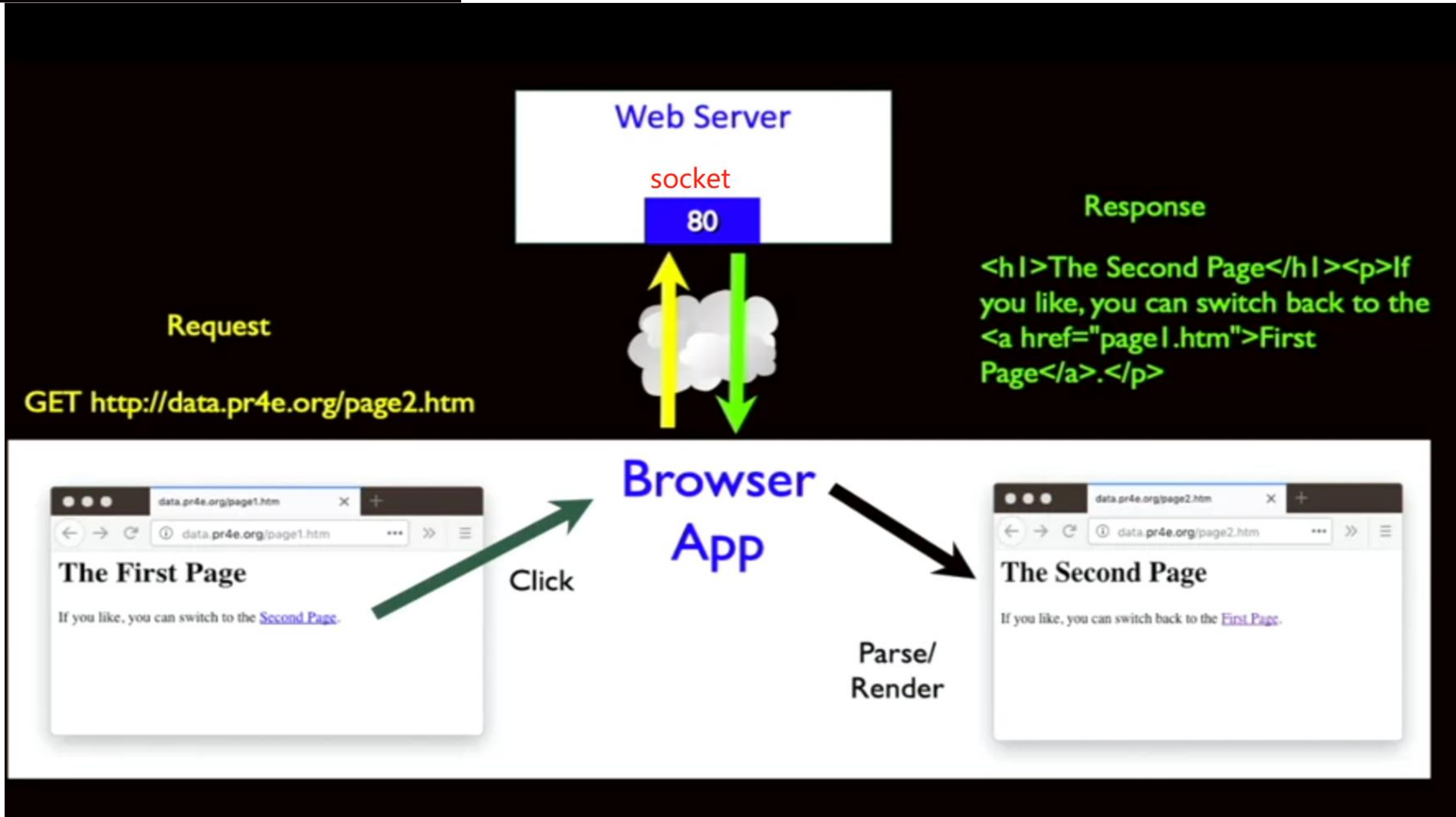
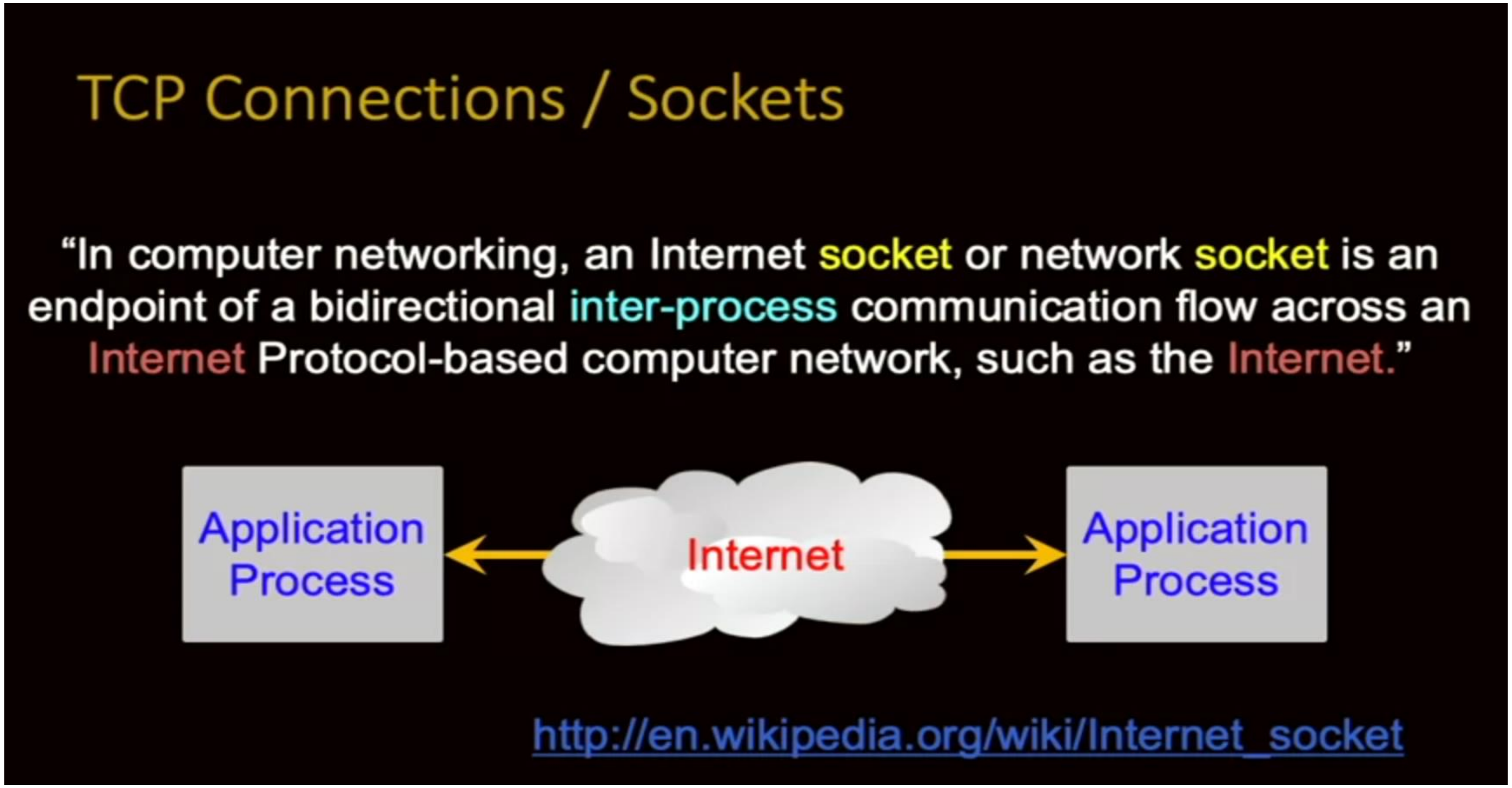
# Introduction to Dynamic Web Content

## Getting Data from the Server

- Each time the user clicks on an anchor tag with an href = value to switch to a new page, the browser makes a connection to the web server and issues a "GET" request - to GET the content of the page at the specified URL.
- The server returns the HTML document to the browser, which formats and displays the document to the user.


Web Server
socket
80
Request
GET http://data.pr4e.org/page2.htm
Response
<h1>The Second Page</h1><p>If you like, you can switch back to the <a href="page1.htm">First Page</a>.</p>
data.pr4e.org/page1.htm
The First Page
If you like, you can switch to the Second Page.
Click
Browser App
Parse/ Render
data.pr4e.org/page2.htm
The Second Page
If you like, you can switch back to the First Page.

## Network Sockets and Connections

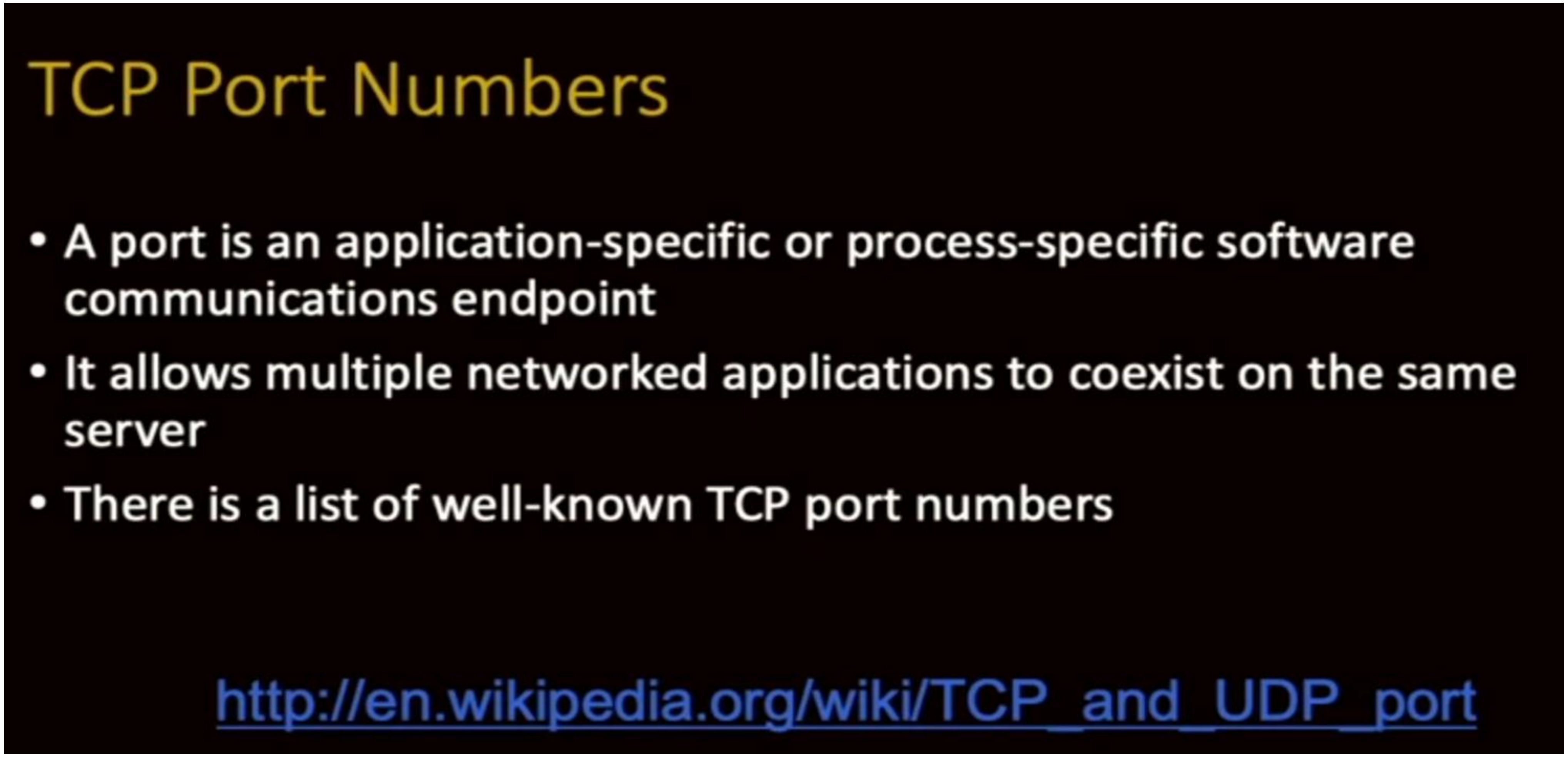


TCP Connections / Sockets

"In computer networking, an Internet socket or network socket is an endpoint of a bidirectional inter-process communication flow across an Internet Protocol-based computer network, such as the Internet."

http://en.wikipedia.org/wiki/Internet_socket

# Network Sockets and Connections

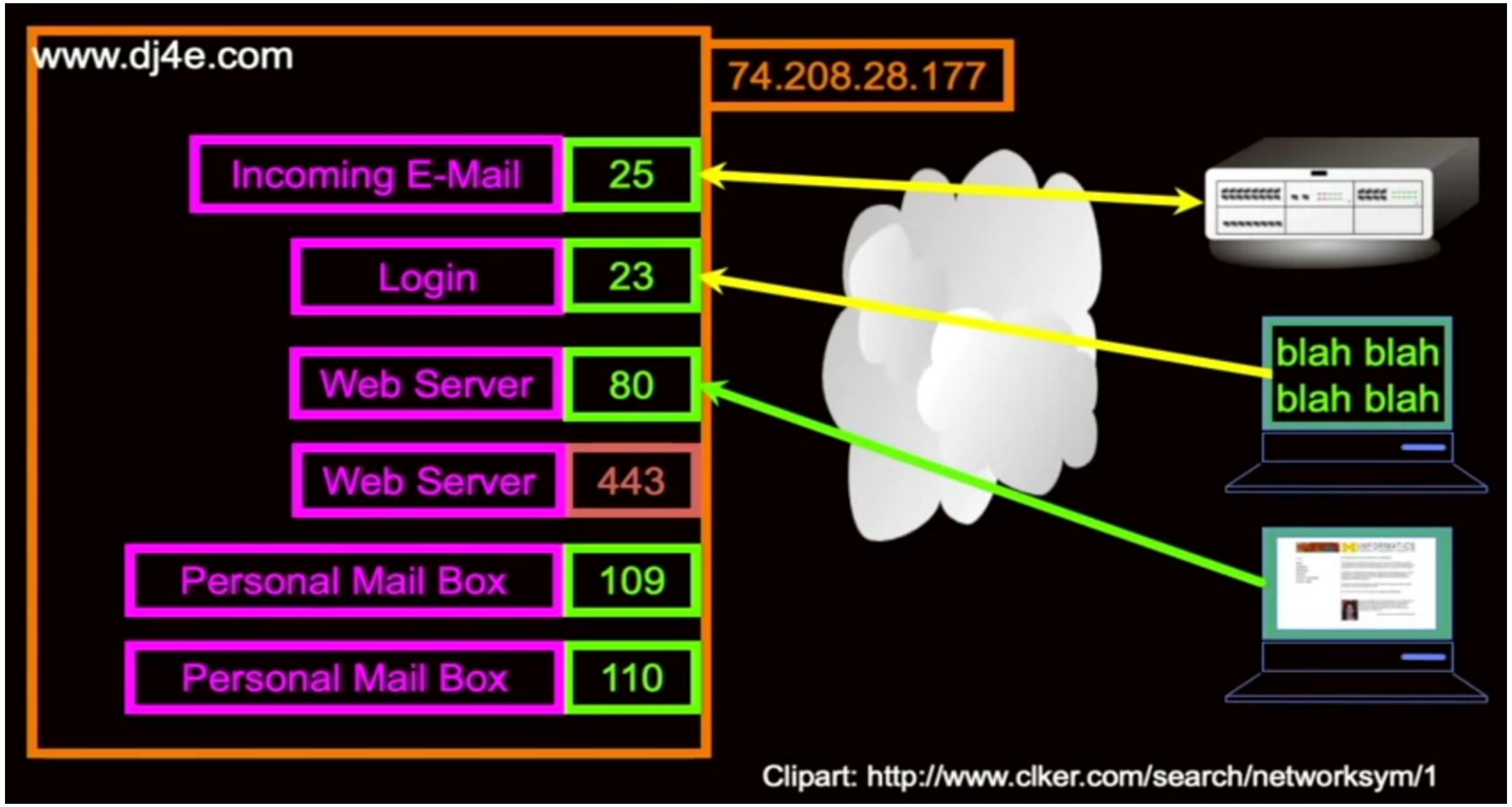


## TCP Port Numbers

- A port is an application-specific or process-specific software communications endpoint
- It allows multiple networked applications to coexist on the same server
- There is a list of well-known TCP port numbers

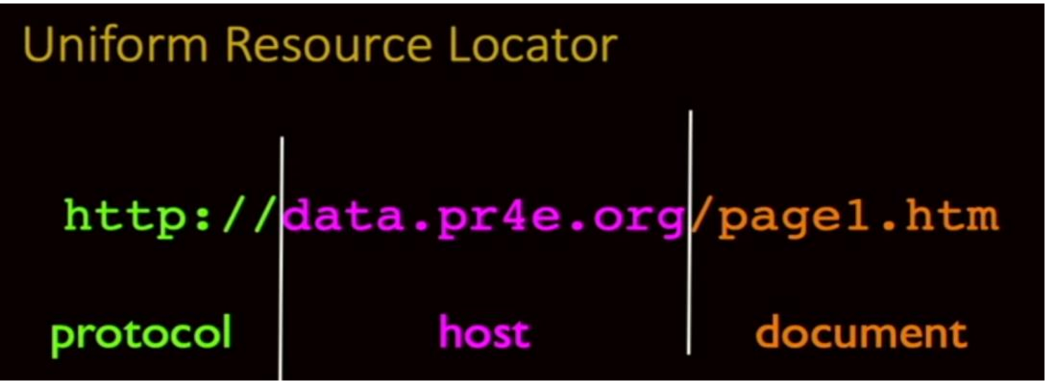
http://en.wikipedia.org/wiki/TCP_and_UDP_port

# Network Sockets and Connections



Left side is server side, right side is client side. Different color of arrow represents different protocol. 443 for https, 80 for http.

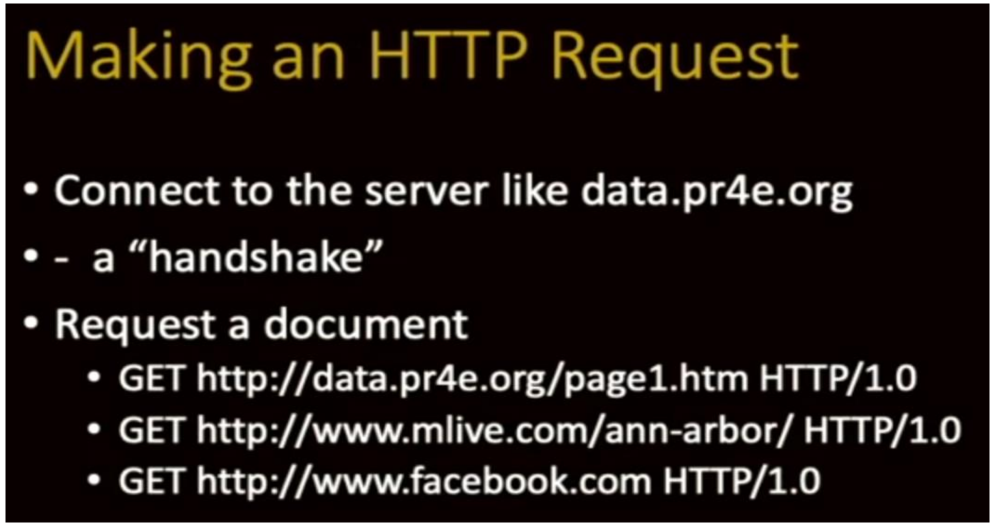
# HyperText Transfer Protocol



## Uniform Resource Locator

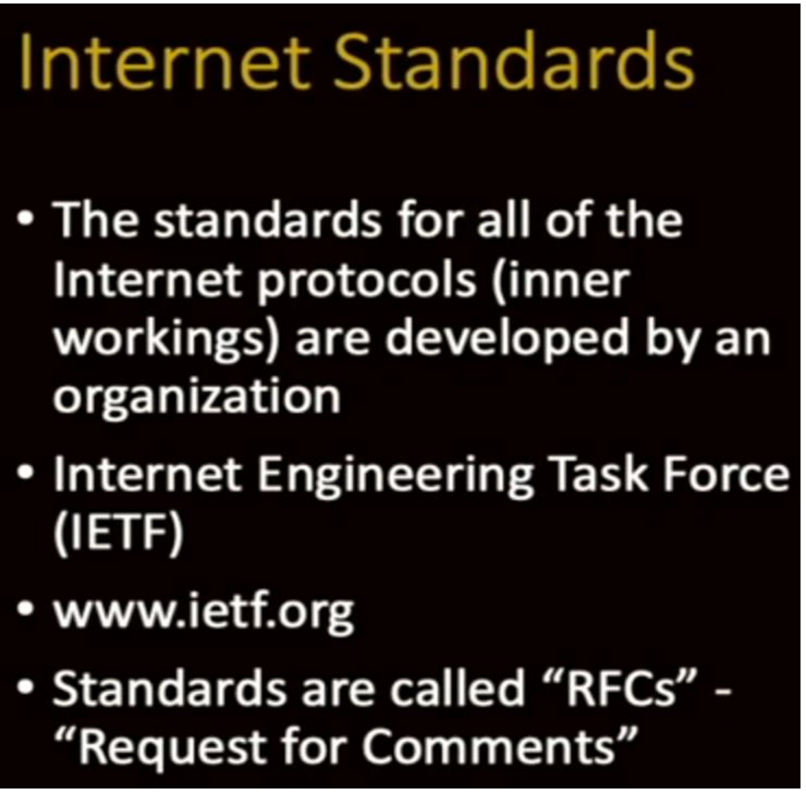
```
http://data.pr4e.org/page1.htm
```

protocol | host | document



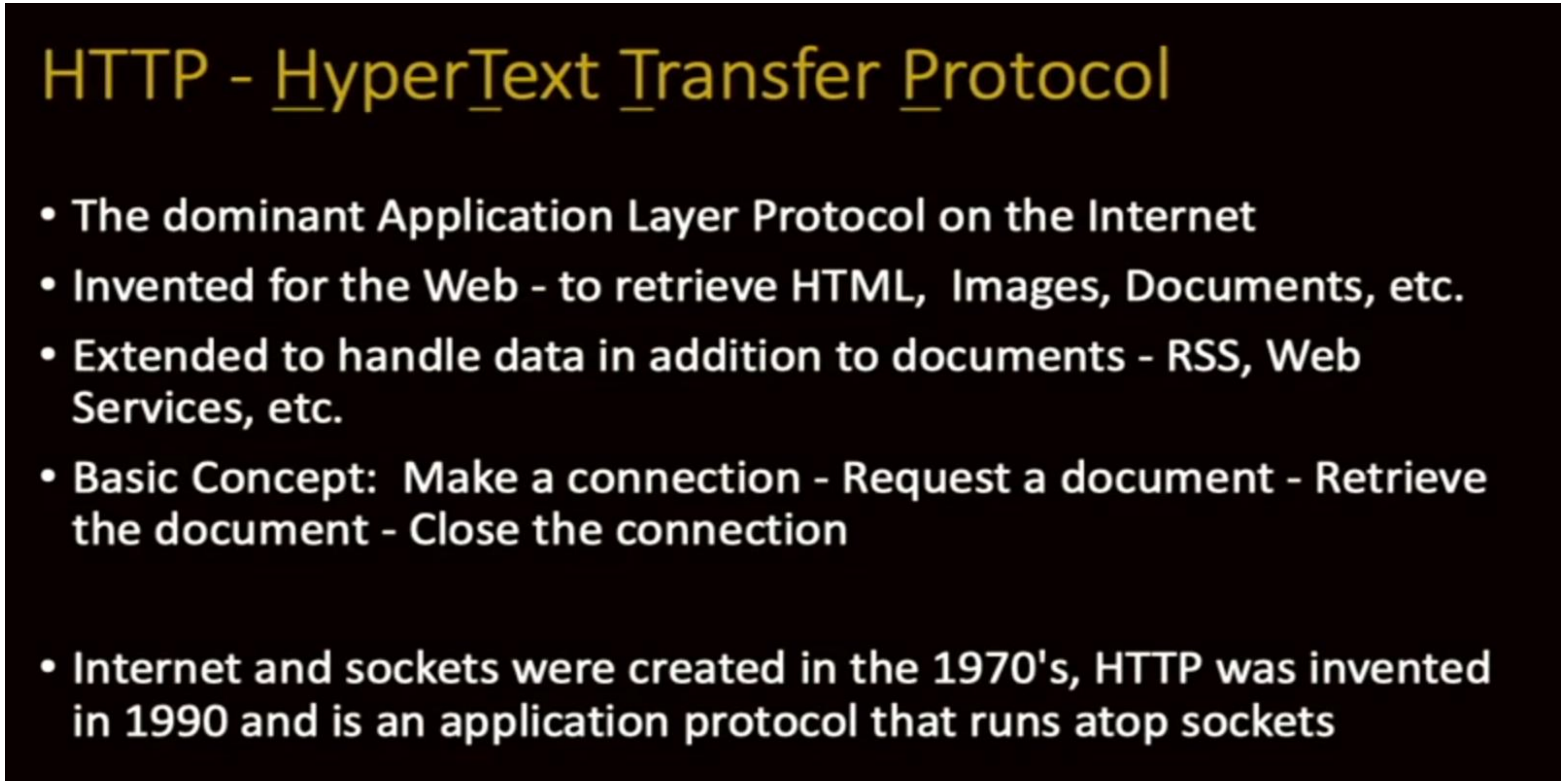
## Making an HTTP Request

- Connect to the server like data.pr4e.org
- - a “handshake”
- Request a document
  - GET http://data.pr4e.org/page1.htm HTTP/1.0
  - GET http://www.mlive.com/ann-arbor/ HTTP/1.0
  - GET http://www.facebook.com HTTP/1.0



## Internet Standards

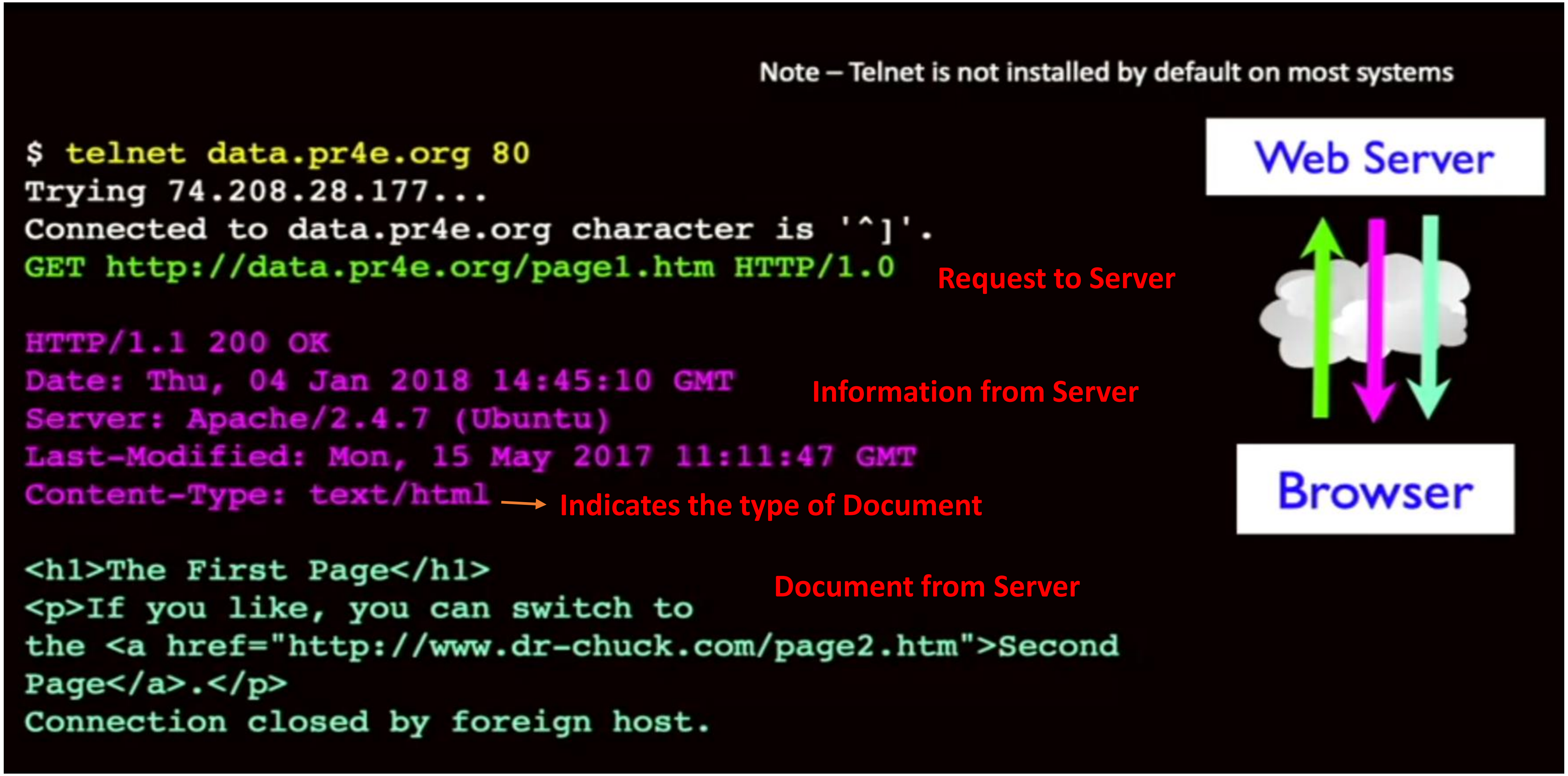
- The standards for all of the Internet protocols (inner workings) are developed by an organization
- Internet Engineering Task Force (IETF)
- www.ietf.org
- Standards are called “RFCs” - “Request for Comments”



## HTTP - HyperText Transfer Protocol

- The dominant Application Layer Protocol on the Internet
- Invented for the Web - to retrieve HTML, Images, Documents, etc.
- Extended to handle data in addition to documents - RSS, Web Services, etc.
- Basic Concept: Make a connection - Request a document - Retrieve the document - Close the connection
- Internet and sockets were created in the 1970's, HTTP was invented in 1990 and is an application protocol that runs atop sockets

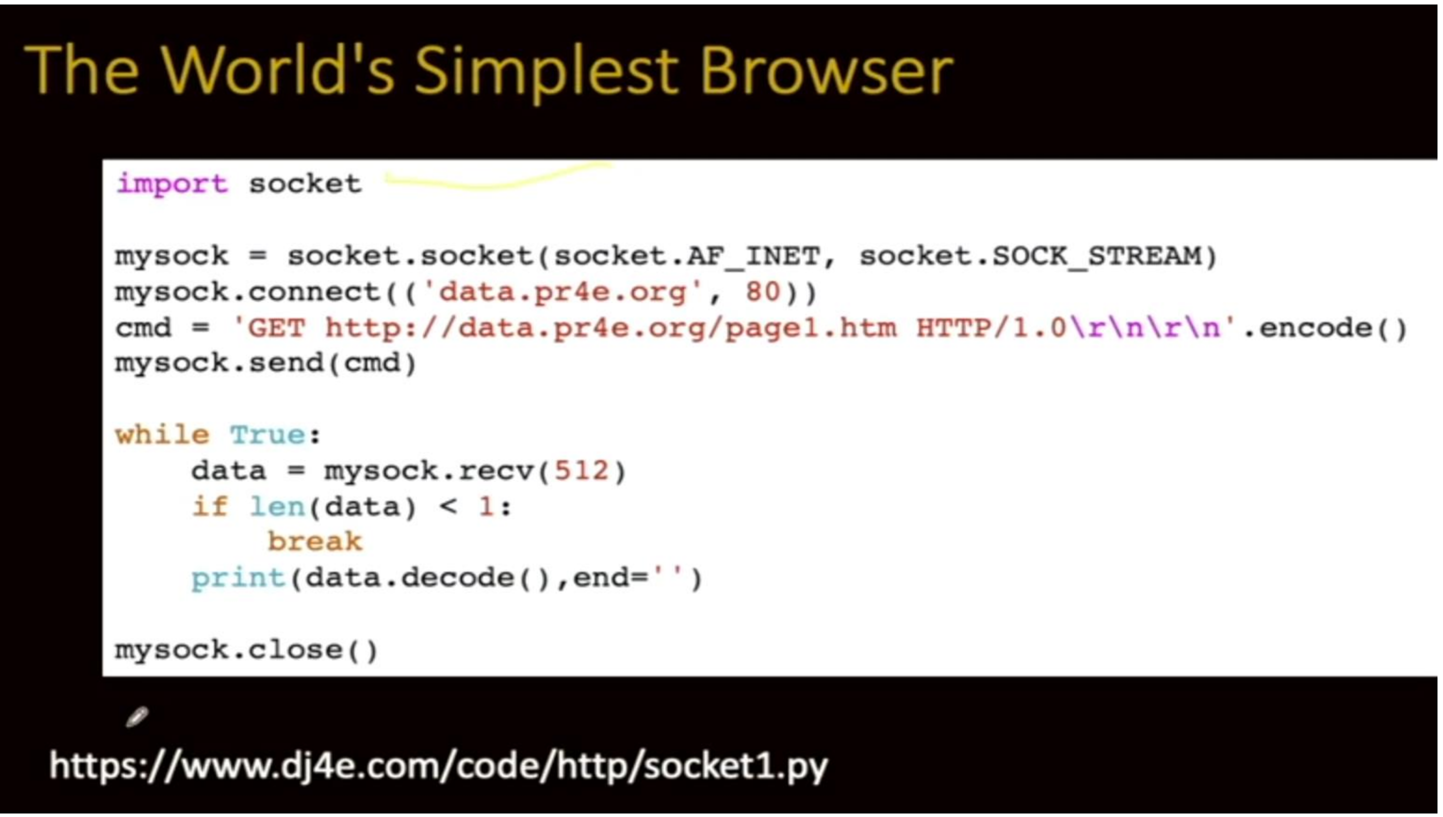
# HyperText Transfer Protocol (HTTP)



Note – Telnet is not installed by default on most systems

```
$ telnet data.pr4e.org 80
Trying 74.208.28.177...
Connected to data.pr4e.org character is '^]'.
GET http://data.pr4e.org/page1.htm HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 04 Jan 2018 14:45:10 GMT
Server: Apache/2.4.7 (Ubuntu)
Last-Modified: Mon, 15 May 2017 11:11:47 GMT
Content-Type: text/html

<h1>The First Page</h1>
<p>If you like, you can switch to
the <a href="http://www.dr-chuck.com/page2.htm">Second
Page</a>.</p>
Connection closed by foreign host.
```

# Building a Simple Web Browser in Python



## The World's Simplest Browser

```python
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET http://data.pr4e.org/page1.htm HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if len(data) < 1:
        break
    print(data.decode(),end='')

mysock.close()
```

https://www.dj4e.com/code/http/socket1.py

```python
mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org',80))
cmd = 'GET http://data.pr4e.org/page1.htm HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

while True:
    data = mysock.recv(512)
    if len(data) < 1:
        break
    print(data.decode(),end = '')

mysock.close()
```

```
HTTP/1.1 200 OK
Date: Wed, 30 Jun 2021 01:09:34 GMT
Server: Apache/2.4.18 (Ubuntu)
Last-Modified: Mon, 15 May 2017 11:11:47 GMT
ETag: "80-54f8e1f004857"
Accept-Ranges: bytes
Content-Length: 128
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Pragma: no-cache
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Connection: close
Content-Type: text/html

<h1>The First Page</h1>
<p>
If you like, you can switch to the
<a href="http://data.pr4e.org/page2.htm">
Second Page</a>.
</p>
```

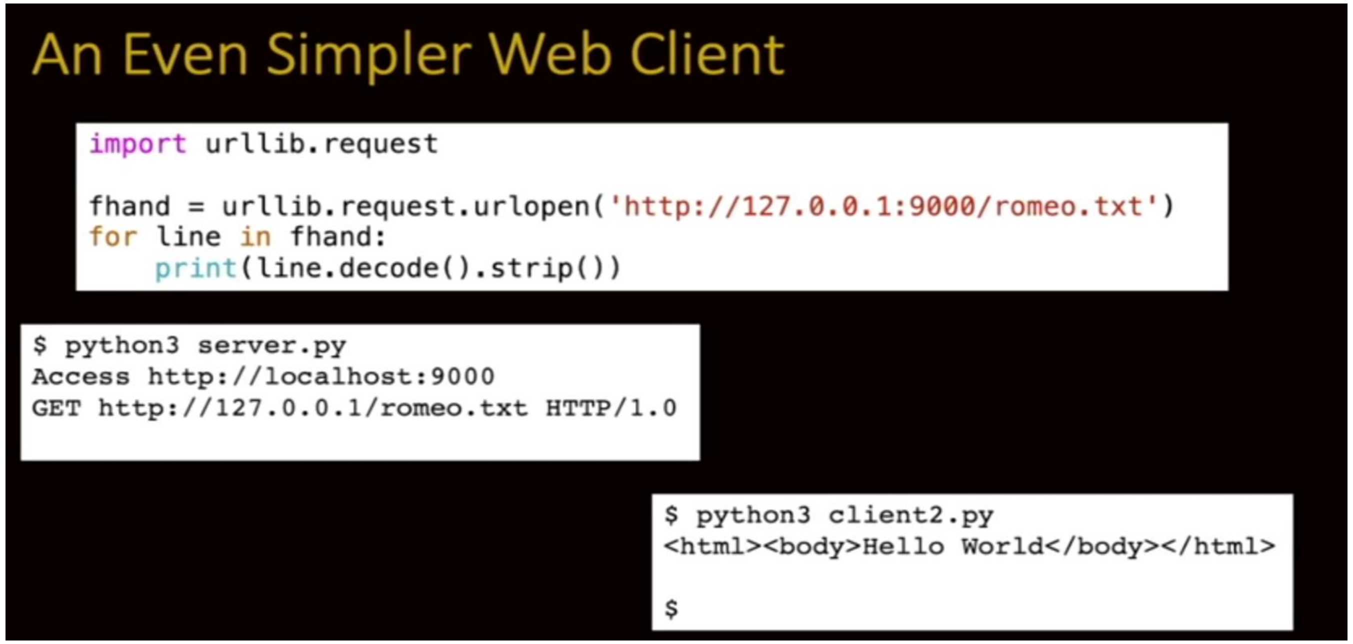
# Building a Simple HTTP Server in Python

```python
from socket import *

def createServer():
    serversocket = socket(AF_INET,SOCK_STREAM)
    try:
        serversocket.bind(('localhost',9000))
        serversocket.listen(5)
        while(1):
            (clientsocket,address) = serversocket.accept()

            rd = clientsocket.recv(5000).decode()
            pieces = rd.split('\n')
            if (len(pieces) > 0) : print(pieces[0])

            data = 'HTTP/1.1 200 OK \r\n'
            data += "Content-Type: text/html; charset=utf-8\r\n"
            data += '\r\n'
            data += '<html><body>Hello World</body></html>\r\n\r\n'
            clientsocket.sendall(data.encode())
            clientsocket.shutdown(SHUT_WR)

    except KeyboardInterrupt:
        print('\n Shutting down....\n')
    except Exception as e:
        print('Error')
        print(e)

    serversocket.close()

print('Access http://localhost:9000')
createServer()
```

$\longrightarrow$ How many request can be held

$\swarrow$ The following code runs only when connection was established

```
Access http://localhost:9000
GET / HTTP/1.1
GET /favicon.ico HTTP/1.1

 Shutting down....
```



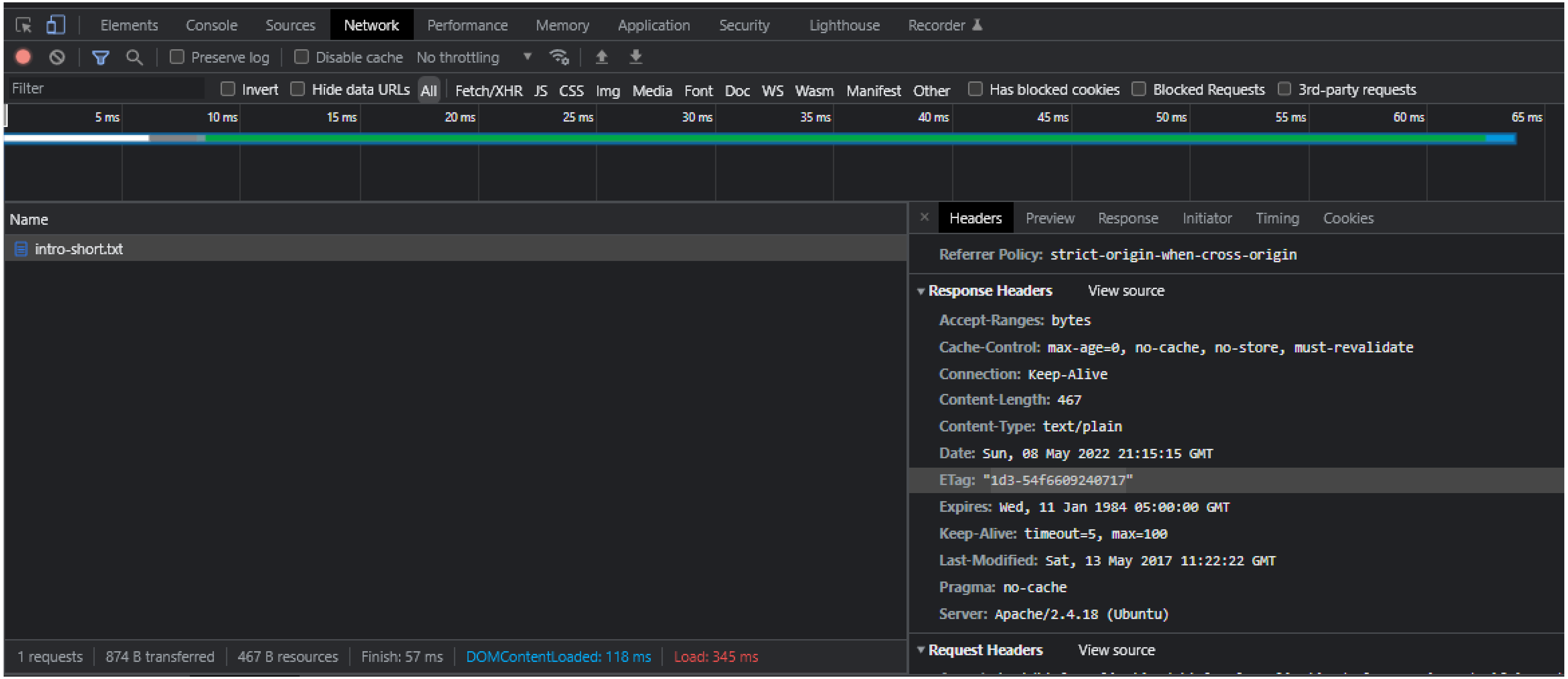
## An Even Simpler Web Client

```python
import urllib.request

fhand = urllib.request.urlopen('http://127.0.0.1:9000/romeo.txt')
for line in fhand:
    print(line.decode().strip())
```

```
$ python3 server.py
Access http://localhost:9000
GET http://127.0.0.1/romeo.txt HTTP/1.0
```

```
$ python3 client2.py
<html><body>Hello World</body></html>

$
```

# Check Header in Brower's Developer mode



intro-short.txt

Referrer Policy: strict-origin-when-cross-origin

▾ Response Headers    View source

```
Accept-Ranges: bytes
Cache-Control: max-age=0, no-cache, no-store, must-revalidate
Connection: Keep-Alive
Content-Length: 467
Content-Type: text/plain
Date: Sun, 08 May 2022 21:15:15 GMT
ETag: "1d3-54f6609240717"
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Keep-Alive: timeout=5, max=100
Last-Modified: Sat, 13 May 2017 11:22:22 GMT
Pragma: no-cache
Server: Apache/2.4.18 (Ubuntu)
```

▾ Request Headers    View source

1 requests | 874 B transferred | 467 B resources | Finish: 57 ms | DOMContentLoaded: 118 ms | Load: 345 ms