HTTP Protocol

Handling Requests, Constructing Responses, HTTP/2 & HTTP/3

HTTP

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Questions?

Software University

sli.do

#csharp-web

Table of Content

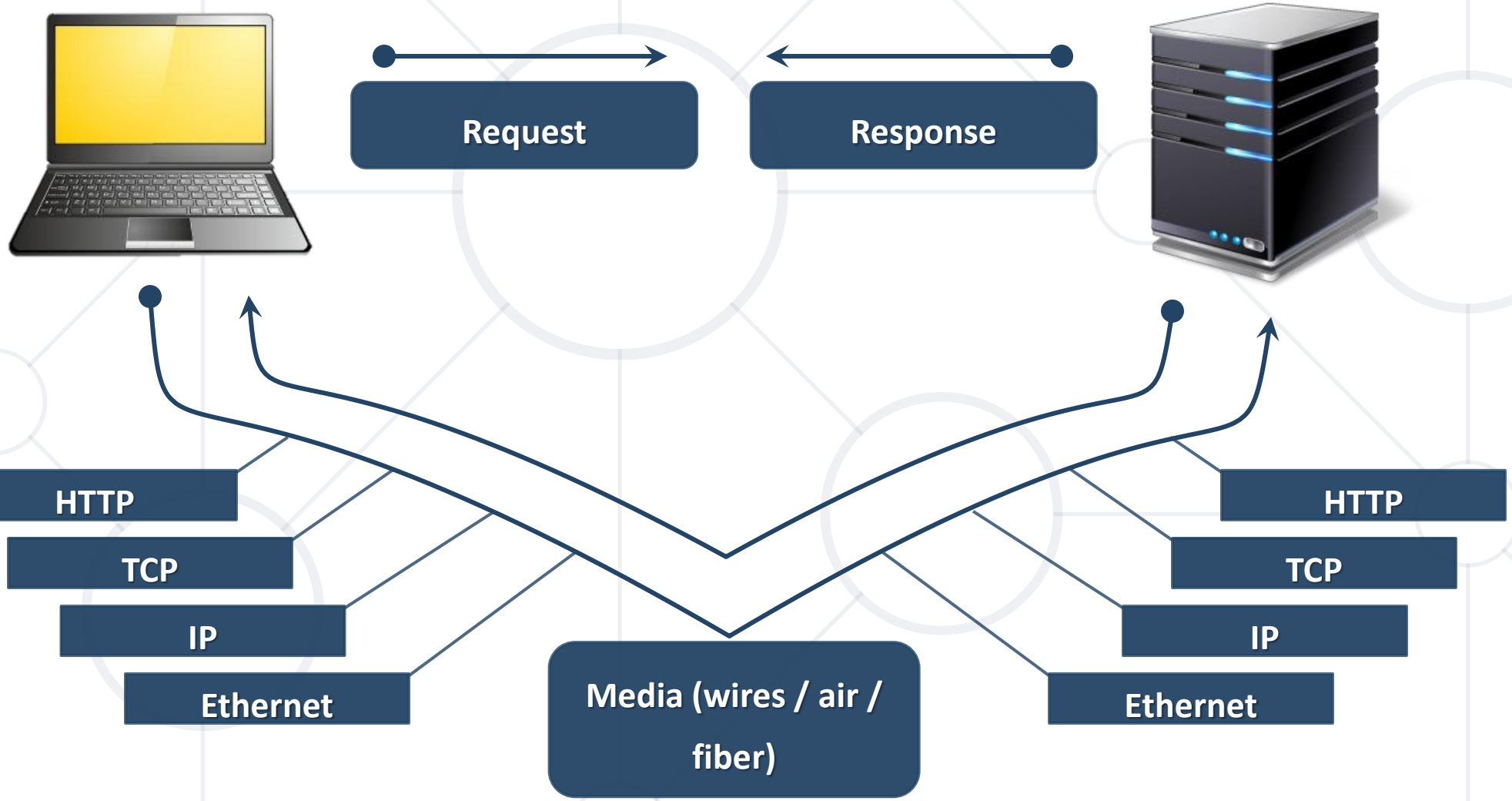
HTTP Basics

Basic Concepts

What is a Protocol?

- A **communication** protocol == **set of rules**, which define how two or more parties are talking to each other
- It is like a common language used for communication between machines

Hyper Text Transfer Protocol

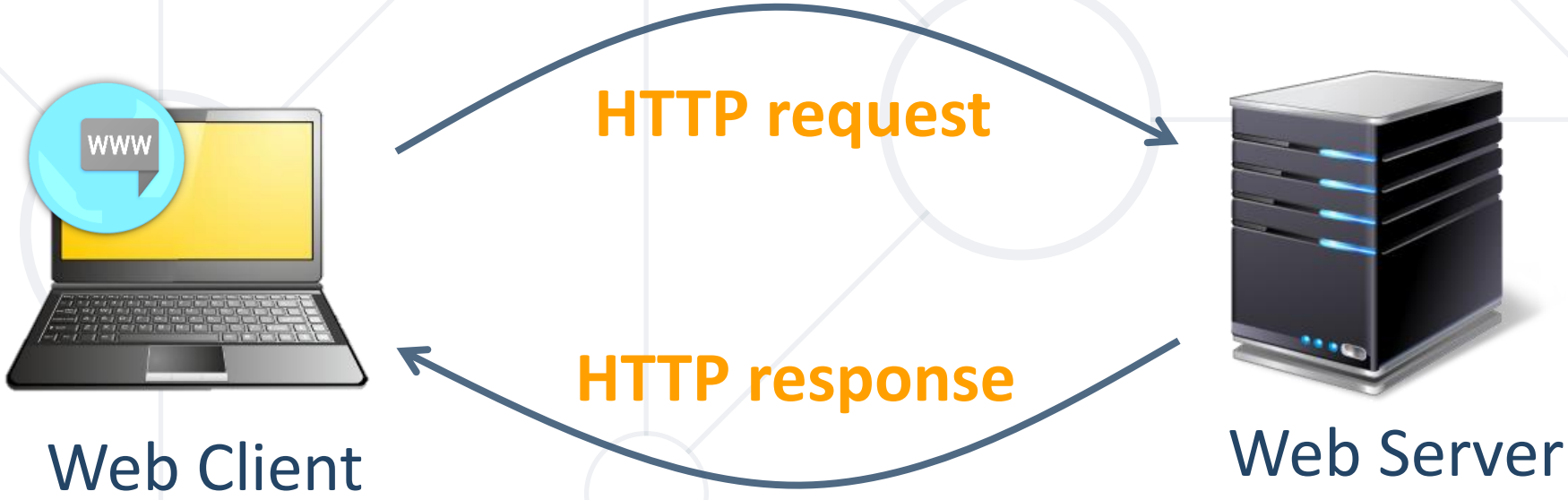


HTTP Protocol

- HyperText Transfer Protocol
- Text-based client-server protocol
- General-purpose client-server protocol used to transmit data across the web
  - For transferring Web resources (HTML files, images, styles, etc.)
- Relies on URLs

HTTP Protocol

- Stateless
  - Each HTTP request is **independent** from the others
  - Cookies and Web storages can overcome this
- Uses the **request-response model**



HTTP Conversation: Example

- HTTP request

```
GET /courses/javascript HTTP/1.1
Host: www.softuni.bg
User-Agent: Mozilla/5.0
<CRLF>
```

The empty line denotes the end of the request header

- HTTP response

```
HTTP/1.1 200 OK
Date: Mon, 5 Jul 2020 13:09:03 GMT
Server: Microsoft-HTTPAPI/2.0
Last-Modified: Mon, 12 Jul 2014 15:33:23 GMT
Content-Length: 54
<CRLF>
<html><title>Hello</title>
Welcome to our site</html>
```

The empty line denotes the end of the response header

URL

Uniform Resource Locator

Uniform Resource Locator (URL)

- URL == formatted string, consisting of
  - **Network protocol** (`http`, `ftp`, `https`...) – HTTP in most cases
  - **Host** or **IP** address (`softuni.org`, `gmail.com`, `127.0.0.1`, `web`)
  - **Port** (the default port is `80`) – integer in the range [`0…65535`]
  - **Path** (`/forum`, `/path/index.php`)
  - **Query string** (`?id=27&lang=en`)
  - **Fragment** (`#slides`) – navigate to some section in the page

URL Encoding

- URLs are encoded according RFC 1738
- Safe URL characters: `[0-9a-zA-Z]`, `$`, `-`, `_`, `.`, `+`, `*`, `'`, `(`, `)`, `,`, `!`
- All other characters are escaped by

```
%[character hex code]
```

  - Space is encoded as "`+`" or "`%20`"

```
Наков-爱-SoftUni
```

  - URL-encoded string

```
%D0%9D%D0%B0%D0%BA%D0%BE%D0%B2-%E7%88%B1-SoftUni
```

Char	URL Encoding
space	%20
щ	%D1%89
"	%22
#	%23
$	%24
%	%25
&	%26

Valid and Invalid URLs – Examples

- Some valid URLs

```
http://www.google.bg/search?sourceid=navclient&ie=UTF-
8&rlz=1T4GGLL_enBG369BG369&q=http+get+vs+post
```

```
http://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%
D1%80%D0%BD%D0%B0_%D0%B0%D0%BA%D0%B0%D0%B4%D0%B5%D0%BC%D0%B8%D1%8F
```

- Some invalid URLs

```
http://www.google.bg/search?&q=C# .NET 6.0
```

Should be ?q=C%23+.NET+6.0

```
http://www.google.bg/search?&q=бира
```

Should be ?q=%D0%B1%D0%B8%D1%80%D0%B0

HTTP Request

What is a HTTP Request?

HTTP Request Message

- Request message sent by a client consists of
  - HTTP **request line**
    - Request method (**GET** / **POST** / **PUT** / **DELETE** / …)
    - Resource URI (**URL**)
    - Protocol **version**
  - HTTP **request headers**
    - Additional parameters
  - HTTP **request body** – optional data, e.g., posted form fields

```
<method> <resource> HTTP/<version>
<headers>
(empty line)
<body>
```

HTTP Request Methods

- **HTTP** defines **methods** to indicate the desired action to be performed on the identified resource

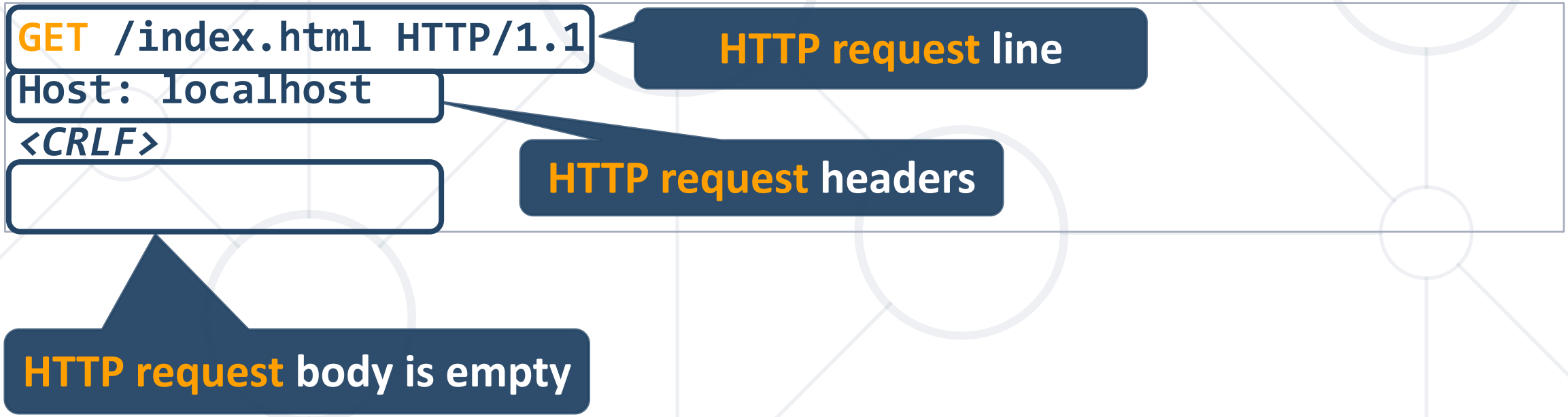
Method	Description
GET	Retrieve a resource
POST	Create / store a resource
PUT	Update (replace) a resource
DELETE	Delete (remove) a resource
PATCH	Update resource partially (modify)
HEAD	Retrieve the resource's headers

CRUD == the four main functions of persistent storage

Other Methods
CONNECT
OPTIONS
TRACE

HTTP GET Request

- **GET** is used to request data from a specified resource
- Example of HTTP **GET** request



HTTP POST Request

- The **POST** method transfers data in the HTTP **body**
- Example of HTTP **POST** request

```
POST /login.html HTTP/1.1
Host: localhost
Content-Length: 59
Content-Type: application/x-www-form-urlencoded
<CRLF>
username=testUser&password=topSecret
<CRLF>
```

HTTP request line

HTTP request headers

HTTP request body holds the submitted form data

HTTP Response

What is a HTTP Response?

HTTP Response Message

- The **response message** sent by the HTTP server consists of
  - HTTP response **status line**
    - Protocol **version**
    - Status **code**
    - Status **phrase**
  - Response **headers**
    - Provide meta data about the returned resource
  - Response **body**
    - The content of the HTTP response (data)

```
HTTP/<version> <status code> <status text>
<headers>
(empty line)
<response body - the requested resource>
```

HTTP Response – Example

```
HTTP/1.1 200 OK
Date: Fri, 17 Jul 2020 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Accept-Ranges: bytes
Content-Length: 84
Content-Type: text/html
<CRLF>
<html>
  <head><title>Test</title></head>
  <body>Test HTML page.</body>
</html>
```

HTTP response status line

HTTP response headers

HTTP response body

HTTP Response Status Codes

- HTTP response **code** classes
  - **1xx**: informational (e.g., "**100** Continue")
  - **2xx**: successful (e.g., "**200** OK", "**201** Created")
  - **3xx**: redirection (e.g., "**304** Not Modified", "**301** Moved Permanently", "**302** Found")
  - **4xx**: client error (e.g., "**400** Bad Request", "**404** Not Found", "**401** Unauthorized", "**409** Conflict")
  - **5xx**: server error (e.g., "**500** Internal Server Error", "**503** Service Unavailable")

HTTP Error Response – Example

```
HTTP/1.1 404 Not Found
Date: Fri, 17 Nov 2020 16:09:18 GMT+2
Server: Apache/2.2.14 (Linux)
Connection: close
Content-Type: text/html
<CRLF>
<html><head><title>404 Not Found</title></head>
<body>
<h1>Not Found</h1>
<p>The requested URL /img/logo.gif was not found on this server.</p>
<hr><address>Apache/2.2.14 Server at Port 80</address>
</body></html>
```

HTTP response status line

HTTP response headers

HTTP response body

Browser Redirection

- HTTP **GET** requesting a moved URL

```
GET / HTTP/1.1
Host: http://softuni.org
User-Agent: Gecko/20100115 Firefox/3.6
<CRLF>
```

- The following HTTP response (**301** Moved Permanently) tells the browser to request another URL

```
HTTP/1.1 301 Moved Permanently
Location: http://softuni.bg
…
```

Content-Type and Disposition

- The `Content-Type` response header specifies how the output should be processed
- Examples

UTF-8 encoded HTML page.
Will be shown in the browser.

```
Content-Type: text/html; charset=utf-8
```

```
Content-Type: application/pdf
Content-Disposition: attachment; filename="Report-April-2020.pdf"
```

This will download a PDF file named
Report-April-2020.pdf

MIME and Media Types

Multi-Purpose Internet Mail Extensions

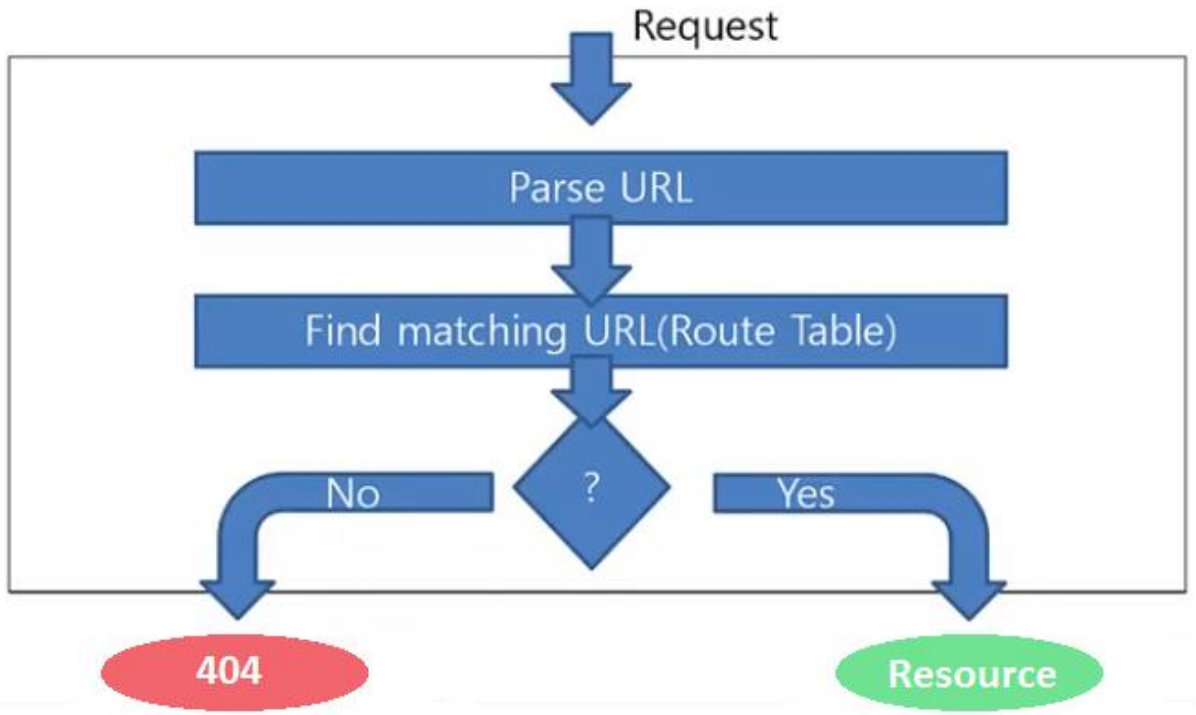
What is MIME?

- **MIME** == **M**ulti-Purpose **I**nternet **M**ail **E**xtensions
  - Internet standard for encoding resources
  - Originally developed for email attachments
  - Used in many Internet protocols like HTTP and SMTP
- MIME defines several concepts
  - `Content-Type`, e.g. `text/html`, `image/gif`, `application/pdf`
    - Content `charset`, e.g. `utf-8`, `ascii`, `windows-1251`
  - `Content-Disposition`, e.g. `attachment; filename=logo.jpg`
  - Multipart messages (multiple resources in a single document)

Common MIME Media Types

Software University

MIME Type / Subtype	Description
application/json	JSON data
image/png	PNG image
image/gif	GIF image
text/html	HTML
text/plain	Text
text/xml	XML
video/mp4	MP4 video
application/pdf	PDF document



Routing

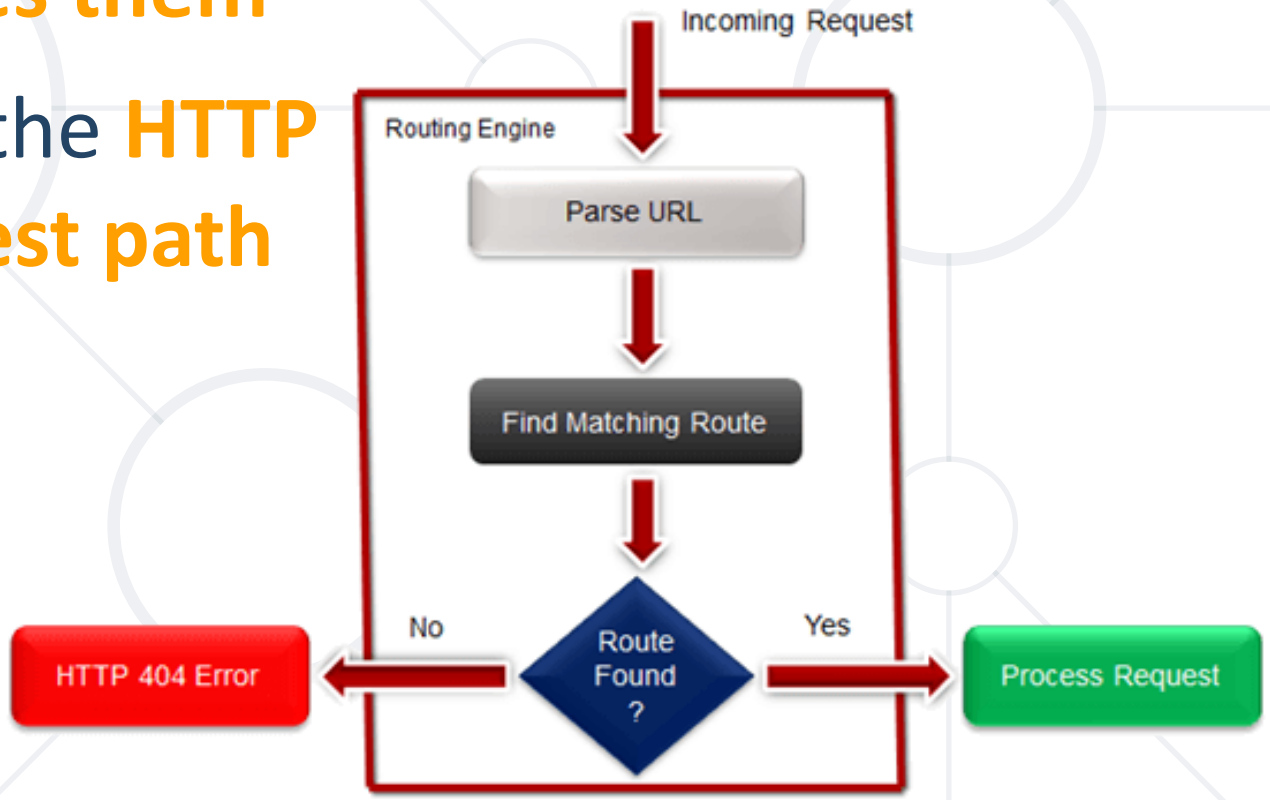
Mapping HTTP Request to HTTP Responses

What is Web Routing?

- Web Routing is a mechanism where **HTTP requests** are **routed** to the **code that handles them**
- Requests are routed based on the **HTTP request method** and the **request path**

```
GET / HTTP/1.1
```

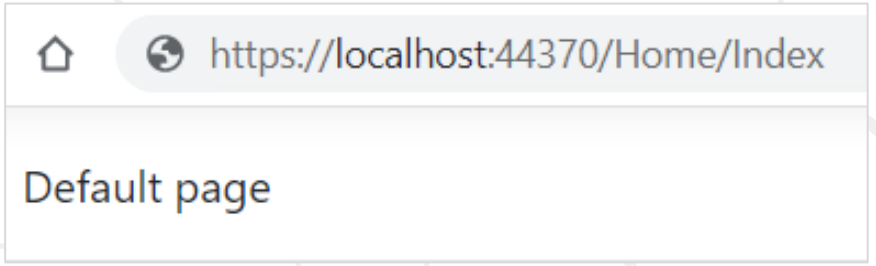
- **HTTP requests** are mapped to **HTTP responses**
- Example: route "**/**" is often mapped to the app's **Home page**

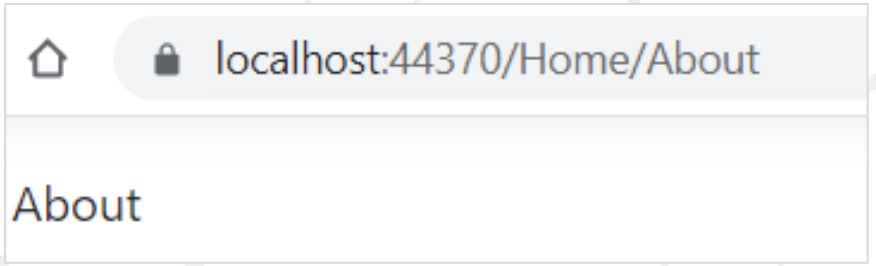


Mapping Physical Files

RouteTable.Routes.MapPageRoute		
Route Name	Route URL	Physical File Name
Home	Home/Index	Index.html
About	Home/About	About.html

User-friendly URL replacement of the physical file name





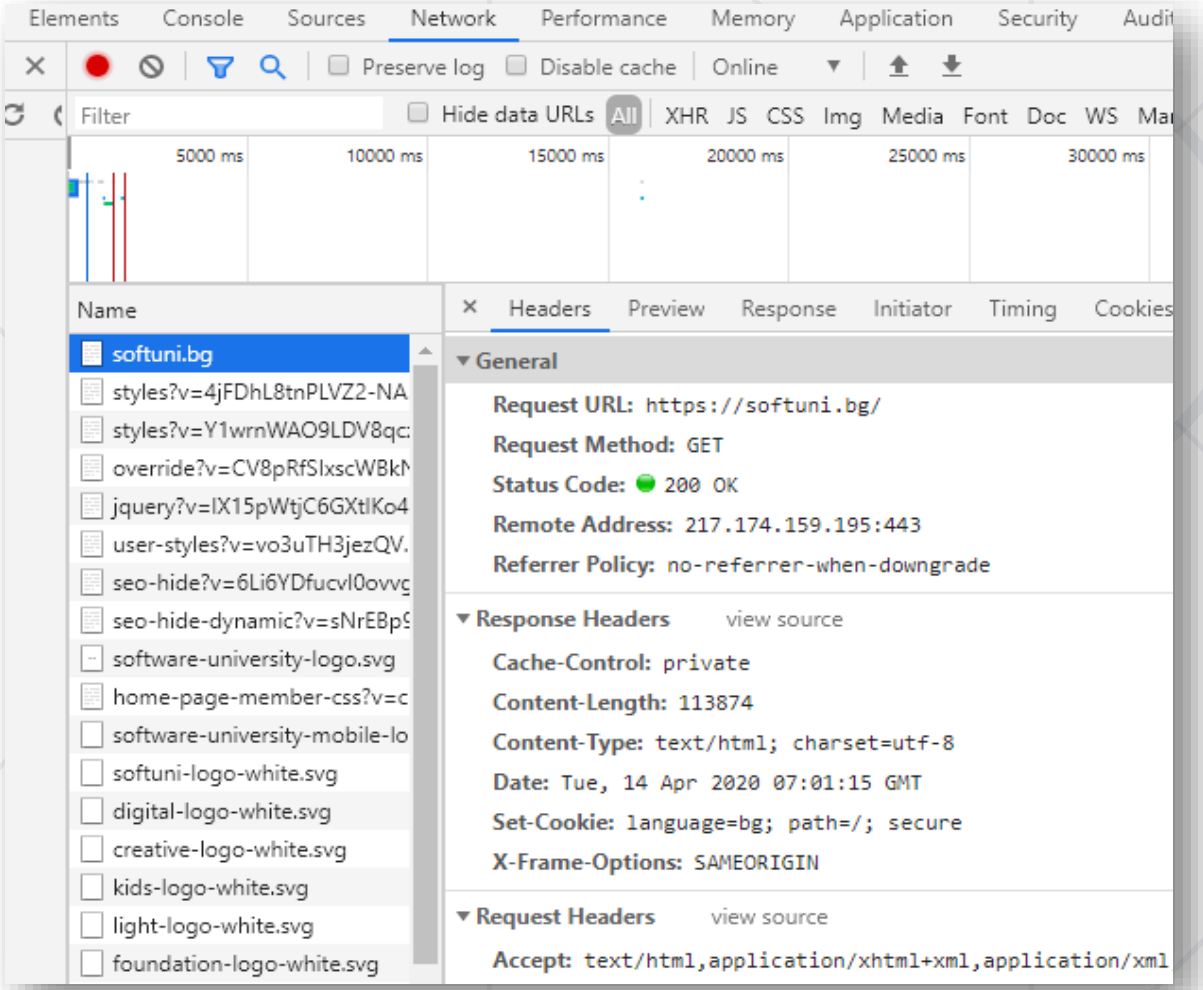

```
void RegisterRoutes(RouteCollection routes)
{
  routes.MapPageRoute("Home","Home/Index","/wwwroot/Home/Index.html");
  routes.MapPageRoute("About","Home/About","/wwwroot/Home/About.html");
}
```

HTTP Tools

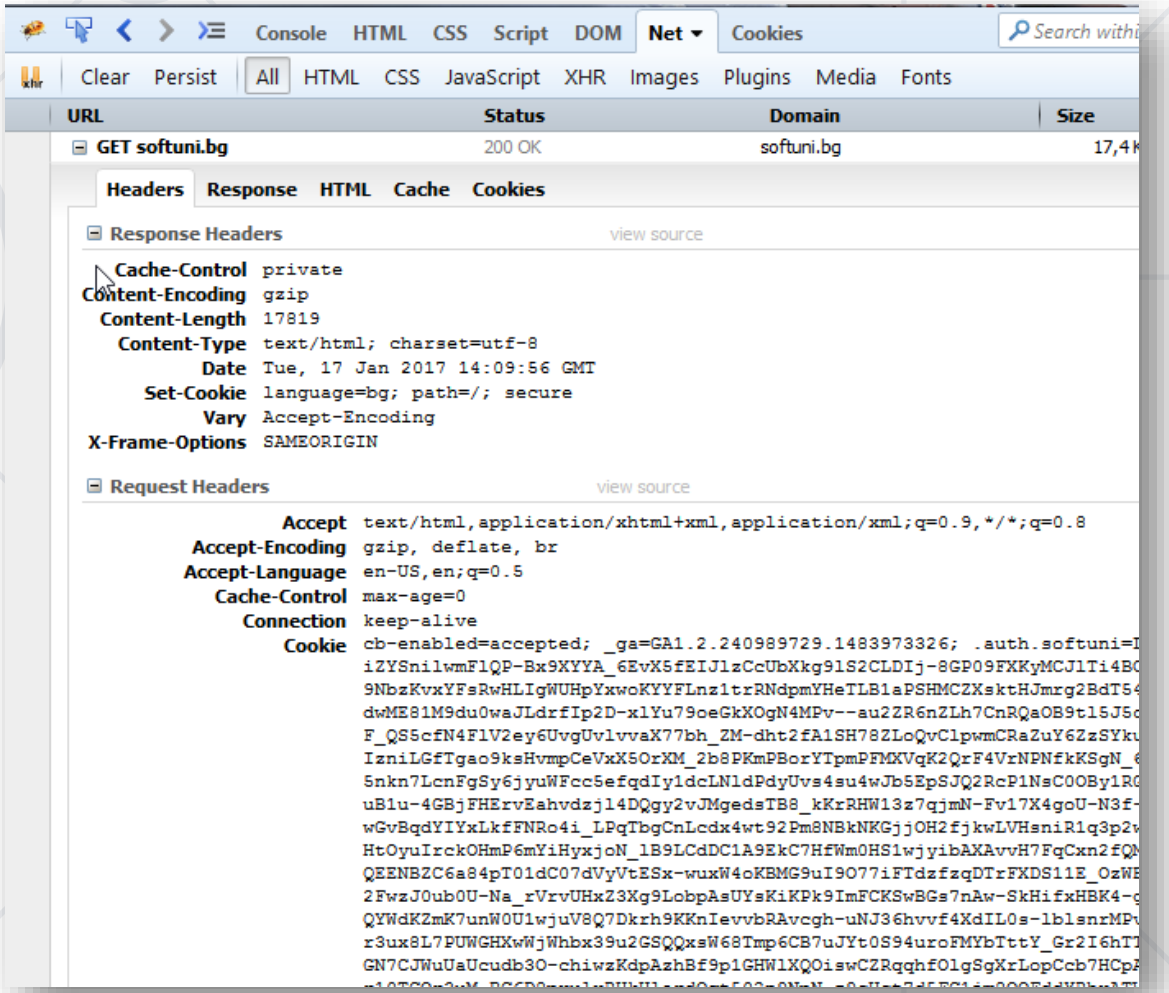
Tools for Developers

HTTP Tools for Developers – Browser

Software University



Chrome Developer Tools



Firebug

HTTP Tools for Developers

Insomnia Rest

Postman

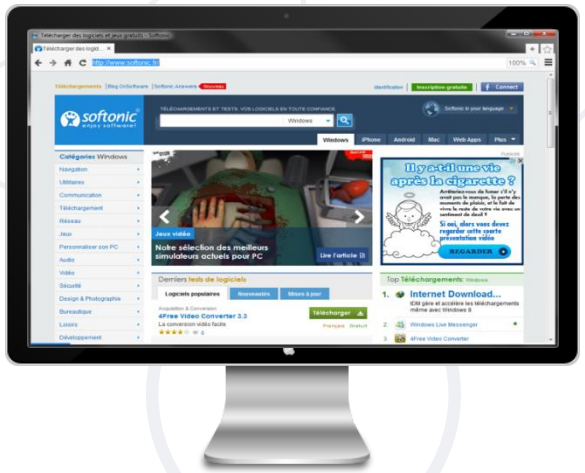
RESTClient

Web Server

What is a Web Server?

- Computer system that processes requests via **HTTP**, the basic network protocol

Web Client



Communication

Web Server

Web Server Work Model

Protocols:

HTTP

FTP

WebSocket


```
http://localhost:8000/softuni.jpg
```

Web Server Work Model

Software University

Web Client

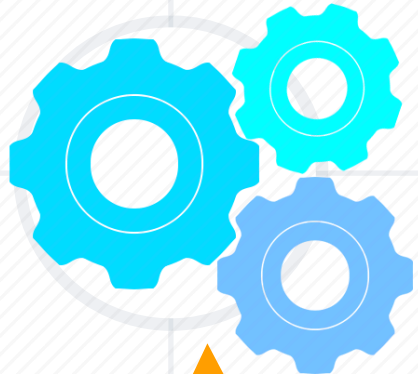
Request

Protocol

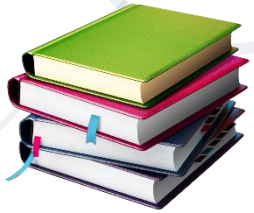
Response

Web Server

Technology



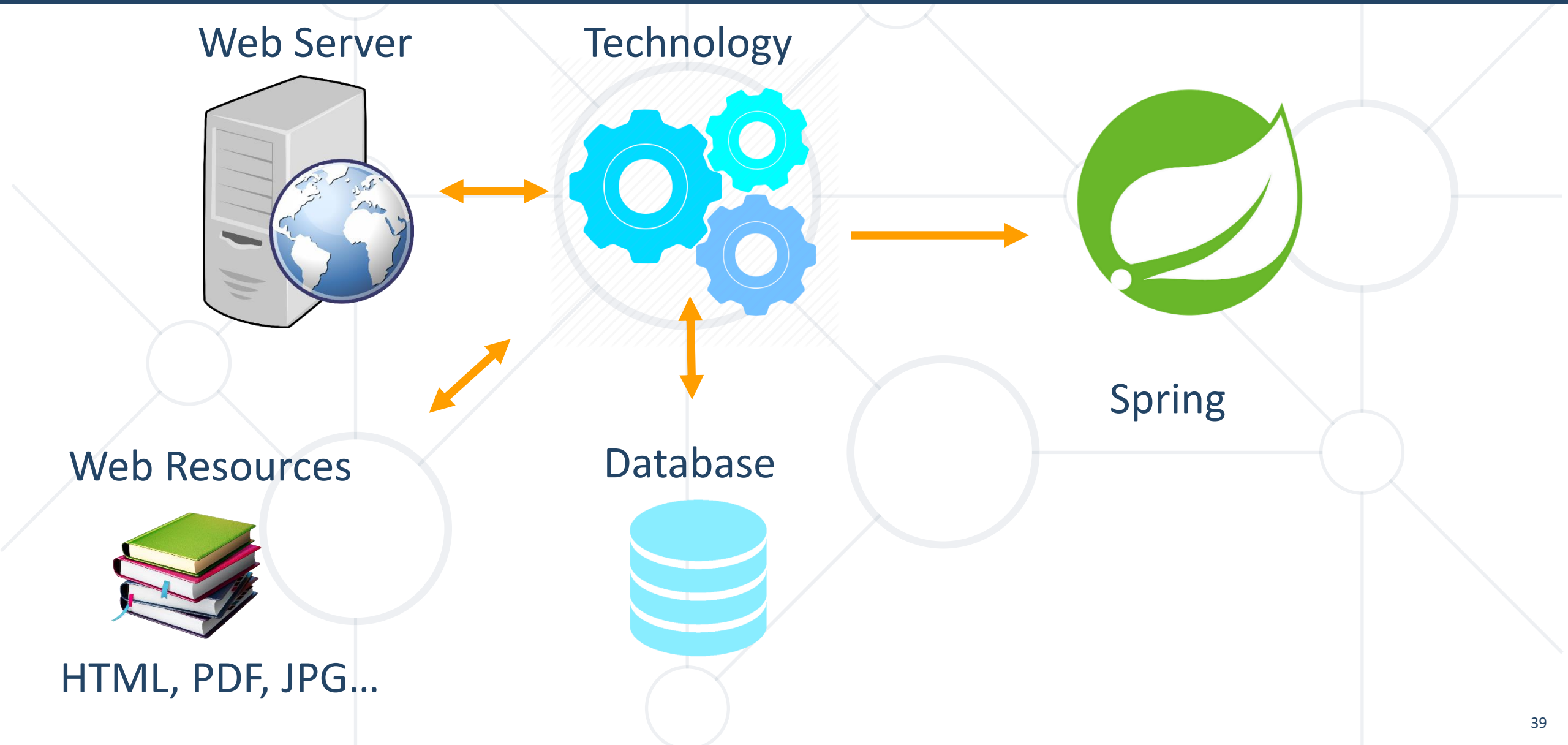
Web Resources



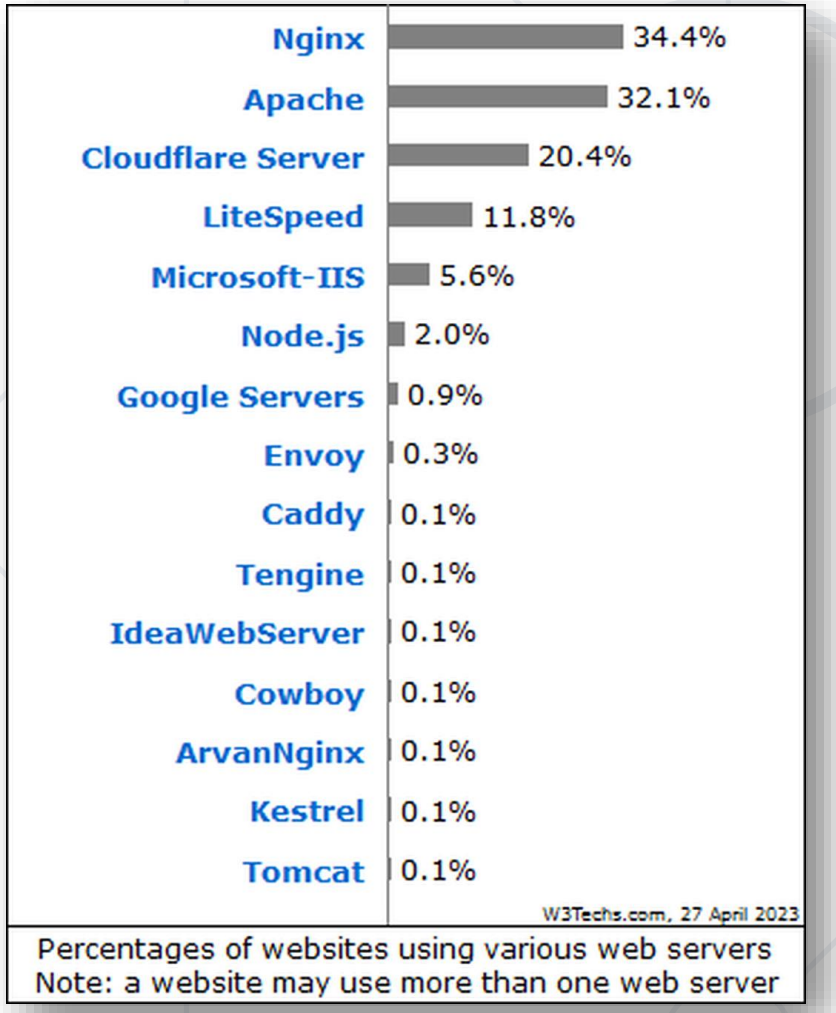
HTML, PDF, JPG…

Database

Web Server Work Model



Most Popular Web Servers (W3Techs)



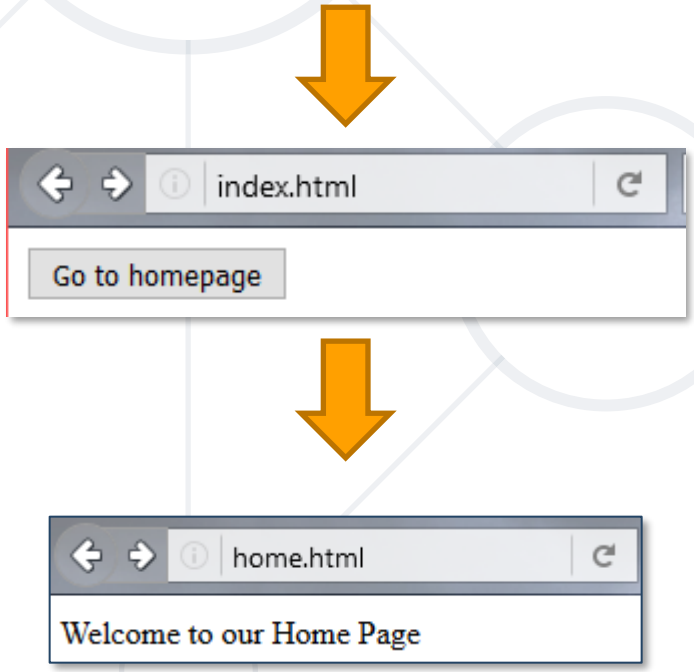
HTML Forms

Form Method and Action

HTML Forms – Action Attribute

- Defines where to submit the form data

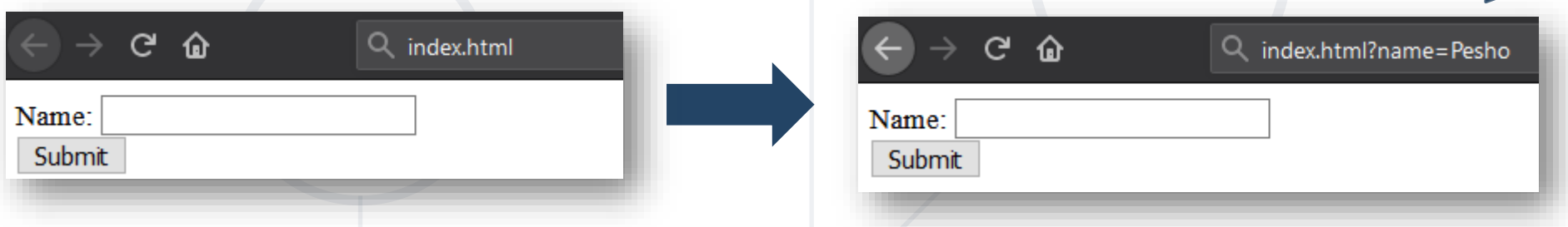
```
<form action="home.html">
  <input type="submit" value="Go to homepage"/>
</form>
```



HTML Forms – Method Attribute

- Specifies the HTTP method to use when sending form data

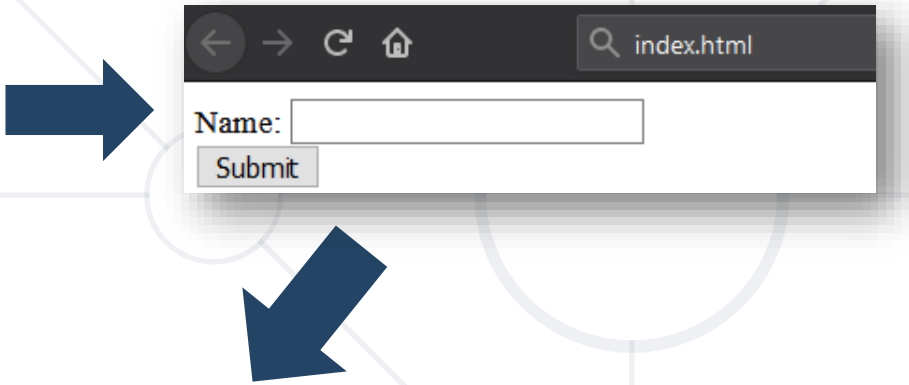
```
<form action="/" method="get">
  Name: <input type="text" name="name">
  <br>
  <input type="submit" value="Submit">
</form>
```



The form data is in the URL

HTML Forms – Method Attribute

```
<form action="/" method="post">
  Name: <input type="text" name="name">
  <br>
  <input type="submit" value="Submit">
</form>
```



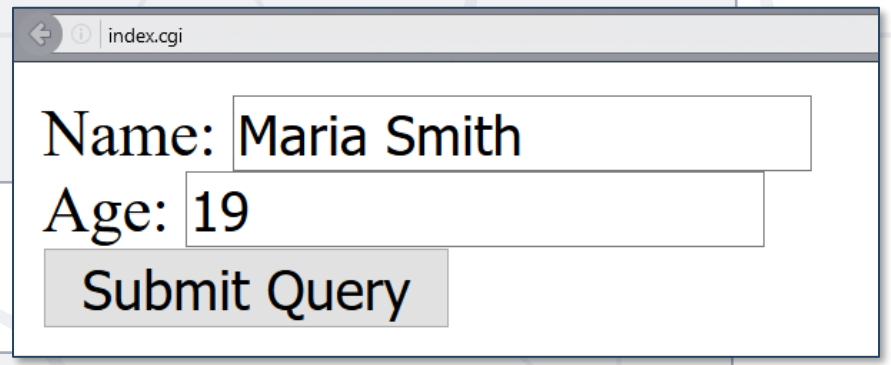

```
POST http://localhost/index.html HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

name=Pesho
```

HTTP request body holds the form data

URL Encoded Form Data – Example

```
<form action="/" method="post">
  Name: <input type="text" name="name"/> <br/>
  Age: <input type="text" name="age"/> <br/>
  <input type="submit" />
</form>
```




```
POST http://localhost/cgi-bin/index.cgi HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 23

name=Maria+Smith&age=19
```

File uploads are not supported

GET vs. POST Method

- **GET**
  - Values are contained in the **URL**
  - Has a length limitation of **255 characters**
  - It is **often** cacheable
  - Supports only **string data types**
  - Parameters **are saved** in browser history
  - Results **can** be bookmarked
- **POST**
  - Values are contained in the **message's body**
  - Does **not have** a length limitation
  - It is **hardly** cacheable
  - Supports **different data types**
  - Parameters **are not saved** in browser history
  - Results **cannot** be bookmarked

HTTP/2

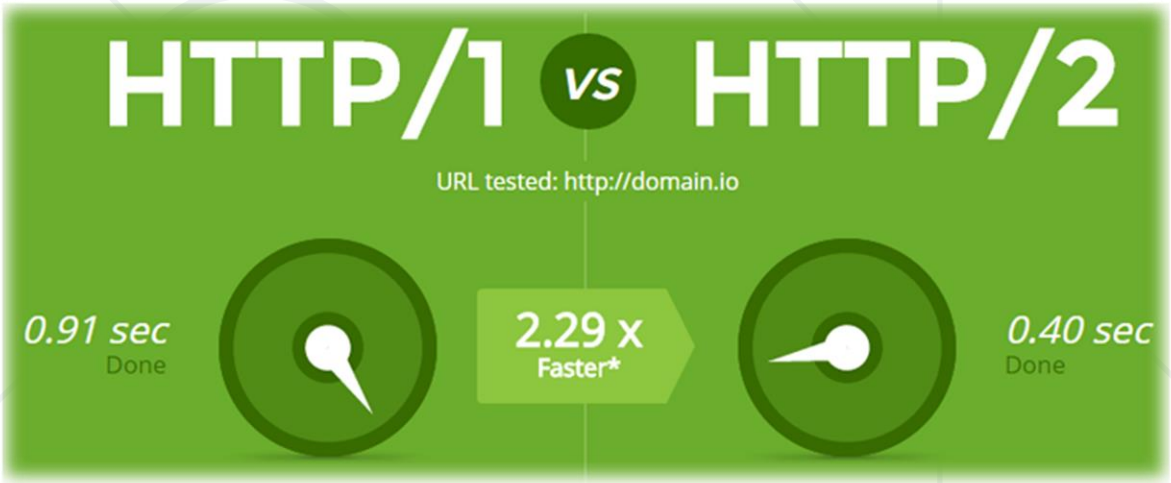
What's HTTP/2? What's New? What's Better?

What's HTTP/2

- **HTTP/2** (originally named **HTTP/2.0**) is a major revision of the **HTTP** network protocol used by the **World Wide Web**
  - Supported by most of the popular web browsers (Chrome, Mozilla, Opera, ...)
  - Fast & optimized
  - Meets modern web usage requirements
  - Completely Backwards-Compatible
- As of Apr 2023, **40%** of all the websites support **HTTP/2** (W3Techs statistics)

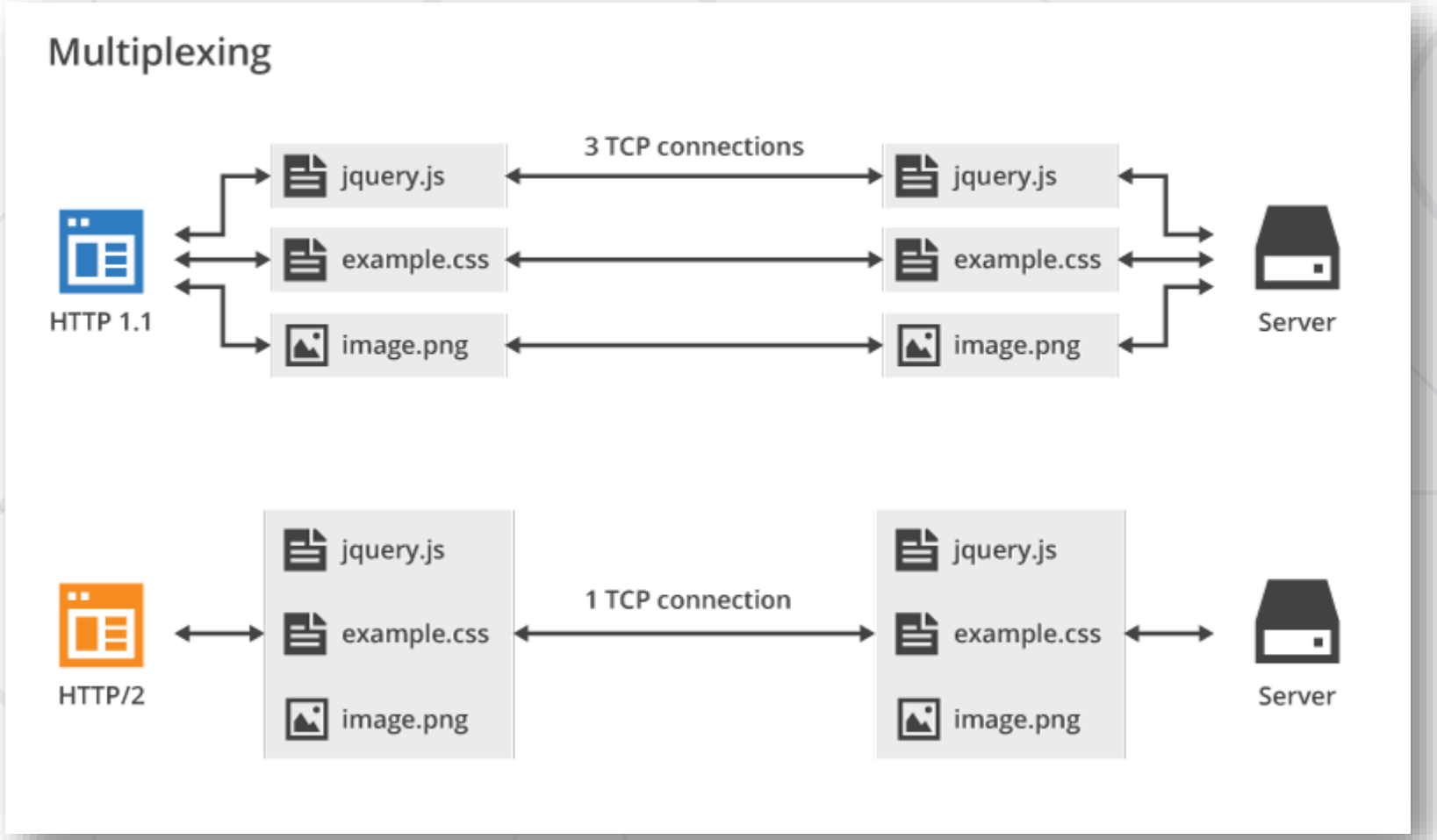
What's New?

- **HTTP/2** is meant to erase the need of maintaining complex server infrastructures in order to perform well
- **HTTP/2** communicates in binary data frames
- **HTTP/2** introduces several new important elements
  - HTTP/2 Multiplexing
  - HTTP/2 Header Compression
  - HTTP/2 Server Push



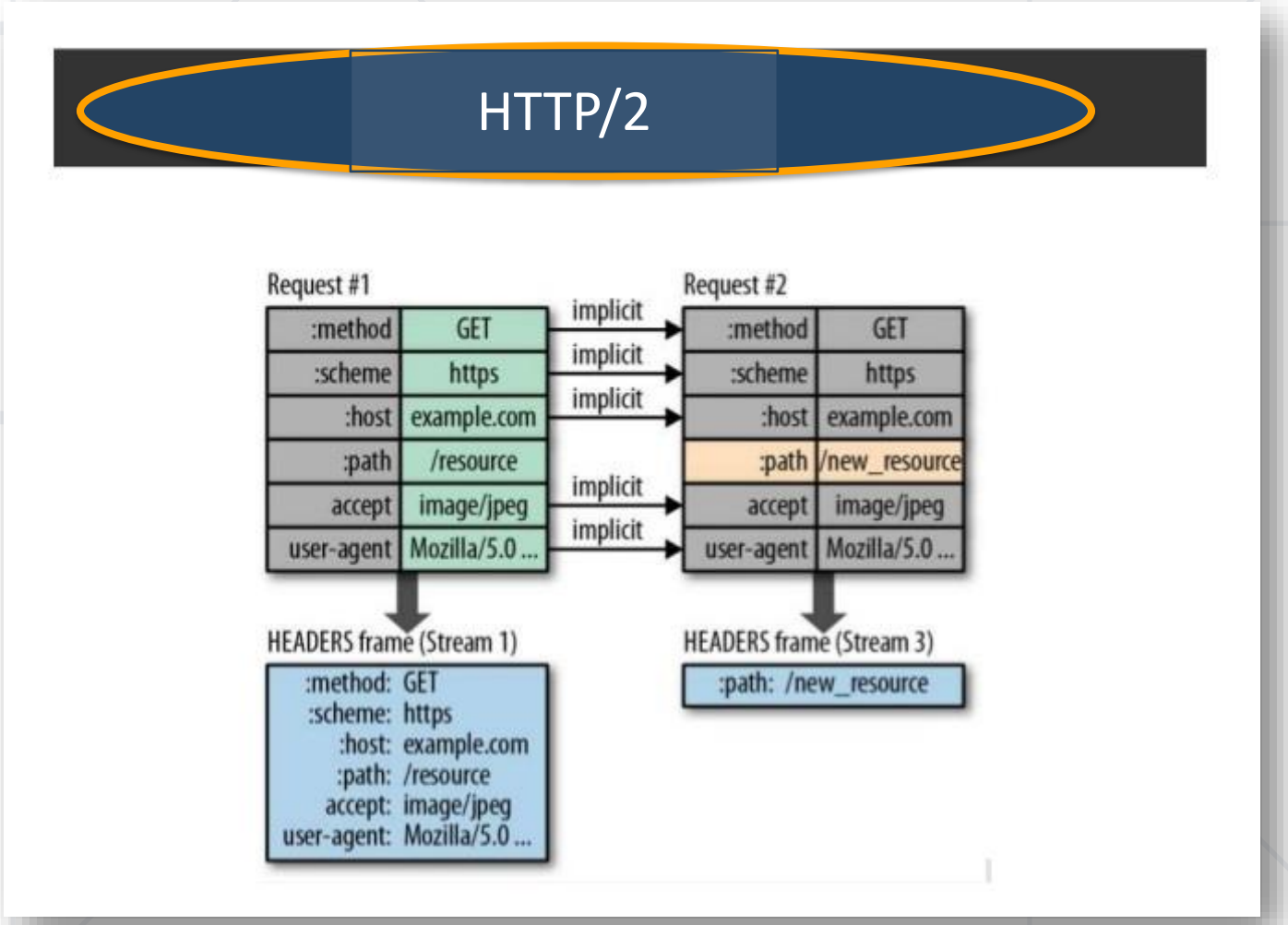
HTTP/2 Multiplexing

- The art of handling multiple streams over a **single** TCP connection



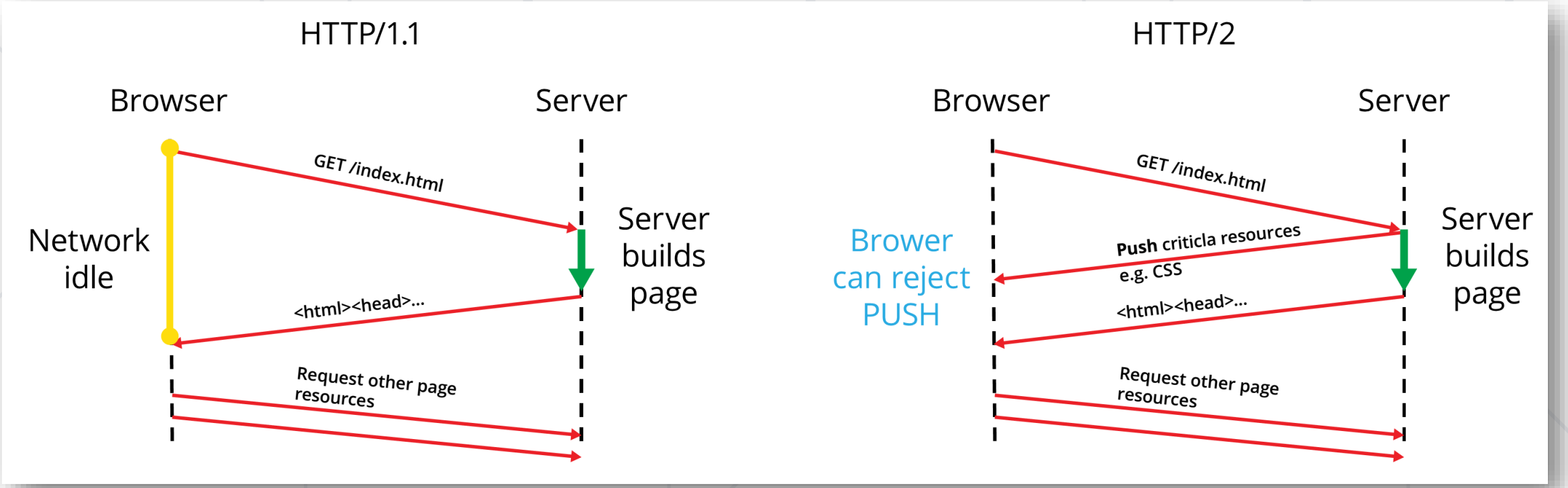
HTTP/2 Header Compression

- **HTTP/2** maintains a **HTTP Header Table** across requests
- Optimizes communication drastically
- The process is essentially a **de-duplication**, rather than compression



HTTP/2 Server Push

- **HTTP/2 Server Push** == the process of sending resources to clients, without them having to ask for it



HTTP/3

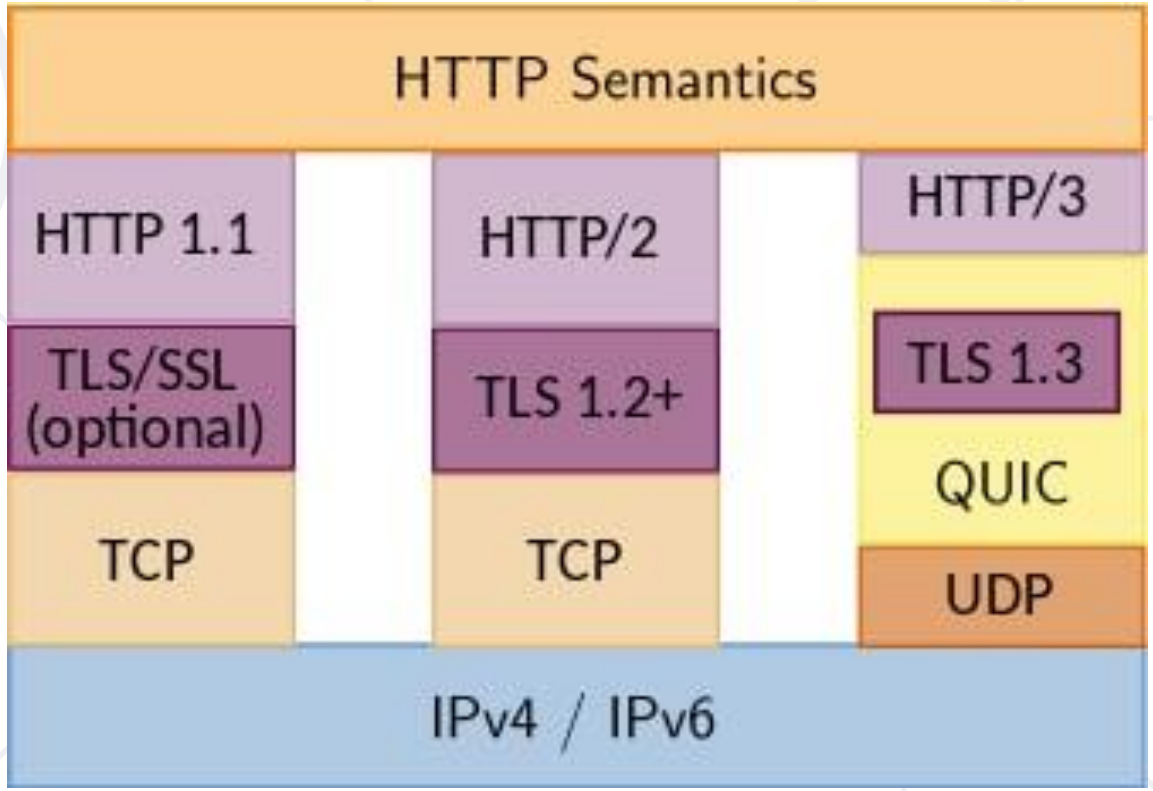
What's HTTP/3

What's HTTP/3

- **HTTP/3** is a new standard in development that will affect how web browsers and servers communicate
- As of Apr 2023, HTTP/3 is supported by 26% of web browsers (W3Techs statistics)
- Uses **QUIC** Protocol

What's new?

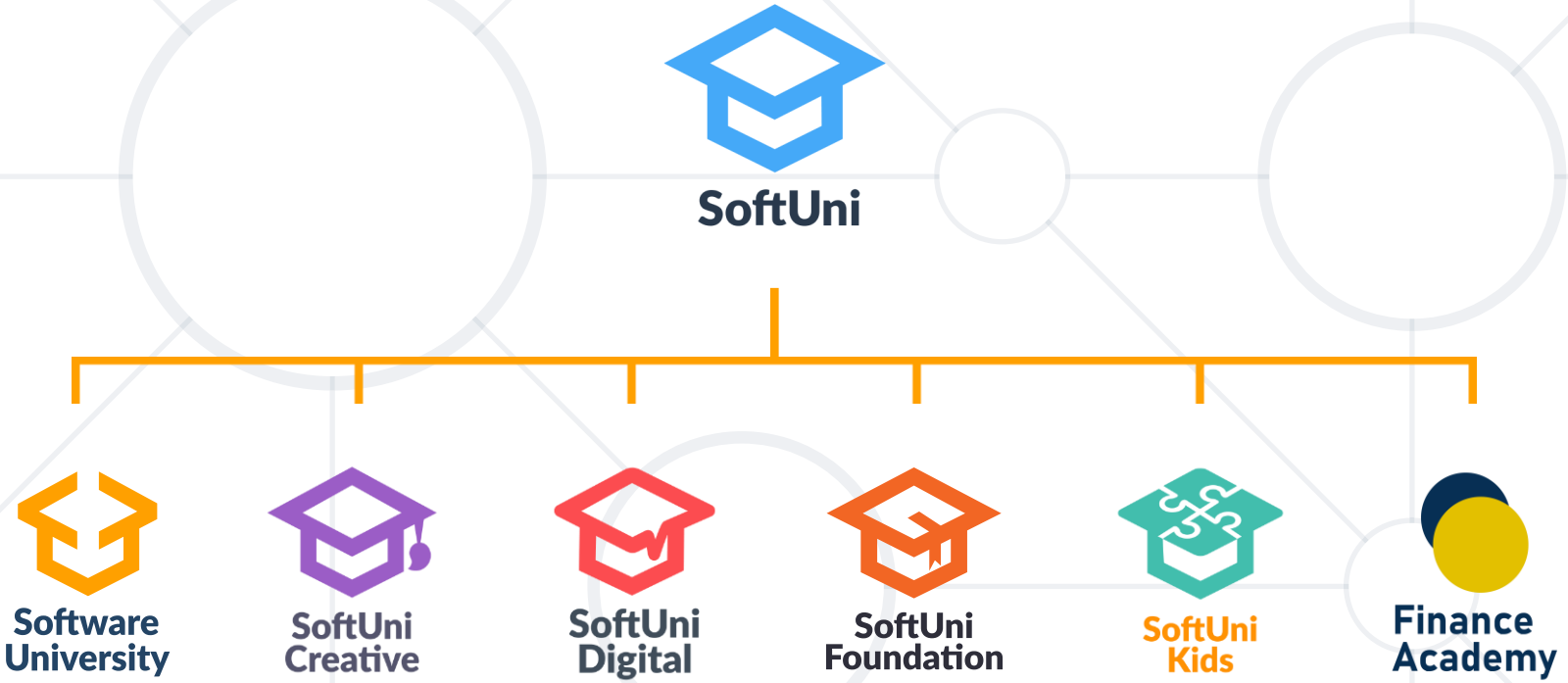
- **Significant upgrades** for user experience
  - Decreasing the effects of packet loss
  - Workaround for the slow performance, e. g., when a smartphone switches from Wi-Fi to cellular data
- **Performance**, **Reliability** and **Security**



Summary

- HTTP
  - HTML Forms & Actions
  - URLs
  - Request & Response
  - MIME & Media Types
- Web Server
  - Web Communication

Questions?



SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SUPER HOSTING .BG

createX

Postbank

Решения за твоето утре

DRAFT KINGS

THE CROWN IS YOURS

VIVACOM

AMBITIONED

INDEAVR

Serving the high achievers

PHAR VISION

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://about.softuni.bg/
- © Software University – https://softuni.bg