTP/0.9

1996 – HTTP/1.0

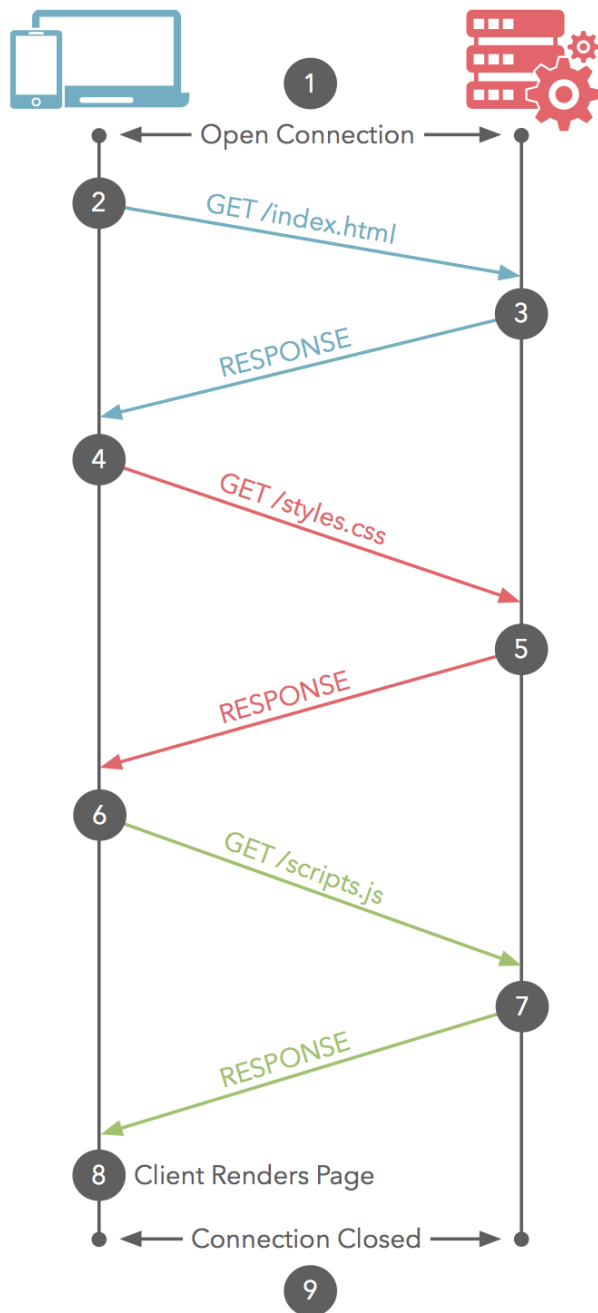
1999 – HTTP/1.1

2015 - HTTP/2

Multiplexing

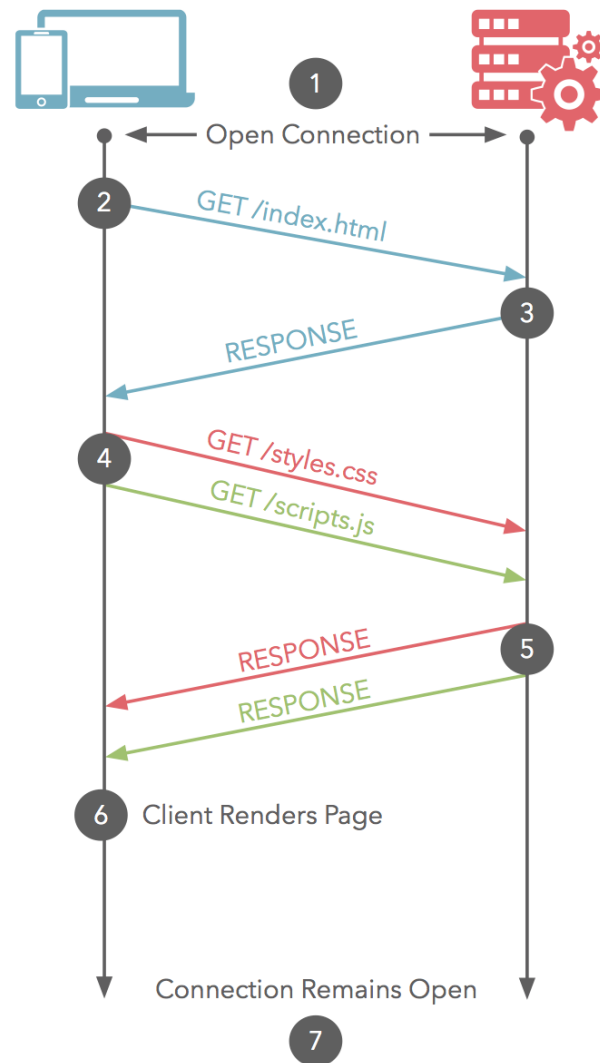


HTTP/1.1 Baseline

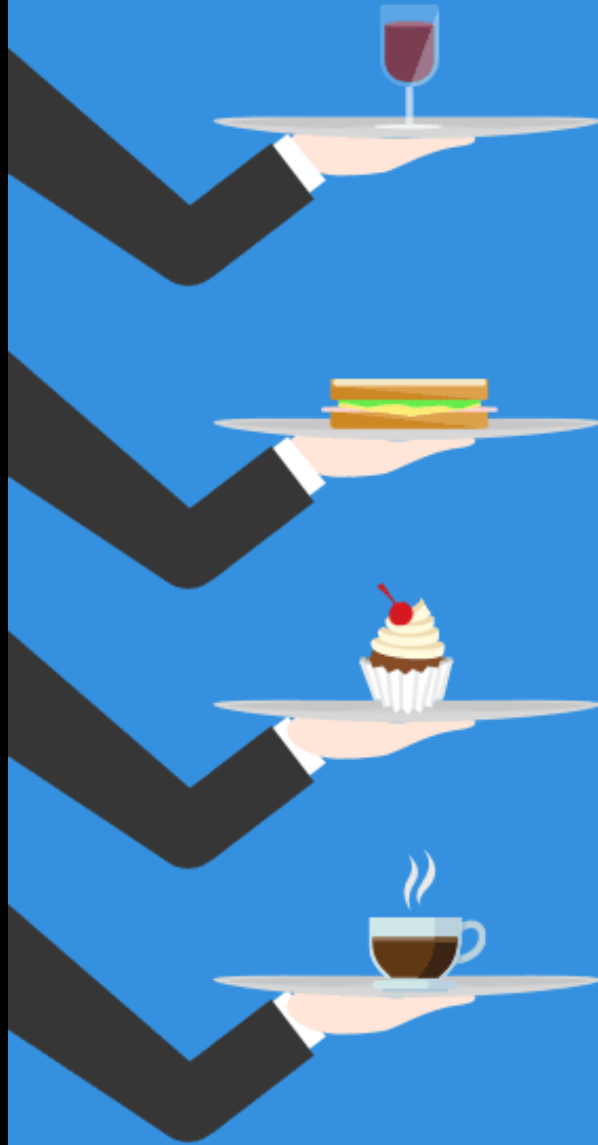


Time

HTTP/2 Multiplexing



HTTP/1.1



HTTP/2



HTTP/1 vs HTTP/2

URL tested: <http://domain.io>

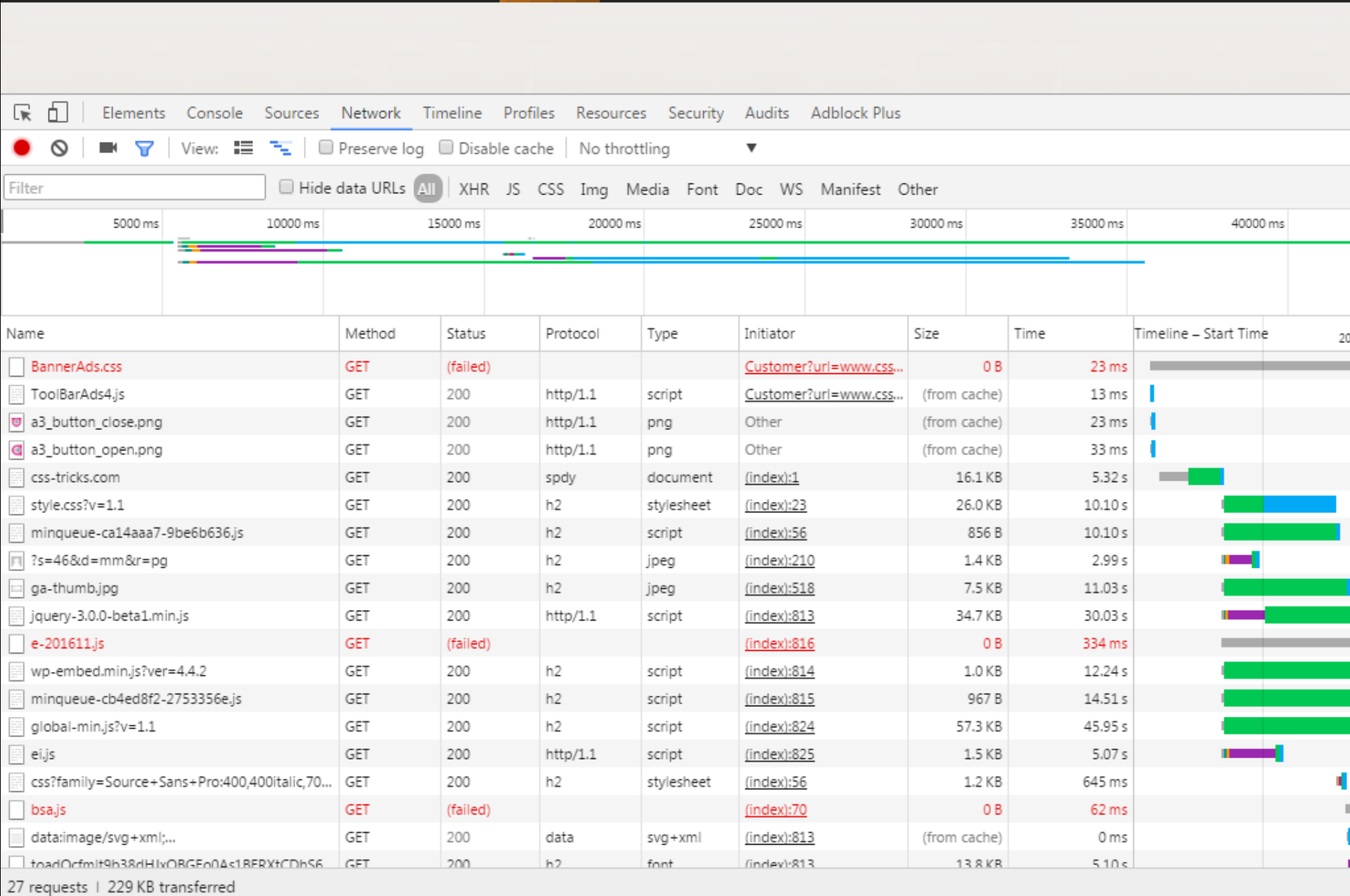
0.91 sec
Done

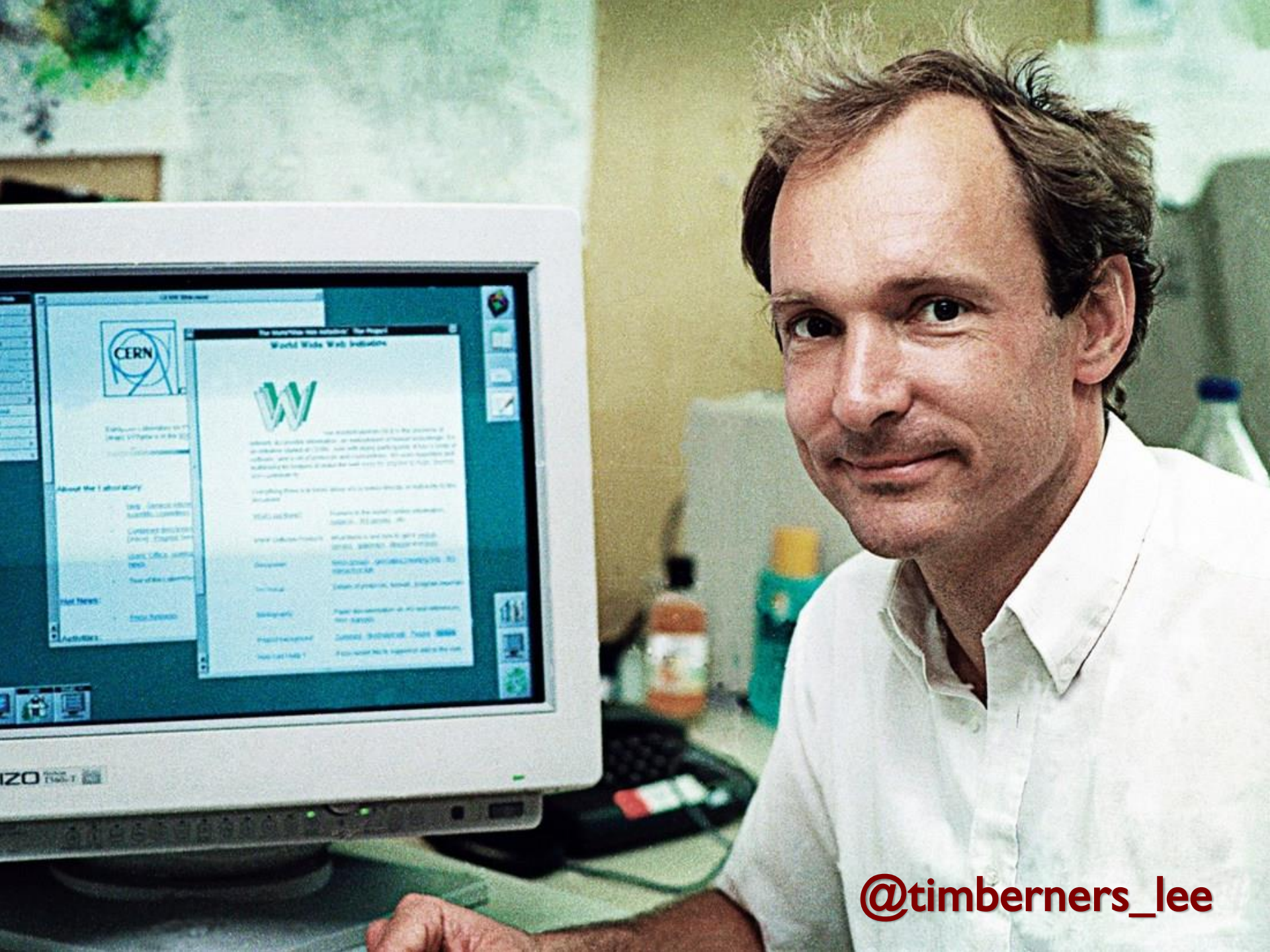


2.29 x
Faster*



0.40 sec
Done





@timberners_lee

HTTP

- HTTP requests and responses are in **plain text**
- Update: in HTTP/2, it's **binary**

HTTP/2

- Focuses on improved **performance**
- Based on **SPDY**, an experimental protocol developed at Google
- **Extending**, not replacing, the previous HTTP standard
- All communication is performed over a **single TCP connection**

HTTP Request

GET /tentang/profil-universitas HTTP/1.1

Host: ub.ac.id

Connection: keep-alive

Accept: text/html

User-Agent: Chrome/40.0.2214.115

Accept-Encoding: gzip, deflate

Accept-Language: id,en-US

Referer: http://ub.ac.id

HTTP Response

HTTP/1.1 200 OK

Date: Wed, 04 Mar 2015 04:44:52 GMT

Server: Apache/2.2.15 (CentOS)

X-Powered-By: PHP/5.3.3

Vary: Accept-Encoding, User-Agent

Content-Encoding: gzip

Content-Length: 6338

Connection: close

Content-Type: text/html; charset=UTF-8

HTTP responses
are followed
by the HTML
content

HTTP Response

The diagram shows an HTTP response structure overlaid on a background image of a server rack. The response is divided into two main sections: a 'Header' section at the top and a 'Body (HTML)' section below it. The 'Header' section is represented by a small rectangular area at the top of the response structure. The 'Body (HTML)' section is represented by a larger rectangular area below the header. The background image shows a server rack with various cables and components, with some green lights visible.

Header

Body
(HTML)

Header

```
HTTP/1.1 200 OK
Date: Wed, 04 Mar 2015 04:44:52 GMT
Server: Apache/2.2.15 (CentOS)
X-Powered-By: PHP/5.3.3
Vary: Accept-Encoding,User-Agent
Content-Encoding: gzip
Content-Length: 6338
Connection: close
Content-Type: text/html; charset=UTF-8
```

Body (HTML)

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

HTTP Status Code

- An HTTP response contains an **HTTP status code**

HTTP Status Code

status code

HTTP/1.1 **200 OK**

Date: Wed, 04 Mar 2015 04:44:52 GMT

Server: Apache/2.2.15 (CentOS)

X-Powered-By: PHP/5.3.3

Vary: Accept-Encoding, User-Agent

Content-Encoding: gzip

Content-Length: 6338

Connection: close

Content-Type: text/html; charset=UTF-8

HTTP Status Code

- **1xx** – Informational
- **2xx** – Success
- **3xx** – Redirection
- **4xx** – Client error
- **5xx** – Server error

Common HTTP Status Codes

- 200 OK: Request is fulfilled
- 304 Not Modified: Requested resource has not been modified
- 400 Bad Request: Request could be understood (syntax error)
- 403 Forbidden: Server refuses to supply the resource

HTTP Methods

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



A close-up, slightly blurred photograph of a server rack. The focus is on a row of network ports where several black cables are plugged in. Several green LED indicator lights are illuminated, suggesting active connections. The background shows other server units and more cables, creating a sense of a large data center environment.

PHP Introduction

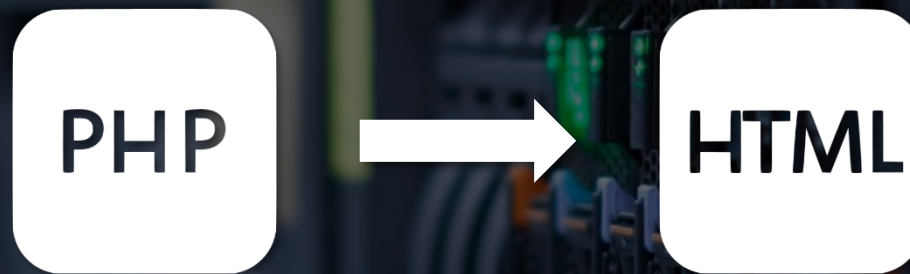
Created by
Rasmus Lerdorf in 1994



@rasmus

PHP Introduction

- A simple yet powerful language designed for creating **HTML content**



PHP Introduction

- Can be used in three primary ways:
 1. Server-side scripting
 2. Command-line scripting
 3. Client-side application
(using PHP-GTK, gtk.php.net)

PHP Introduction

- PHP runs on all major operating systems: from Unix variants including Linux, FreeBSD, Ubuntu, Debian, and Solaris to Windows and Mac OS X
- It can be used with many web servers: Apache, Microsoft IIS, and nginx

PHP Introduction

- PHP also has built-in support for generating PDF files, GIF, JPG, and PNG images



PHP Introduction

- Supports all major databases: MySQL, PostgreSQL, Oracle, Sybase, MS-SQL, and DB2

PHP Introduction

- PHP pages are generally HTML pages with PHP commands embedded in them



PHP

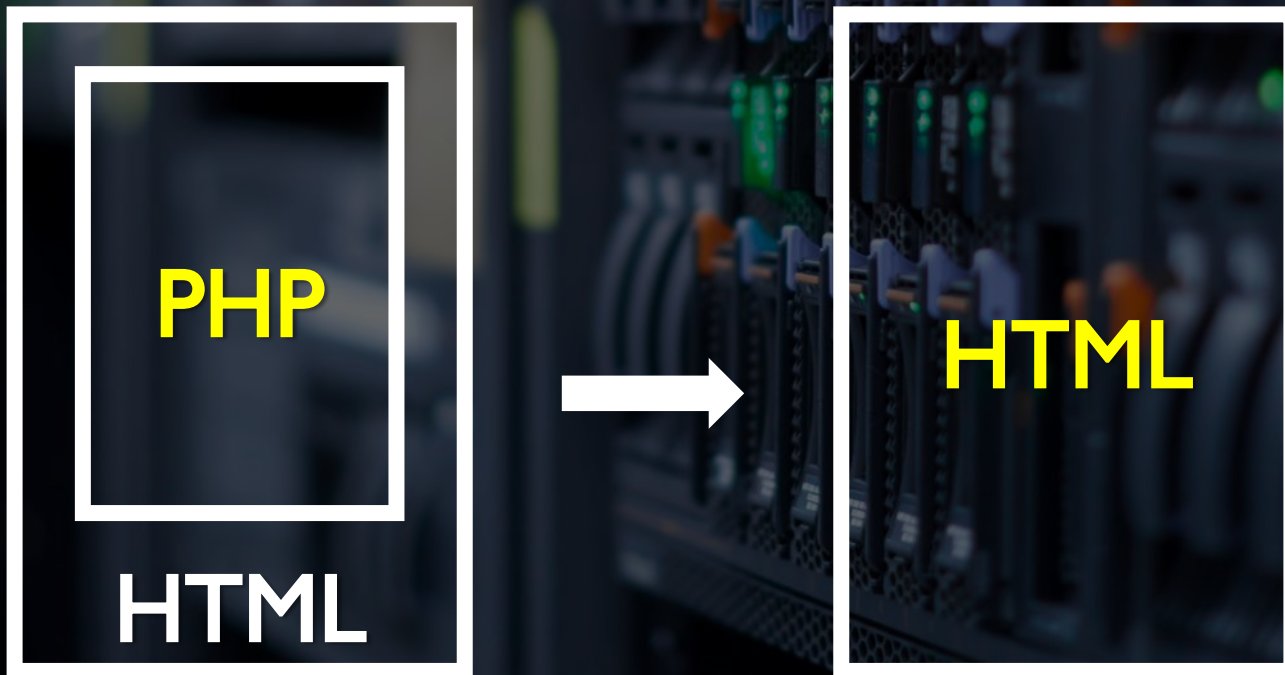
HTML

PHP Introduction

```
<html>
  <head>
    <title>Look Out World</title>
  </head>
  <body>
    <?php echo "Hello, world!"; ?>
  </body>
</html>
```

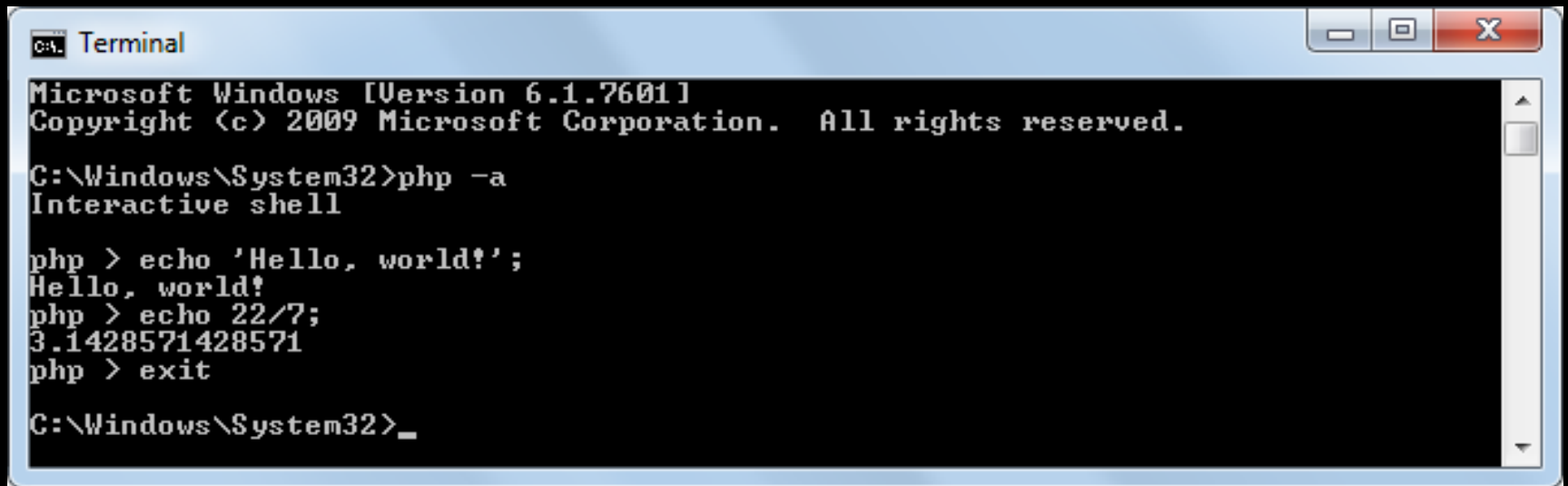

PHP Introduction

- The web server processes the PHP commands and **sends their output**



PHP Introduction

- We can use **PHP interactive shell** to execute PHP code quickly
- We can use **PHP REPL** services

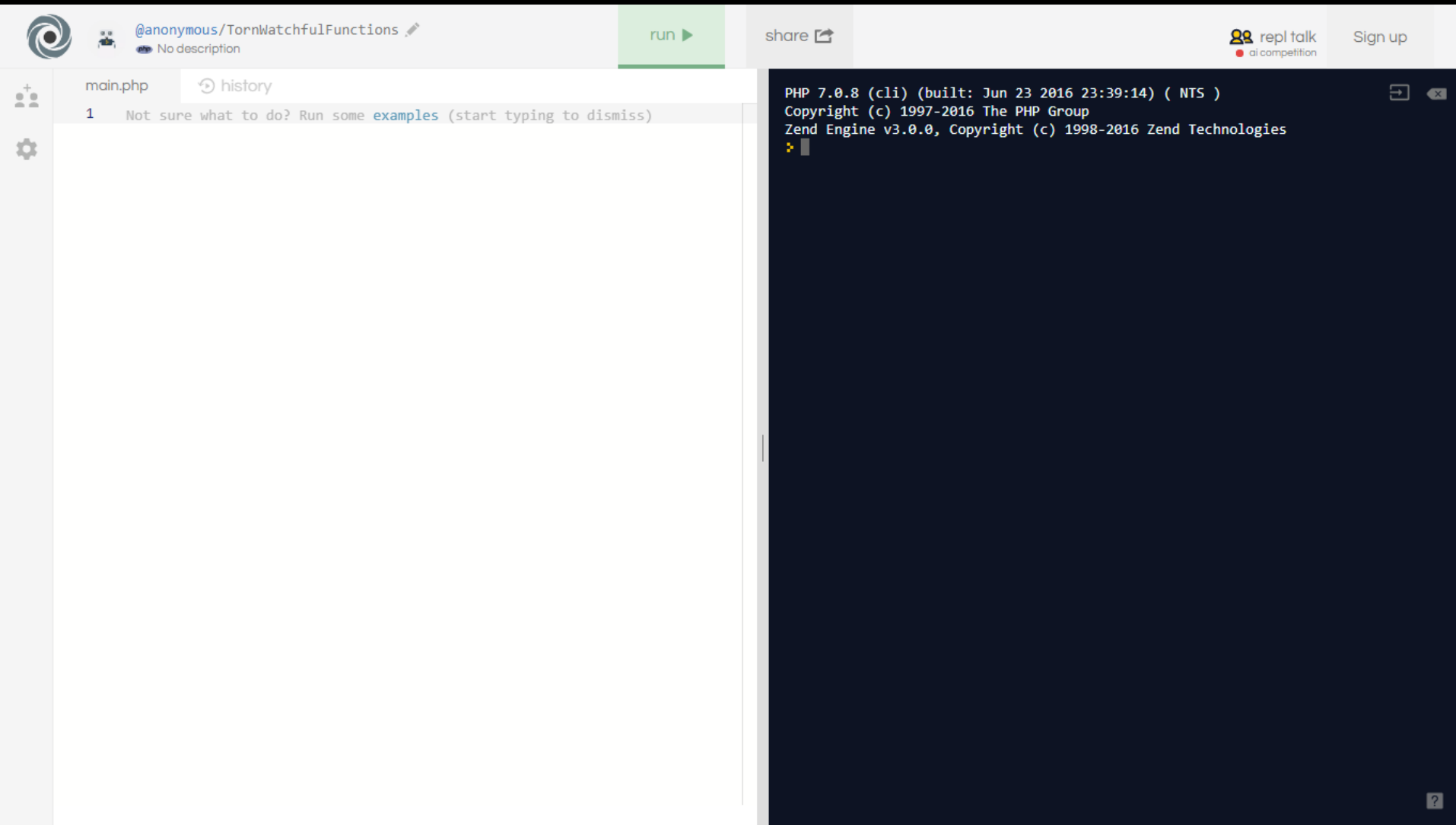


```
C:\Windows\System32>php -a
Interactive shell

php > echo 'Hello, world!';
Hello, world!
php > echo 22/7;
3.1428571428571
php > exit

C:\Windows\System32>_
```

PHP interactive shell (php -a)

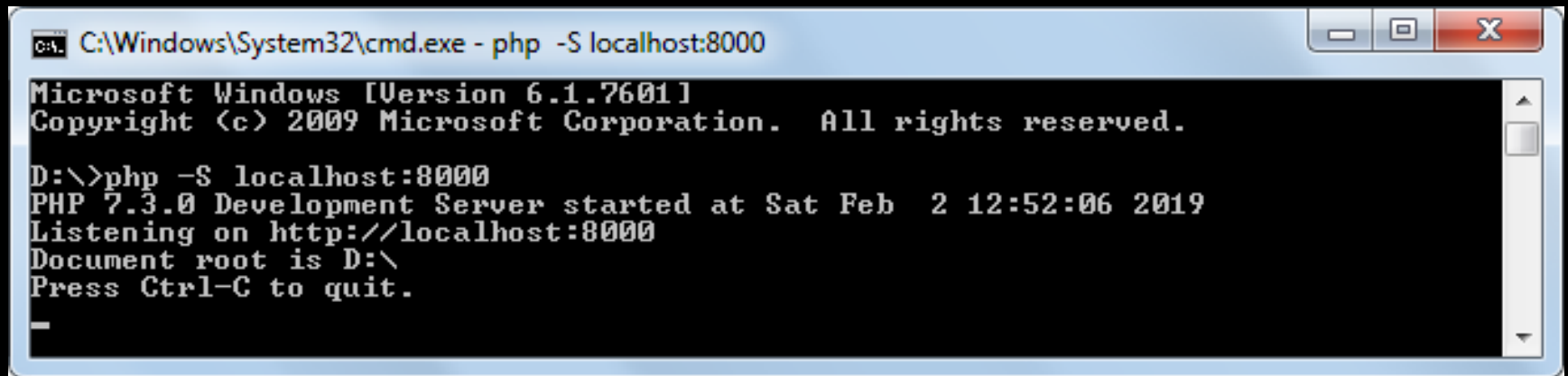


The screenshot displays the Repl.it web interface. At the top, the header includes the Repl.it logo, the username '@anonymous/TornWatchfulFunctions', a 'run' button, a 'share' button, and links for 'repl talk', 'ai competition', and 'Sign up'. The main workspace is divided into two panels. The left panel, titled 'main.php', contains a single line of text: '1 Not sure what to do? Run some examples (start typing to dismiss)'. The right panel is a dark-themed terminal window showing the output of a PHP CLI environment: 'PHP 7.0.8 (cli) (built: Jun 23 2016 23:39:14) (NTS)', 'Copyright (c) 1997-2016 The PHP Group', and 'Zend Engine v3.0.0, Copyright (c) 1998-2016 Zend Technologies'. A cursor is visible at the end of the last line in the terminal.

<https://repl.it/languages/php>

PHP Introduction

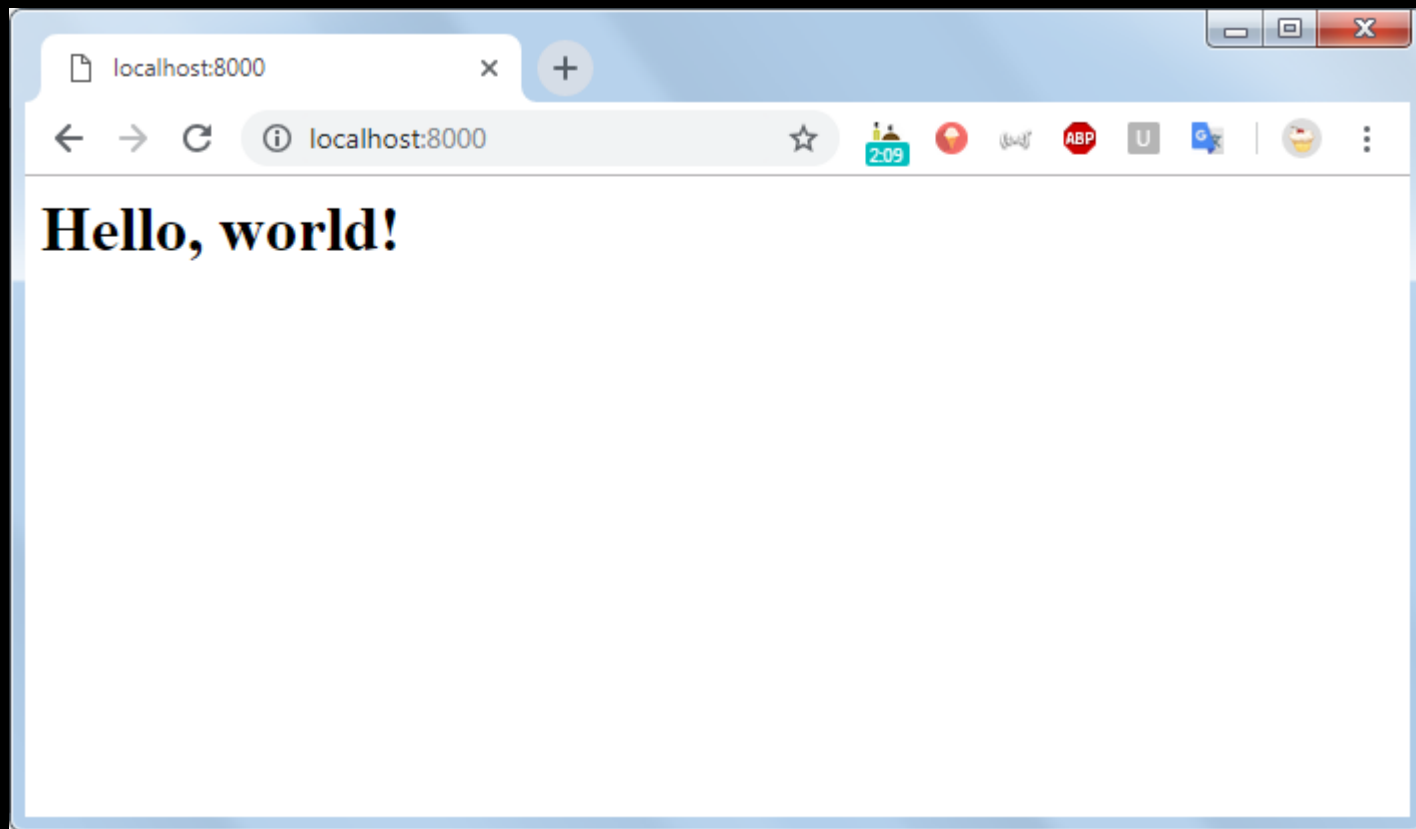
- Since version 5.4.0, PHP provides a **built-in web server**
- Warning: it is designed for development purposes only, **not for production**
- The server looks for **index.php** or **index.html** file in the current working directory



A screenshot of a Windows command prompt window. The title bar reads "C:\Windows\System32\cmd.exe - php -S localhost:8000". The window contains the following text:

```
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
D:\>php -S localhost:8000  
PHP 7.3.0 Development Server started at Sat Feb 2 12:52:06 2019  
Listening on http://localhost:8000  
Document root is D:\  
Press Ctrl-C to quit.  
_
```

php -S localhost:8000





Server- and Client- side Scripting

Server- and Client-side Scripting

- Make sure a **web server** is installed on your laptop

Your server-side scripts must be:

- stored in the **htdocs** (or WWW folder in nginx)
- written in a .php file
- written between the `<?php` and `?>` tags

Server- and Client-side Scripting

(htdocs/helloworld/hello.php)

```
<!DOCTYPE html>
```

```
<html><body>
```

```
  <h1><?php
```

```
    $hello = "Hello ";
```

```
    $world = "world!";
```

```
    echo $hello, $world;
```

```
  ?></h1>
```

```
</body></html>
```

Server- and Client-side Scripting

- Now open
“localhost/helloworld/hello.php” on your
browser



Server- and Client-side Scripting

- Rename “hello.php” to “index.php” then open “localhost/helloworld” on your browser
- “index.php” is the default file in the web server so you don’t have to specify the file name in the URL

Server- and Client-side Scripting

- Server-side scripts are run on server, while client-side ones are run on client (browser)



Server- and Client-side Scripting

(htdocs/helloworld/server_client.php)

```
<!DOCTYPE html>
<html><body>
    <?php
        $var1 = 10;
        $var2 = 20;
        echo "Server output: ", $var1 * $var2;
    ?>
    <div id="client"></div>
    <script>
        var var1 = 10;
        var var2 = 30;
        var hasil = "Client output: " + var1 * var2;
        document.querySelector("#client").innerHTML =
hasil;
    </script>
</body></html>
```

Server- and Client-side Scripting

- Open
“localhost/helloworld/server_client.php”
on your browser

Server- and Client-side Scripting

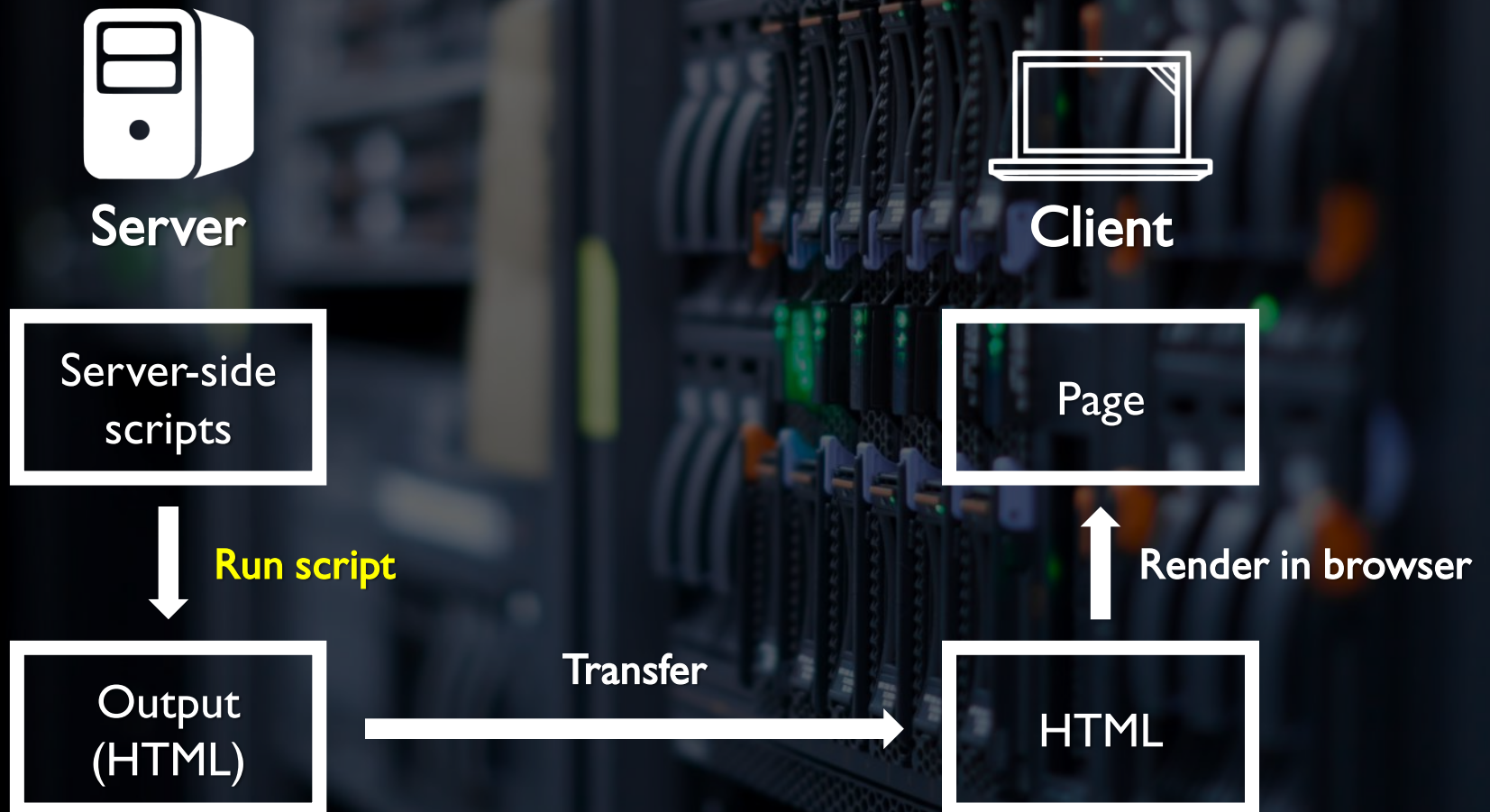
- Now view the page source by pressing Ctrl+U on your browser



Page Source

```
<!DOCTYPE html>
<html><body>
  Server output: 200
  <div id="client"></div>
  <script>
    var var1 = 10;
    var var2 = 30;
    var hasil = "Client output: " + var1 * var2;
    document.querySelector("#client").innerHTML =
hasil;
  </script>
</body></html>
```

Server-side Scripts



Client-side Scripts

