

# Table of Contents

This slide deck consists of slides used in 5 lecture videos in Week 1. Below is a list of shortcut hyperlinks for you to jump into specific sections.

- (page 2) [Week 1: Introduction to Dynamic Web Content](#)
- (page 7) [Week 1: Network Sockets and Connections](#)
- (page 12) [Week 1: HyperText Transfer Protocol](#)
- (page 21) [Week 1: Building a Simple Web Browser in Python](#)
- (page 25) [Week 1: Building a Simple HTTP Server in Python](#)

Charles Severance  
[www.dj4e.com](http://www.dj4e.com)

# Introduction to Dynamic Web Content

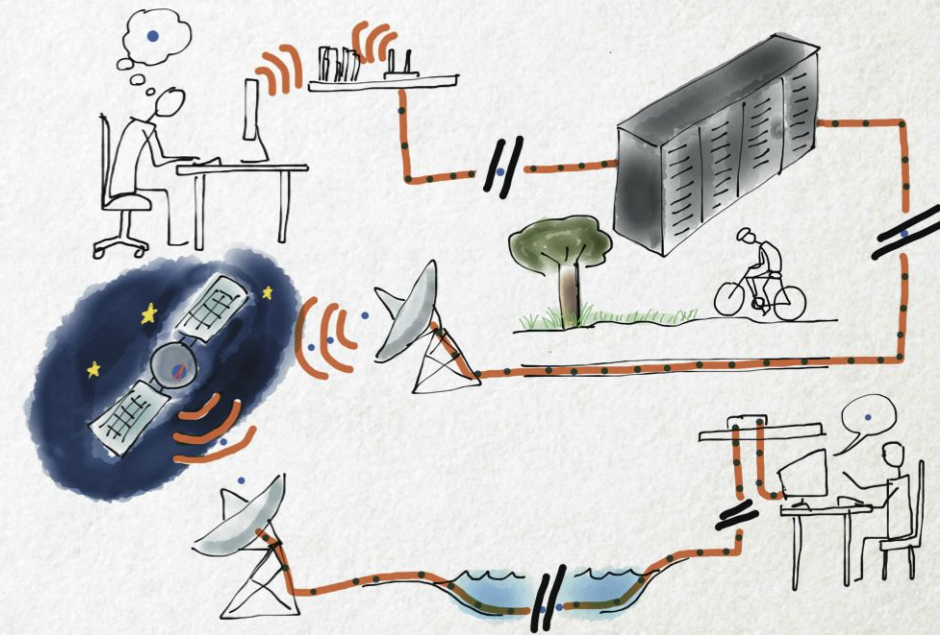


# A Free Book on Networking

If you find this topic area interesting and/or need more detail, please visit:

- **Open Educational Resources:**  
[www.net-intro.com](http://www.net-intro.com)
- **A Coursera course on this topic:**  
[www.coursera.org/learn/insidetheinternet](http://www.coursera.org/learn/insidetheinternet)

## Introduction to Networking HOW THE INTERNET WORKS



BY Charles R. Severance

ILLUSTRATIONS BY MAURO TOSELLI

# Web Application Technologies

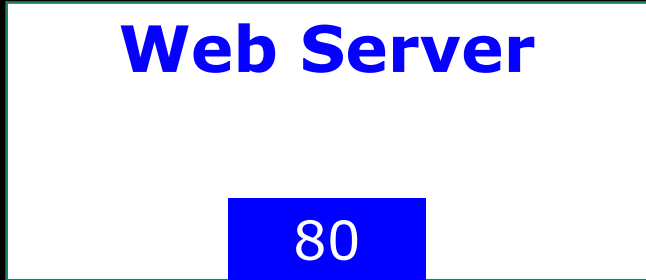


<http://data.pr4e.org/page1.htm>



# 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.



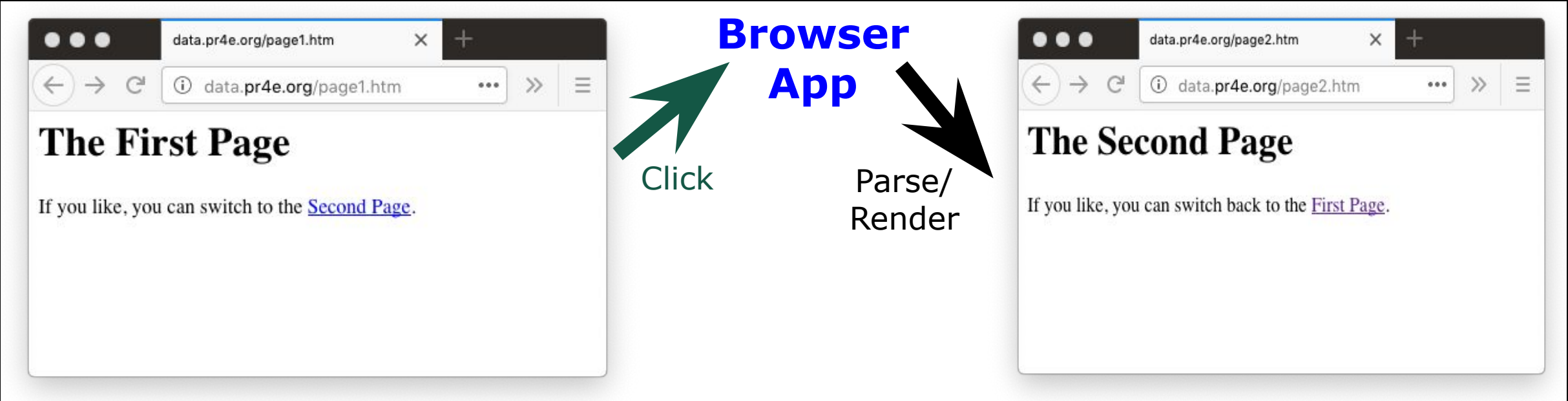
**Request**

**Response**

GET http://data.pr4e.org/page2.htm



```
<h1>The Second Page</h1>  
<p>If you like, you can switch back to the  
<a href="page1.htm">First Page</a>.</p>
```



# Network Sockets

Phone calls for pairs of applications

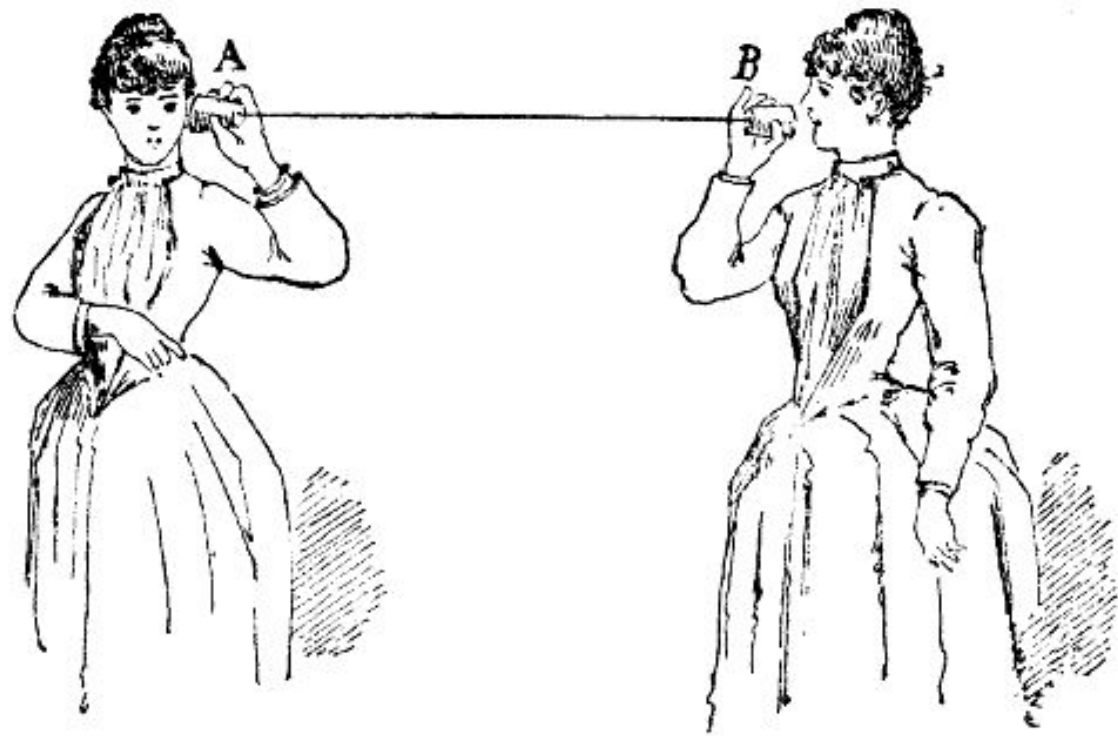


FIG. 76. Trådtelefon.



Image source:

[http://en.wikipedia.org/wiki/Tin\\_can\\_telephone](http://en.wikipedia.org/wiki/Tin_can_telephone)

<http://www.flickr.com/photos/kitcowan/2103850699/>



# 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](http://en.wikipedia.org/wiki/Internet_socket)

# 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](http://en.wikipedia.org/wiki/TCP_and_UDP_port)

www.dj4e.com

74.208.28.177

Incoming E-Mail

25

Login

23

Web Server

80

Web Server

443

Personal Mail Box

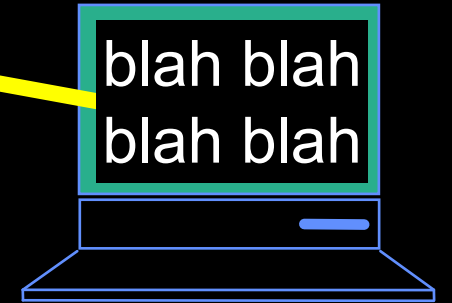
109

Personal Mail Box

110



blah blah  
blah blah



INFORMATICS



# HyperText Transfer Protocol

Wandering through linked documents on the Internet

# Uniform Resource Locator

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

protocol

host

document

# 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

# 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](http://www.ietf.org)
- Standards are called "RFCs" - "Request for Comments"

INTERNET PROTOCOL  
DARPA INTERNET PROGRAM  
PROTOCOL SPECIFICATION  
September 1981

The internet protocol treats each internet datagram as an independent entity unrelated to any other internet datagram. There are no connections or logical circuits (virtual or otherwise).

The internet protocol uses four key mechanisms in providing its service: Type of Service, Time to Live, Options, and Header Checksum.

Source: <http://tools.ietf.org/html/rfc791>

Network Working Group  
Request for Comments: 2616  
Obsoletes: 2068  
Category: Standards Track

R. Fielding  
UC Irvine  
J. Gettys  
Compaq/W3C  
J. Mogul  
Compaq  
H. Frystyk  
W3C/MIT  
L. Masinter  
Xerox  
P. Leach  
Microsoft  
T. Berners-Lee  
W3C/MIT  
June 1999

## Hypertext Transfer Protocol -- HTTP/1.1

### Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

### Abstract

The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information



## 5 Request

A request message from a client to a server includes, within the first line of that message, the method to be applied to the resource, the identifier of the resource, and the protocol version in use.

```
Request          = Request-Line           ; Section 5.1
                  * ( ( general-header     ; Section 4.5
                      | request-header     ; Section 5.3
                      | entity-header ) CRLF) ; Section 7.1
                  CRLF
                  [ message-body ]       ; Section 4.3
```

### 5.1 Request-Line

The Request-Line begins with a method token, followed by the Request-URI and the protocol version, and ending with CRLF. The elements are separated by SP characters. No CR or LF is allowed except in the final CRLF sequence.

```
Request-Line     = Method SP Request-URI SP HTTP-Version CRLF
```

# 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`

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.
```

Web Server



Browser

# Accurate Hacking in the Movies

- Matrix Reloaded
- Bourne Ultimatum
- Die Hard 4
- ...

<http://nmap.org/movies.html>

```
10 [nmap] hosts2.nc [nobile]
11 # nmap -v -ss -O 10.2.2.2
11
13 Starting nmap U. 2.54BETA25
13 Insufficient responses for TCP sequencing (3), OS detect
13 accurate
14 Interesting ports on 10.2.2.2:
44 (The 1539 ports scanned but not shown below are in state
51 Port      State      Service
51 22/tcp    open      ssh
58
68 No exact OS matches for host
68
24 Nmap run completed -- 1 IP address (1 host up) scanned
50 # sshnuke 10.2.2.2 -rootpw="210H0101"
Connecting to 10.2.2.2:ssh ... successful.
Re Attempting to exploit SSHv1 CRC32 ... successful.
IP Resetting root password to "210H0101".
System open: Access Level (9)
```



# Simple "Browser" in Python

# The World's Simplest Browser

```
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>

```
$ python3 socket1.py
HTTP/1.1 200 OK
Date: Sat, 19 Jan 2019 04:23:25 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>
```

```
import socket

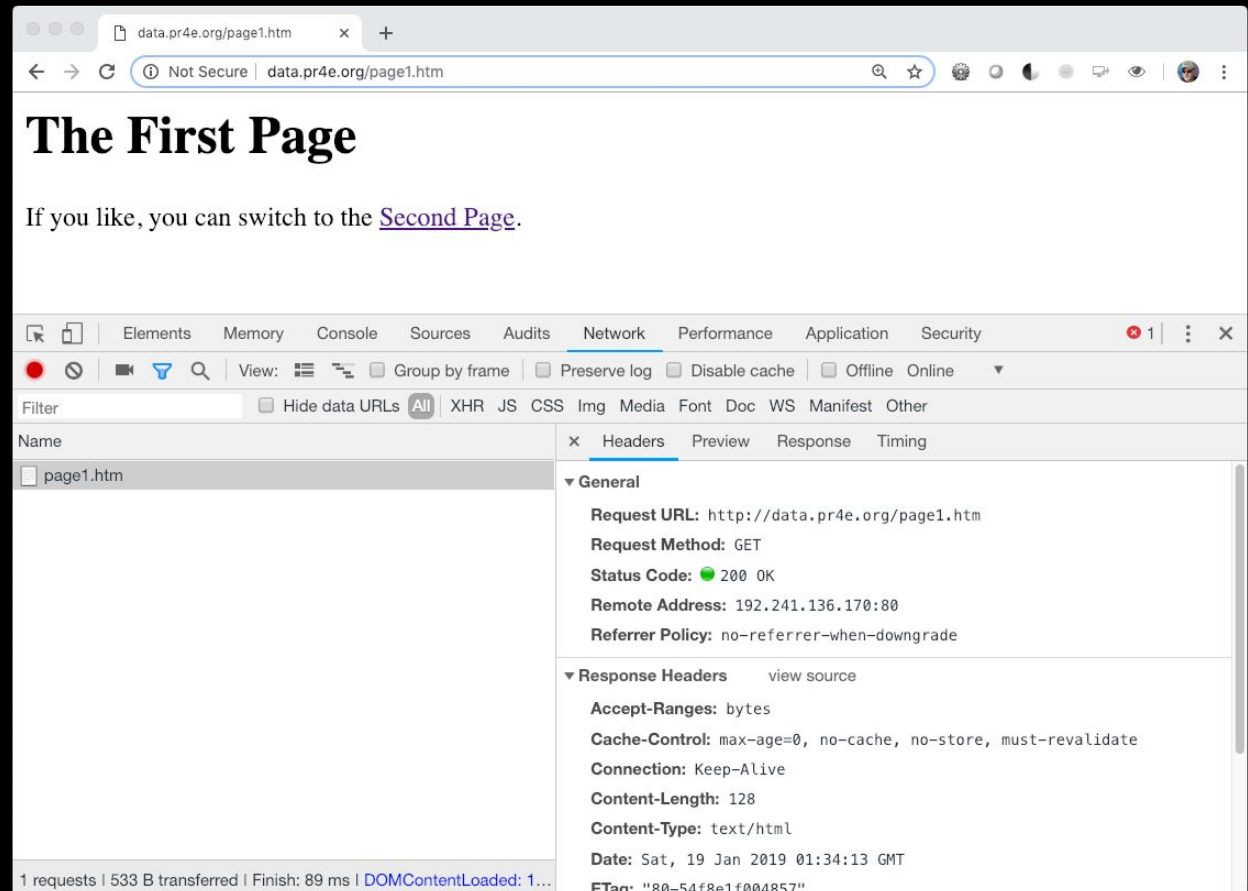
mysock = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
mysock.connect(('data.pr4e.org', 80))
cmd = 'GET
http://data.pr4e.org/page1.htm
HTTP/1.0\r\n\r\n'.encode()
mysock.send(cmd)

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

mysock.close()
```

# Viewing Headers – Browser Developer Mode

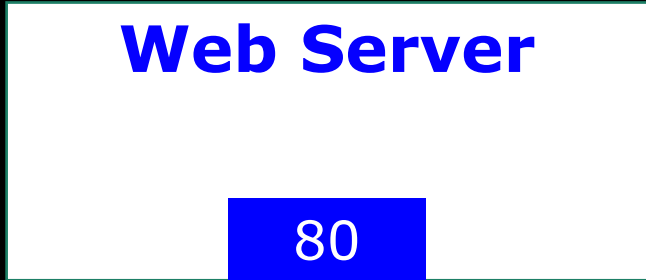
- Chrome: View > Developer
- FireFox: Tools -> Web Developer -> Toggle
- Safari: Preferences > Advanced > Show Develop Menu





# In the server

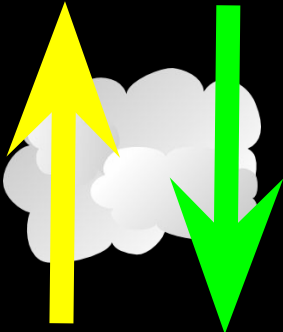
... the mighty server



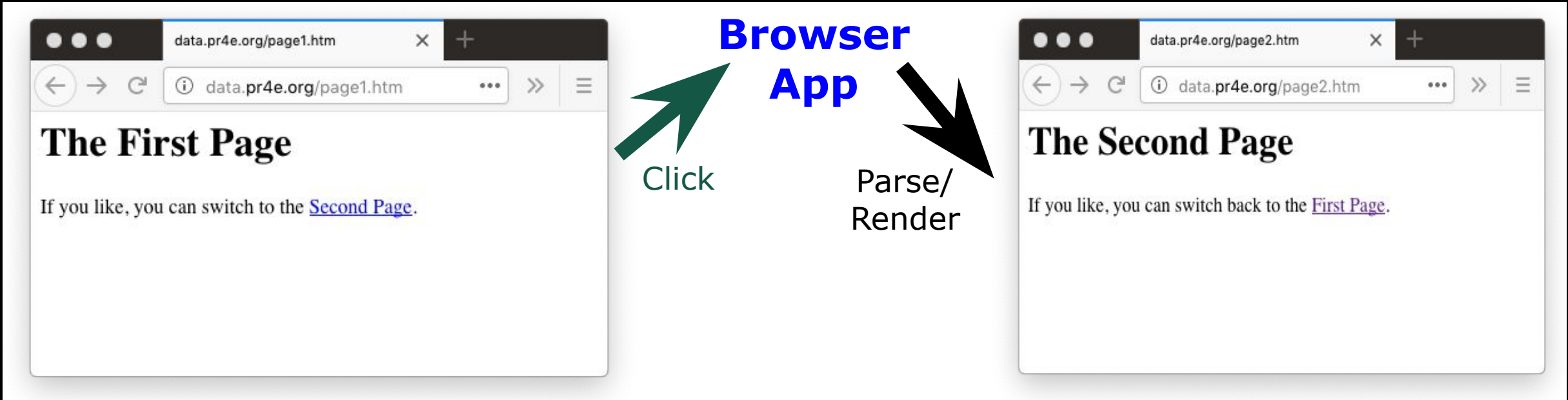
**Request**

**Response**

GET http://data.pr4e.org/page2.htm



```
<h1>The Second Page</h1>  
<p>If you like, you can switch back to the  
<a href="page1.htm">First Page</a>.</p>
```



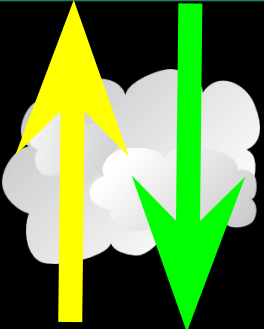
Web Server

??????

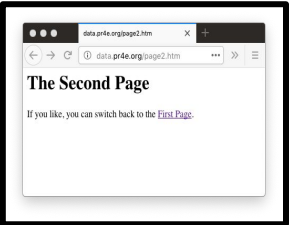
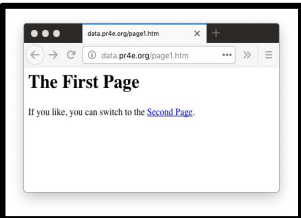
80

Request

Response



Browser App



```
<h1>The Second Page</h1><p>If you like, you can switch back to the <a href="page1.htm">First Page</a>.</p>
```

GET http://data.pr4e.org/page2.htm

# The World's Simplest Web Server

```
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])

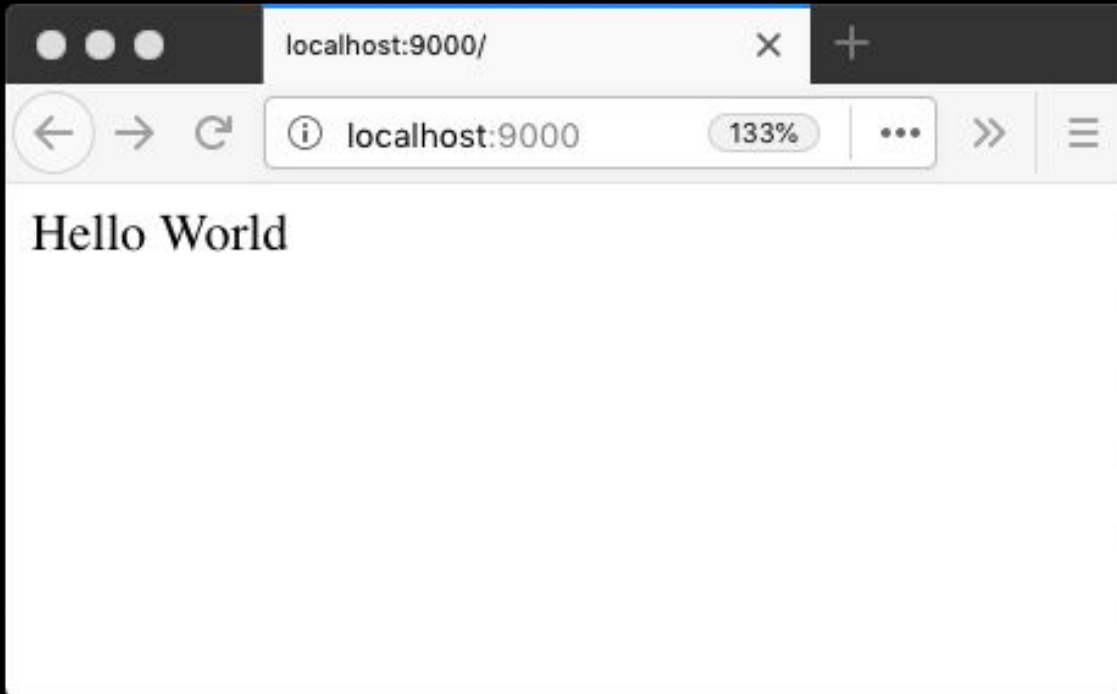
            data = "HTTP/1.1 200 OK\r\n"
            data += "Content-Type: text/html; charset=utf-8\r\n"
            data += "\r\n"
            data += "<html><body>Hello World</body></html>\r\n\r\n"
            clientsocket.sendall(data.encode())
            clientsocket.shutdown(SHUT_WR)

    except KeyboardInterrupt :
        print("\nShutting down...\n");
    except Exception as exc :
        print("Error:\n");
        print(exc)

    serversocket.close()

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

# Browser / Server Communication



```
$ pwd
dj4e/code/http
$ python3 server.py
Access http://localhost:9000
GET / HTTP/1.1
GET /favicon.ico HTTP/1.1
```

<https://www.dj4e.com/code/http/server.py>

# A Very Simple Web Client

```
import socket

mysock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
mysock.connect(('127.0.0.1', 9000))
cmd = 'GET http://127.0.0.1/romeo.txt 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/client1.py>

# Client / Server Communication

```
$ pwd
dj4e/code/http
$ python3 server.py
Access http://localhost:9000
GET http://127.0.0.1/romeo.txt HTTP/1.0
```

```
$ python3 client1.py
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8

<html><body>Hello World</body></html>

$
```

# An Even Simpler Web Client

```
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>

$
```

<https://www.dj4e.com/code/http/client2.py>



# Browser / Django Communication

```
0587357624:mytestsite csev$ python3 manage.py runserver
Performing system checks...
```

```
System check identified no issues (0 silenced).
```

```
September 03, 2019 - 13:28:13
```

```
Django version 2.1.7, using settings 'mytestsite.settings'
```

```
Starting development server at http://127.0.0.1:8000/
```

```
Quit the server with CONTROL-C.
```

```
[03/Sep/2019 13:28:25] "GET / HTTP/1.1" 200 16348
```

```
[03/Sep/2019 13:28:25] "GET /static/admin/css/fonts.css HTTP/1.1" 200 423
```

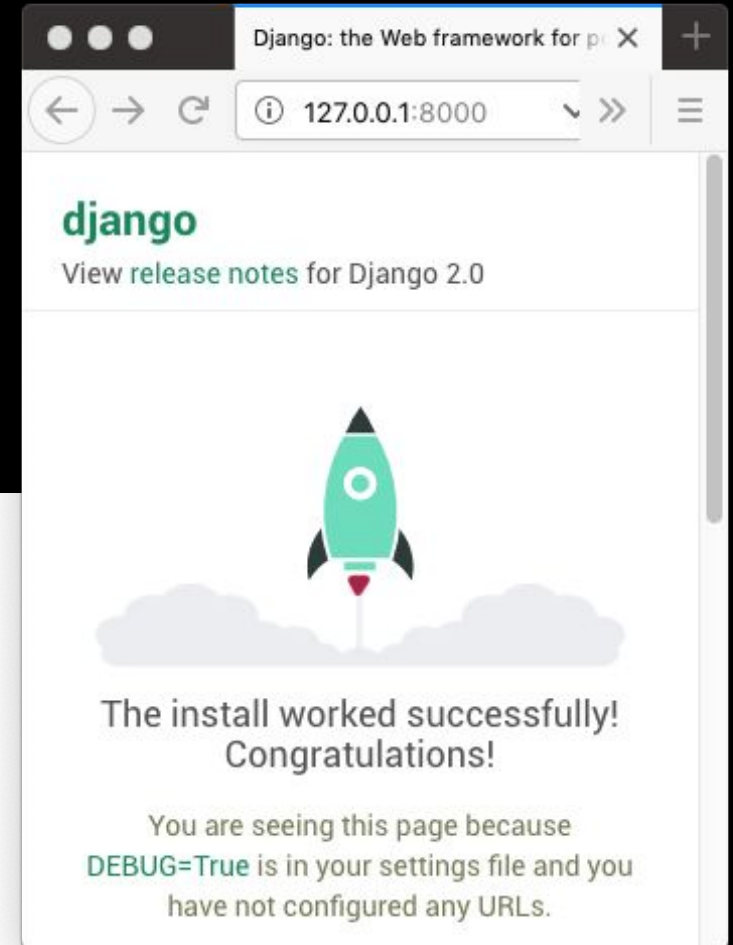
```
Not Found: /favicon.ico
```

```
[03/Sep/2019 13:28:25] "GET /favicon.ico HTTP/1.1" 404 1976
```

```
[03/Sep/2019 13:28:25] "GET /static/admin/fonts/Roboto-Regular-webfont.woff HTTP/1.1" 200 80304
```

```
[03/Sep/2019 13:28:25] "GET /static/admin/fonts/Roboto-Bold-webfont.woff HTTP/1.1" 200 82564
```

```
[03/Sep/2019 13:28:25] "GET /static/admin/fonts/Roboto-Light-webfont.woff HTTP/1.1" 200 81348
```



# Acknowledgements / Contributions

These slides are Copyright 2019- Charles R. Severance (www.dr-chuck.com) as part of www.dj4e.com and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Insert new Contributors and Translators here including names and dates

Continue new Contributors and Translators here