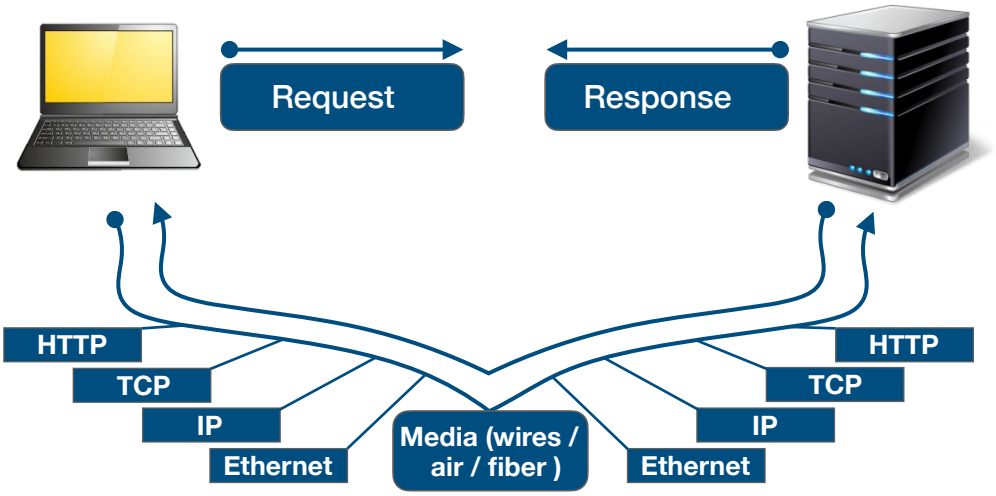
HTTP Protocol
Handling Requests, Constructing Responses, HTTP/2

Table of Content

1. HTTP Basics
2. URL
3. HTTP Request
4. HTTP Response
5. MIME and Media types
6. HTTP Tools
7. Web Server
8. HTML Forms
9. HTTP/2

HTTP Basics
Request and Responses

Hyper Text Transfer Protocol



HTTP Request Methods

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

Method	Description
GET	Retrieve / load a resource
POST	Create / store a resource
PUT	Update a resource
PATCH	Partially update a resource
DELETE	Remove a resource

HTTP Conversation: Example

- HTTP request

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

The empty line denotes the end of the response header

- HTTO response:

```
HTTP/1.1 200 OK
Date: Sat, 14 Oct 2023 14:09:01 GMT
Server: Microsoft-HTTPAPI/2.0
Last-Modified: Tue, 13 Jul 2014 17:44:12 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)

http://localhost:8080/demo/index.html?id=33&lang=en#lecture

Protocol · Host · Port · Path · Query String · Fragment

- URL is a formatted string, consisting of:
  - Protocol for communicating (**http**, **ftp**, **https**...) – HTTP in most cases
  - Host or IP address (**www.google.com**, **gmail.com**, **127.0.0.1**, **web**)
  - Port (the default port is **80**) – a number in range [0...65535]
  - Path (**/forum**, **/path/index.html**)
  - Query string (**?id=33&lang=en**)
  - Fragment (**#lecture**) – used on the client to navigate to some section

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**"

    Пешо-是-Peshev

  - URL-encoded string:

    %D0%B0%BA%D0%D0%9D-%88%D0%E7-AlaBala

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
```

```
https://bg.wikipedia.org/wiki/
%D0%A3%D0%BD%D0%B8%D1%84%D0%B8%D1%86%D0%B8%D1%80%D0%B0%D0%BD_%D0%BB%
D0%BE%D0%BA%D0%B0%D1%82%D0%BE%D1%80_%D0%BD%D0%B0_%D1%80%D0%B5%D1%81%D
1%83%D1%80%D1%81%D0%B8
```

- Some invalid URLs:

```
http://www.google.bg/search?&q=C# .NET 4.0
```

Should be: ?q=C%23+.NET+4.0

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

Should be: ?q=%D0%B1%D0%B8%D1%80%D0%B0%0A

HTTP Request
What is a HTTP Request?

HTTP Request Message

- Request message sent by a client consist 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 Message – Example

- Example of HTTP GET request:

```
GET /index.html HTTP/1.1
Host: localhost
<CRLF>
```

HTTP request line

HTTP request headers

HTTP request body is empty

- 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

The 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 consist of:
  - HTTP response status line
    - Protocol version
    - Status code
    - Status phase
  - 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 Message – Example

- Example of HTTP response from the Web server:

```
HTTP/1.1 200 OK
Date: Fri, 17 Jul 2020 16:09:22 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 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

- Example of HTTP response with error result:

```
HTTP/1.1 404 Not Found

Date: Fri, 17 Jul 2020 16:09:22 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 /index.html 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

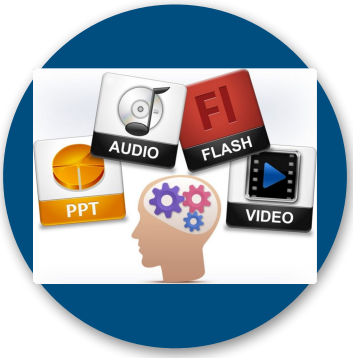
- With the Content-Type response header the server specifies how the output should be processed
- Examples:

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

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

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

This will download a PDF file named Report-April-2021.pdf



MIME and Media Types
Multi-Purpose Internet mail Extensions

What is MIME?

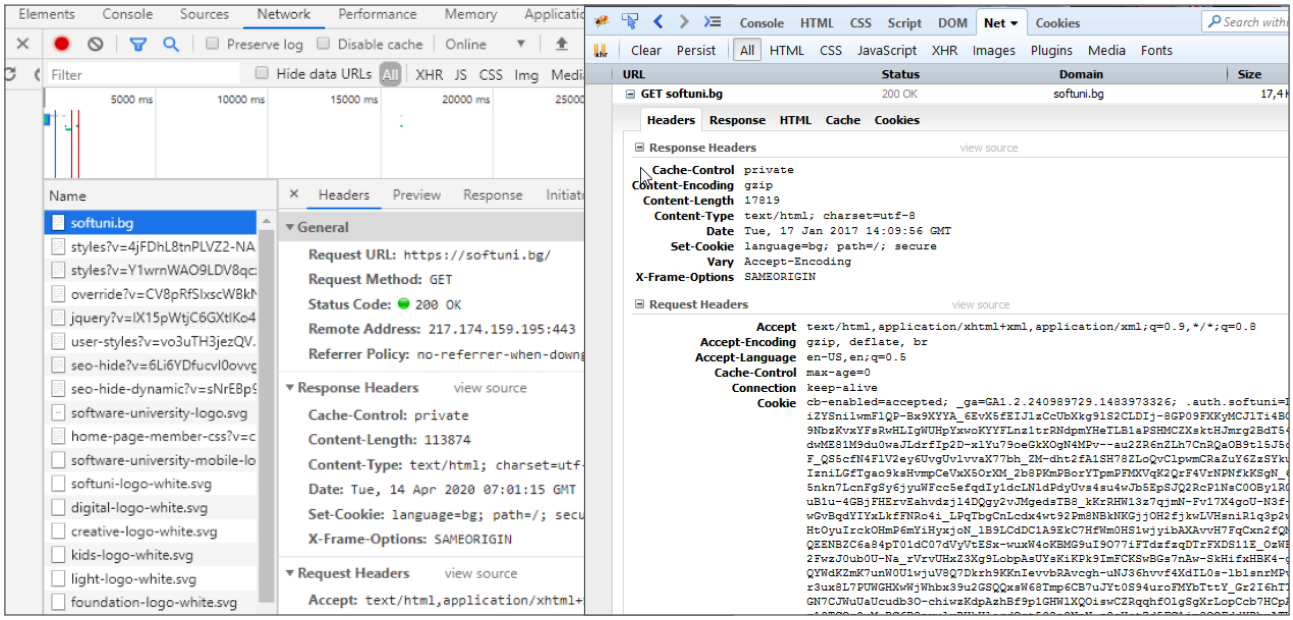
- MIME == Multi-Purpose Internet Mail Extensions
  - Internet standard for encoding resources
  - Originally developed for email attachments
  - Used in many Internet protocols like HTTP and SMTP
- MIME defines several concepts
  - Content-Type, e.g. text/html, image/gif, application/pdf
    - Content charset, e.g. utf-8, ascii, windows-1251
  - Content-Disposition, e.g. attachment; filename-logo.jpg

Common MIME Media Types

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

HTTP Tools
Tools for Developers

HTTP Tools for Developers – Browser



Chrome Developer Tools

Firebug

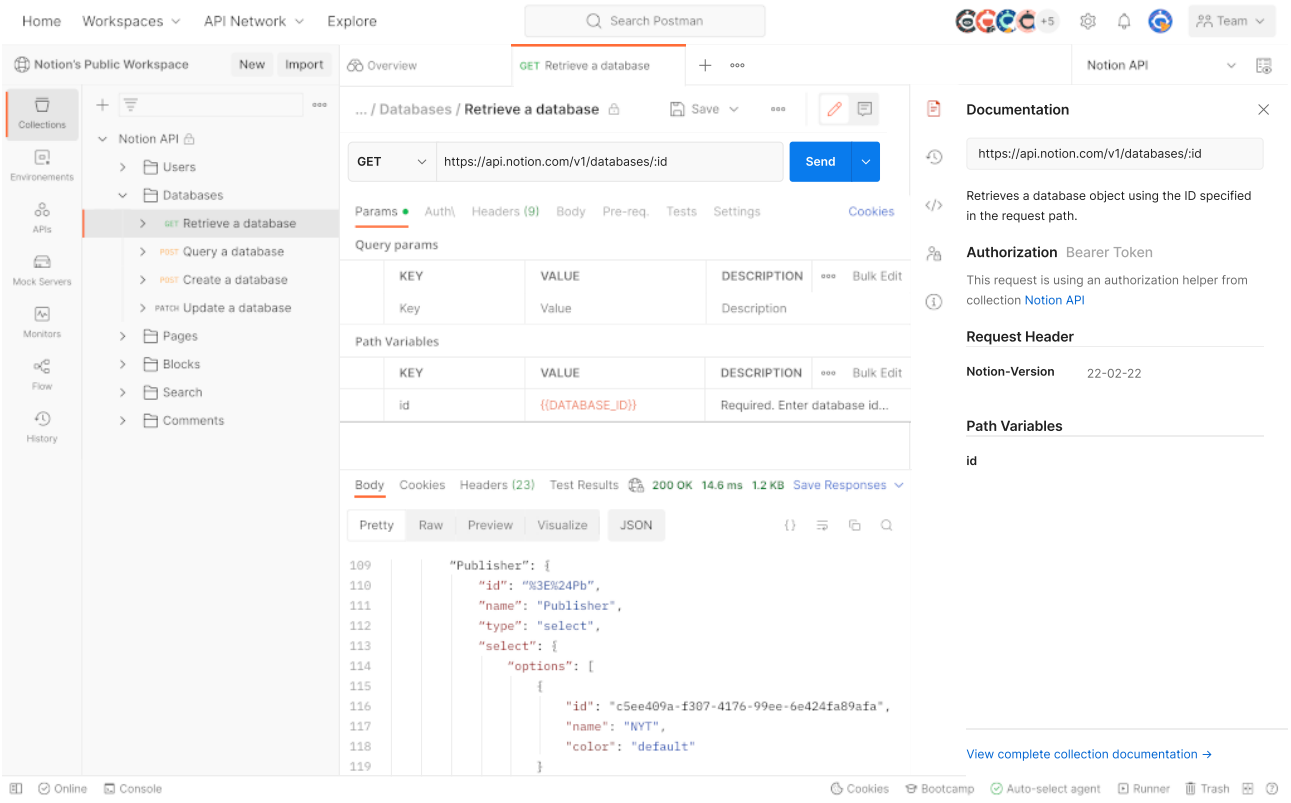
Insomnia Rest

Postman

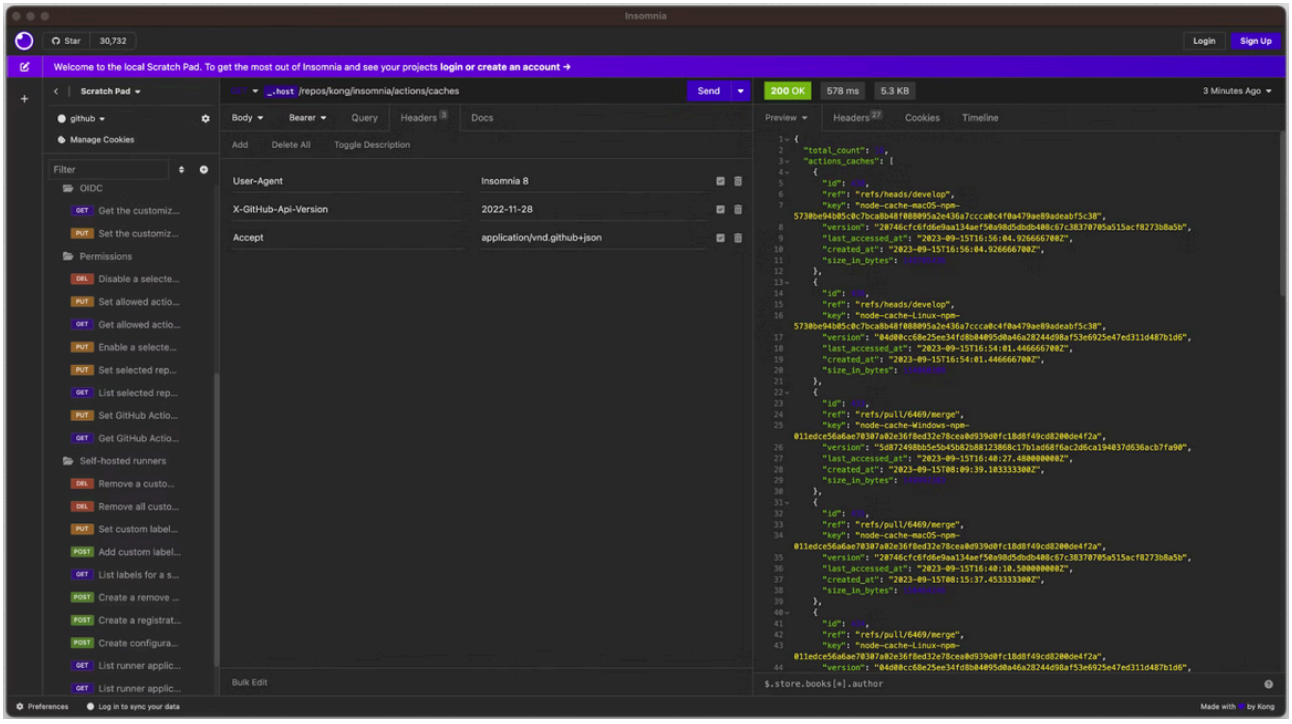
RESTClient

Tiny RestClient

HTTP Tools for Developers – Postman



HTTP Tools for Developers – Insomnia

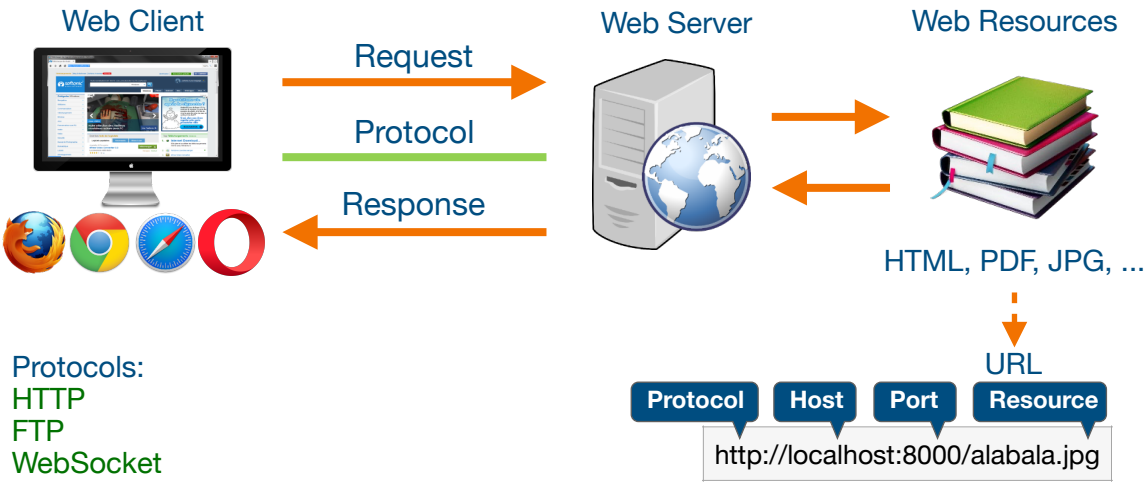


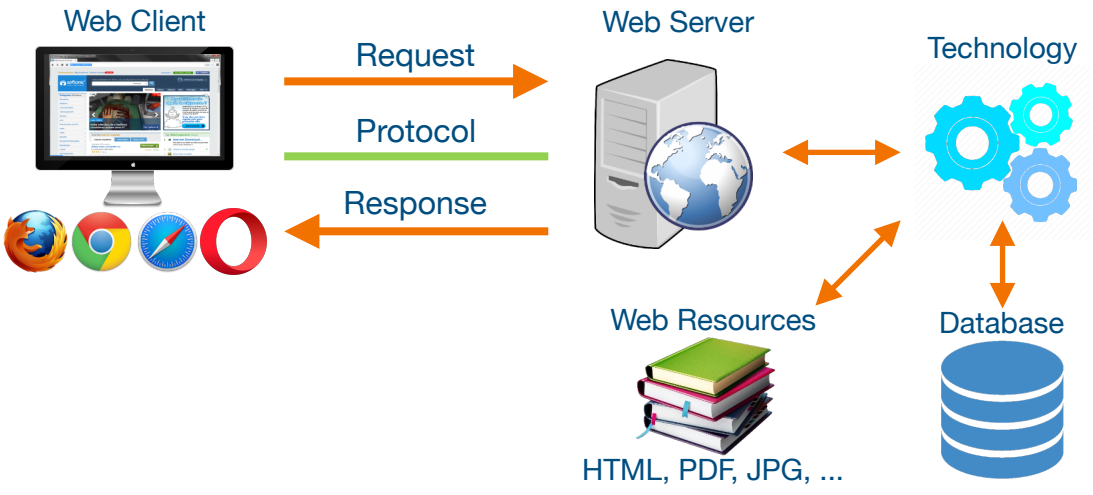
Web Server

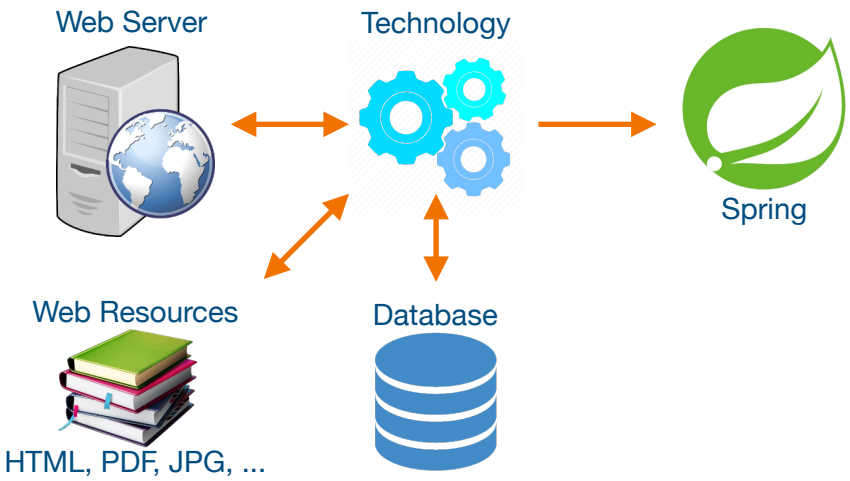
What is a Web Server?

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

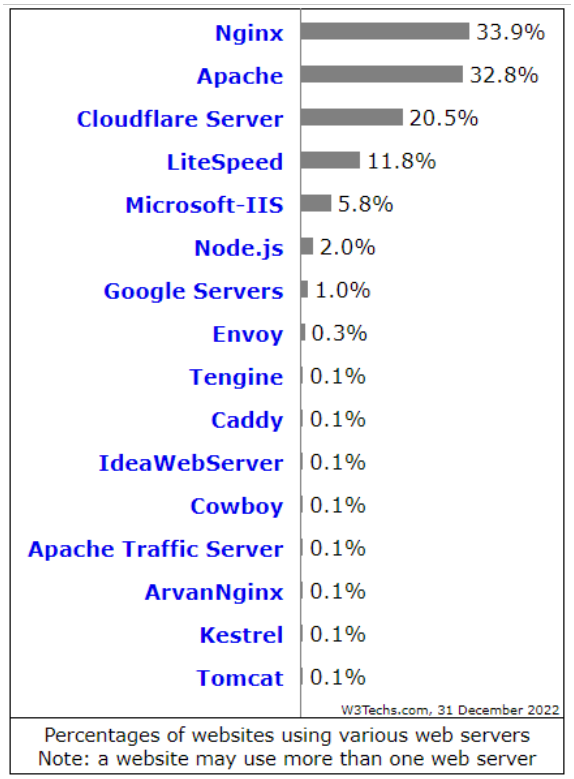
Web Server Work Model







Most Popular Web Servers (W3Techs)



Most Popular Web Servers (NetCraft)

Developer	January 2022	Percent	February 2022	Percent	Change
nginx	377,019,054	32.29%	364,956,731	31.10%	-1.19
Apache	279,709,815	23.95%	277,928,961	23.68%	-0.27
OpenResty	80,238,470	6.87%	90,652,376	7.72%	0.85
Cloudflare	60,881,028	5.21%	62,423,819	5.32%	0.11

Developer	November 2022	Percent	December 2022	Percent	Change
nginx	300,890,891	26.51%	295,366,783	26.25%	-0.26
Apache	242,899,324	21.40%	235,541,408	20.93%	-0.47
Cloudflare	101,367,889	8.93%	102,829,746	9.14%	0.21
OpenResty	91,612,799	8.07%	92,711,293	8.24%	0.17

HTTP Forms
Form Method and Action

HTML Forms – Action Attribute

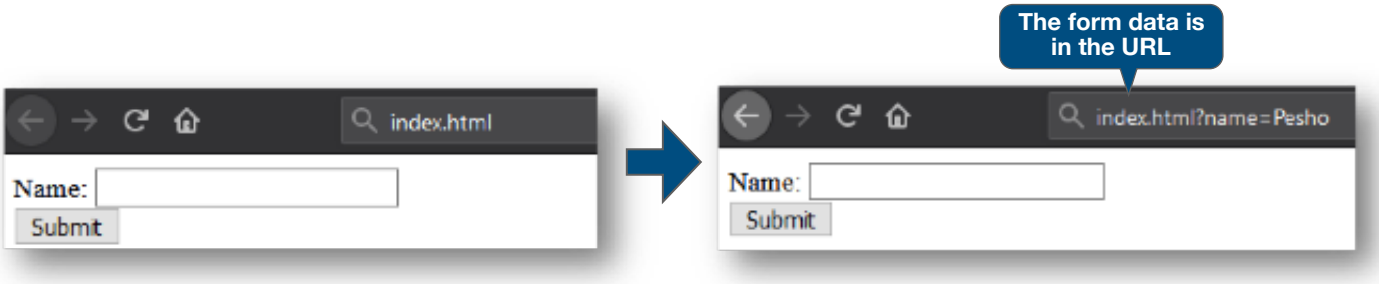
- Defines where to submit the form data:

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

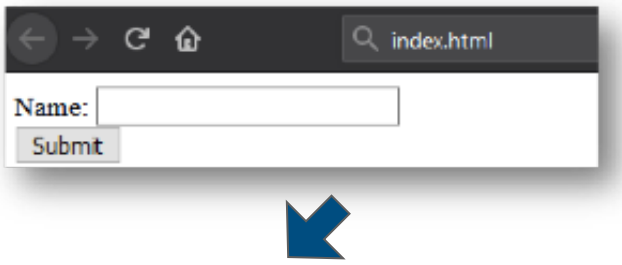
HTML Forms – Method Attribute

- 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>
```



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



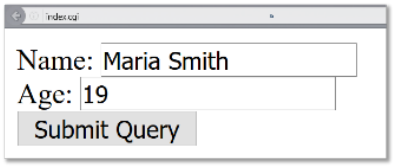

```
POST http://localhost/index.html HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 10

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" value="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

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

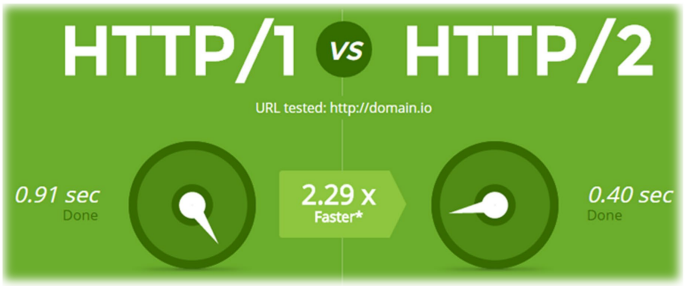
What's HTTP/2

- HTTP/2 (originally named HTTP/2.0) 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 Jan 2021, 50% of all the websites support HTTP/2 (W3Techs statistics).

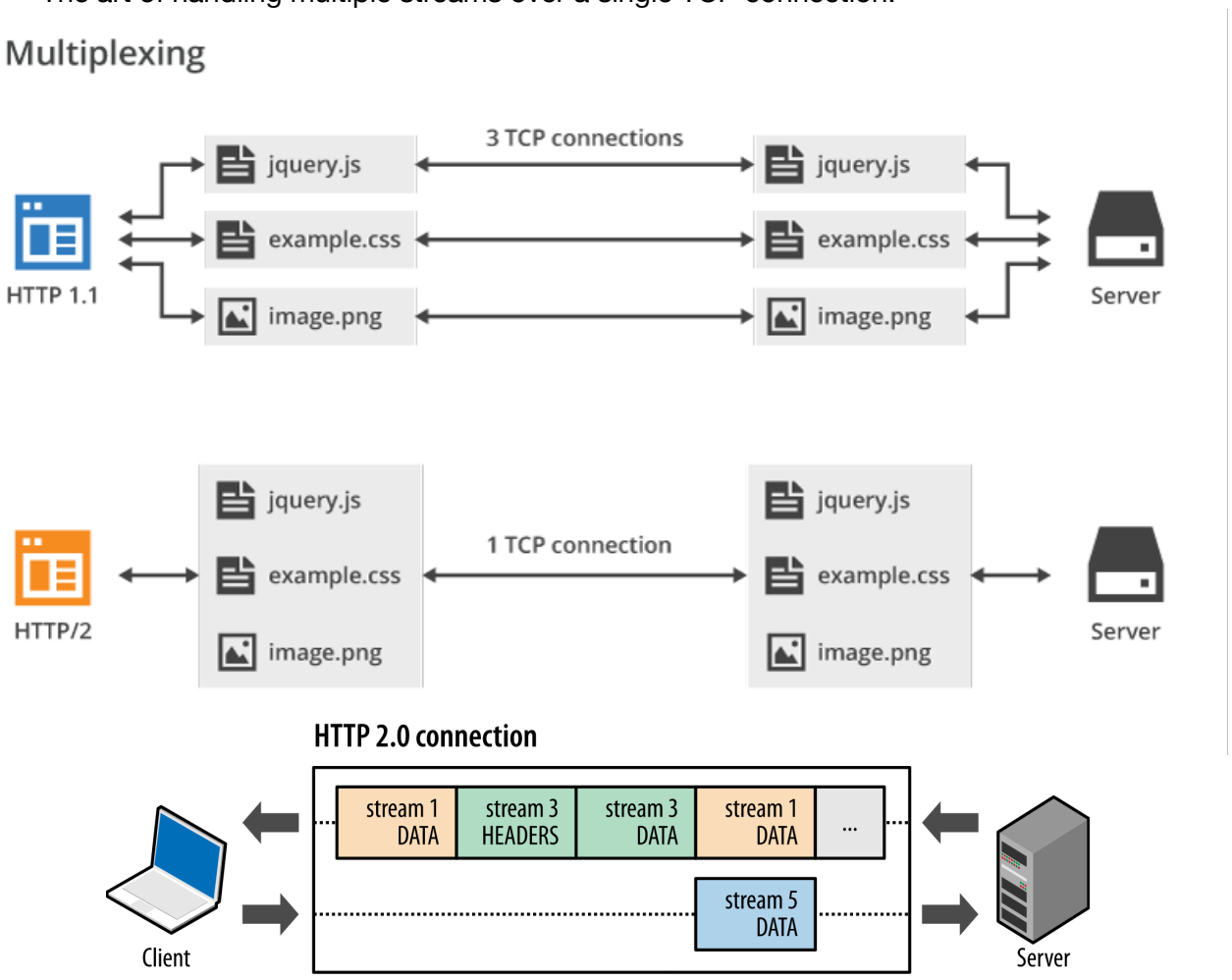
What's New?

- 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



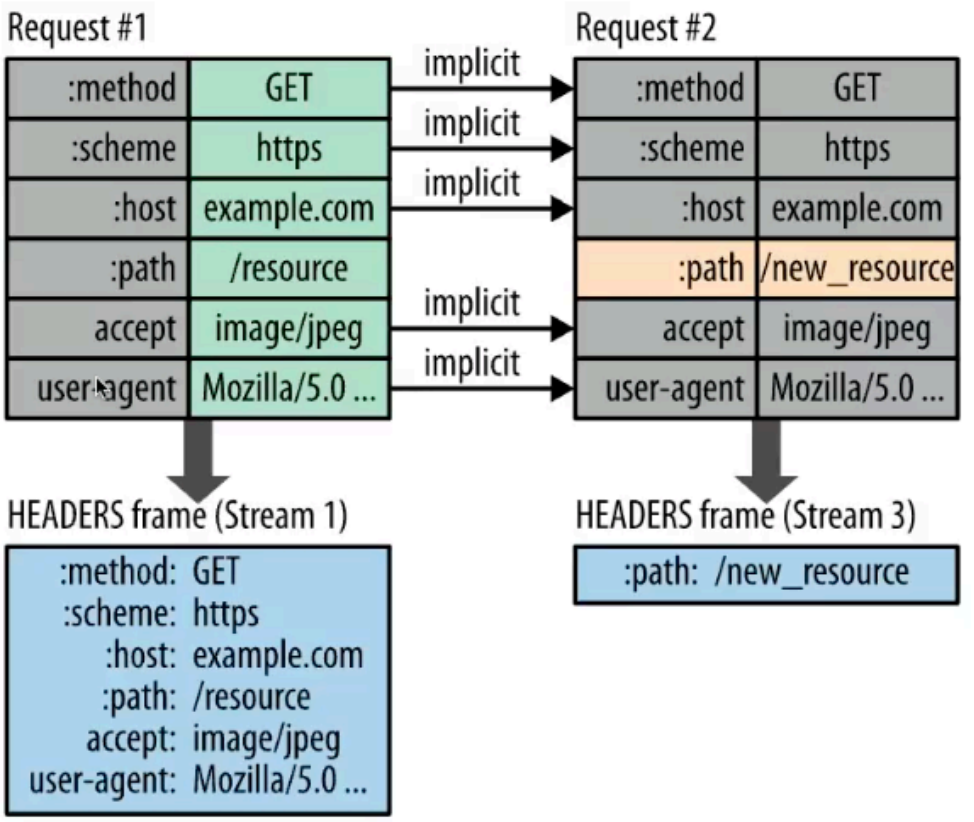
HTTP/2 Multiplexing

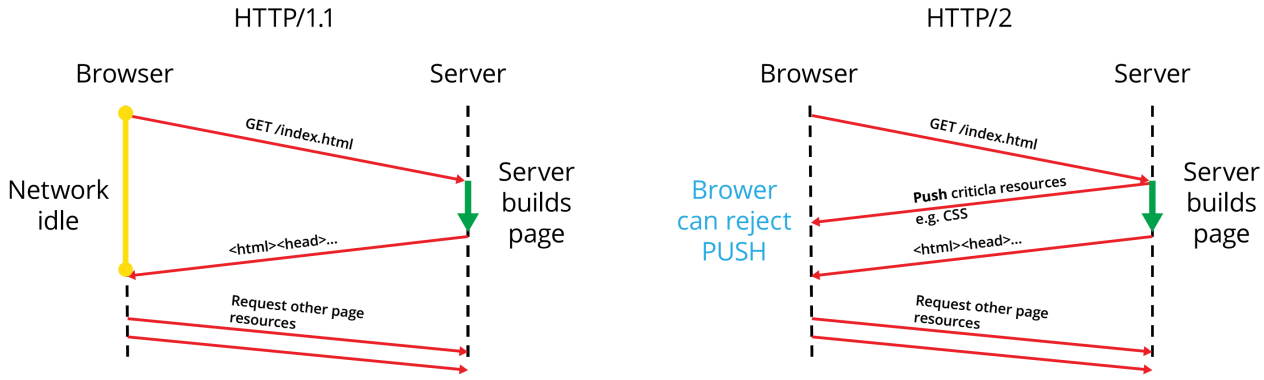
- The art of handling multiple streams over a single TCP connection.



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





What's HTTP/3

- HTTP/3 is a new standard in development that will affect how web browsers and servers communicate
- Significant upgrades for user experience
- Performance, Reliability, and Security

- HTTP/3 runs on QUIC, a new transport protocol designed for mobile-heavy Internet usage

Summary

- HTTP:
  - HTML Forms & Actions
  - URLs
  - Request
  - Response
- Web Server:
  - Web Communication