CSCI-1680
HTTP

Nick DeMarinis

Based partly on lecture notes by Rodrigo Fonseca, Scott Shenker and John Jannotti

Administrivia

- You should have done your milestone II meeting

- You have one week from today to finish TCP.
  Do not wait until the end.

- Final project info: Thursday

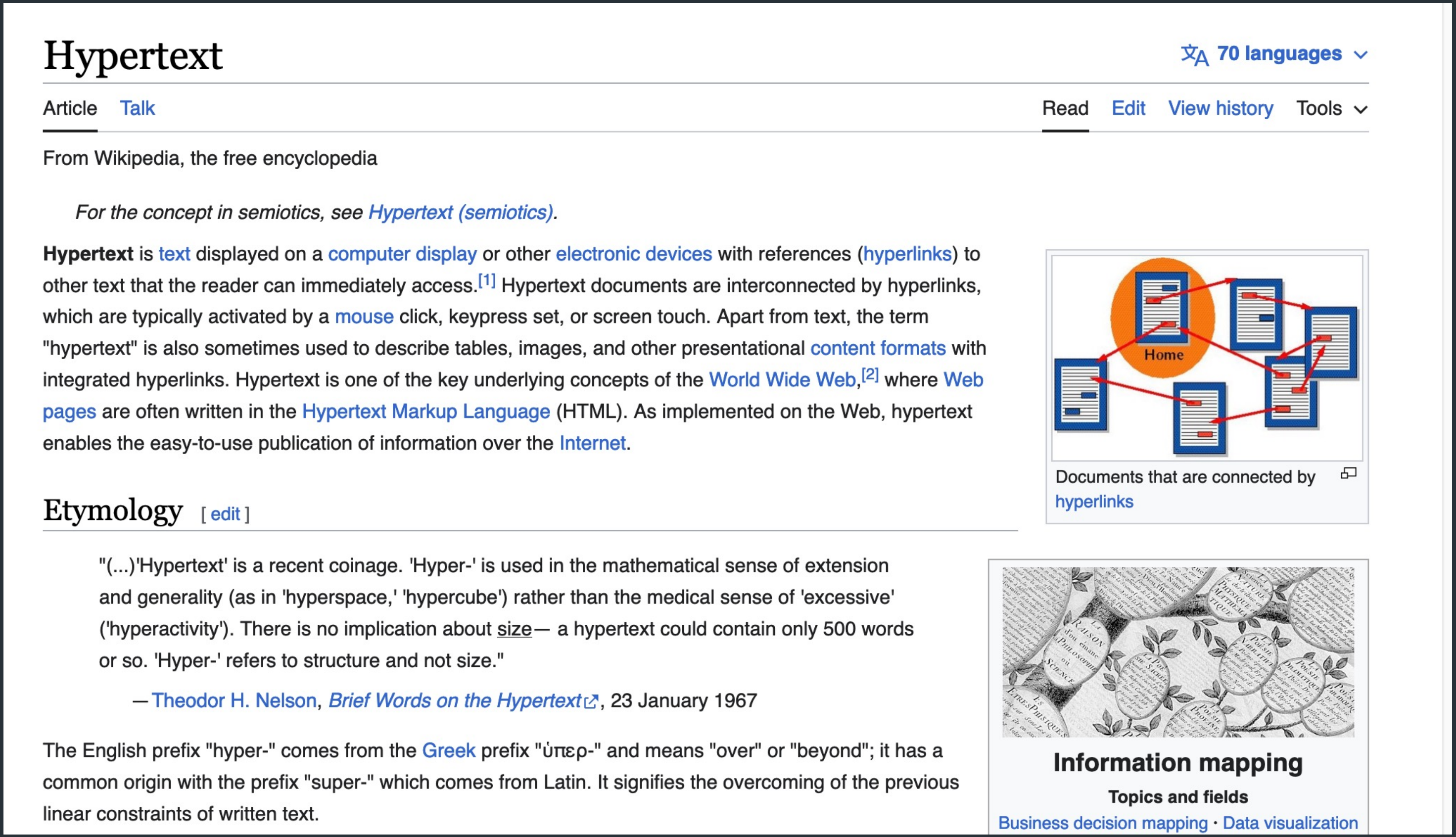
HTTP: Hypertext Transfer Protocol

HTTP

“Application protocol for distributed, collaborative hypermedia information systems”

- Fundamental protocol behind “the web”
- Now part of most things we do on the Internet—so much more than web pages

But what is hypertext?



Hypertext

From Wikipedia, the free encyclopedia

For the concept in semiotics, see Hypertext (semiotics).

Hypertext is text displayed on a computer display or other electronic devices with references (hyperlinks) to other text that the reader can immediately access.[1] Hypertext documents are interconnected by hyperlinks, which are typically activated by a mouse click, keypress set, or screen touch. Apart from text, the term "hypertext" is also sometimes used to describe tables, images, and other presentational content formats with integrated hyperlinks. Hypertext is one of the key underlying concepts of the World Wide Web,[2] where Web pages are often written in the Hypertext Markup Language (HTML). As implemented on the Web, hypertext enables the easy-to-use publication of information over the Internet.

Documents that are connected by hyperlinks

Etymology [edit]

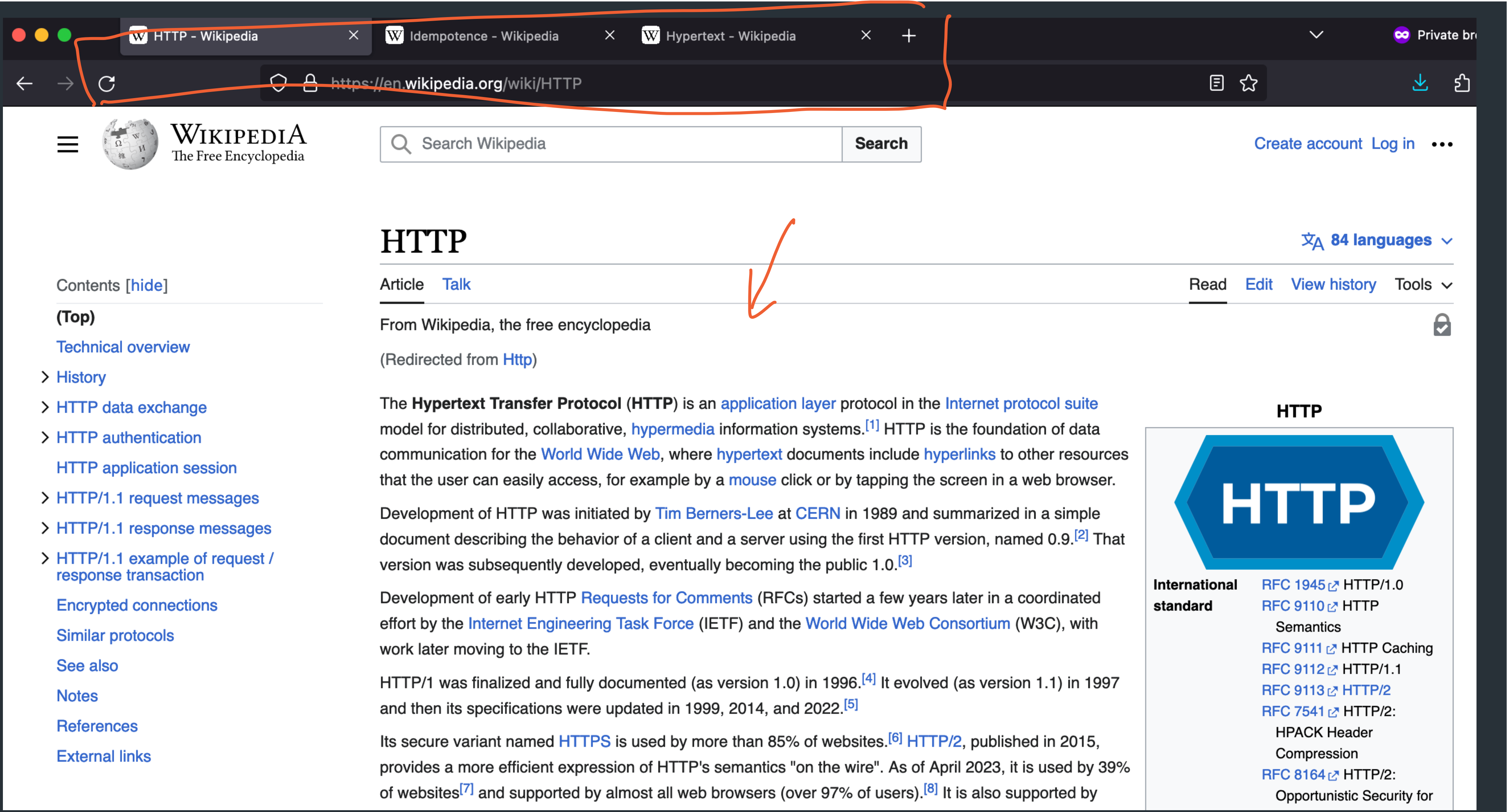
> "(...)'Hypertext' is a recent coinage. 'Hyper-' is used in the mathematical sense of extension and generality (as in 'hyperspace,' 'hypercube') rather than the medical sense of 'excessive' ('hyperactivity'). There is no implication about size— a hypertext could contain only 500 words or so. 'Hyper-' refers to structure and not size."
>
> —Theodor H. Nelson, *Brief Words on the Hypertext*, 23 January 1967

The English prefix "hyper-" comes from the Greek prefix "ὑπερ-" and means "over" or "beyond"; it has a common origin with the prefix "super-" which comes from Latin. It signifies the overcoming of the previous linear constraints of written text.

Information mapping

Topics and fields

Business decision mapping · Data visualization



HTTP

From Wikipedia, the free encyclopedia

(Redirected from Http)

The **Hypertext Transfer Protocol** (**HTTP**) is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web, where hypertext documents include hyperlinks to other resources that the user can easily access, for example by a mouse click or by tapping the screen in a web browser.

Development of HTTP was initiated by Tim Berners-Lee at CERN in 1989 and summarized in a simple document describing the behavior of a client and a server using the first HTTP version, named 0.9.[2] That version was subsequently developed, eventually becoming the public 1.0.[3]

Development of early HTTP Requests for Comments (RFCs) started a few years later in a coordinated effort by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C), with work later moving to the IETF.

HTTP/1 was finalized and fully documented (as version 1.0) in 1996.[4] It evolved (as version 1.1) in 1997 and then its specifications were updated in 1999, 2014, and 2022.[5]

Its secure variant named HTTPS is used by more than 85% of websites.[6] HTTP/2, published in 2015, provides a more efficient expression of HTTP's semantics "on the wire". As of April 2023, it is used by 39% of websites[7] and supported by almost all web browsers (over 97% of users).[8] It is also supported by

HTTP

International standard	RFC 1945 HTTP/1.0 RFC 9110 HTTP Semantics RFC 9111 HTTP Caching RFC 9112 HTTP/1.1 RFC 9113 HTTP/2 RFC 7541 HTTP/2: HPACK Header Compression RFC 8164 HTTP/2: Opportunistic Security for

HTTP: a protocol for distributing hypertext media
(*and now so much more)

OF PAGES/HYPER MEDIA.

Enables the World Wide Web (WWW): a distributed database of pages linked through HTTP

*HTTP: a protocol for distributing hypertext media (*and now so much more)*

Enables the World Wide Web (WWW): a distributed database of pages linked through HTTP

… now synonymous with with "The Internet" itself!

Tim Berners-Lee

History

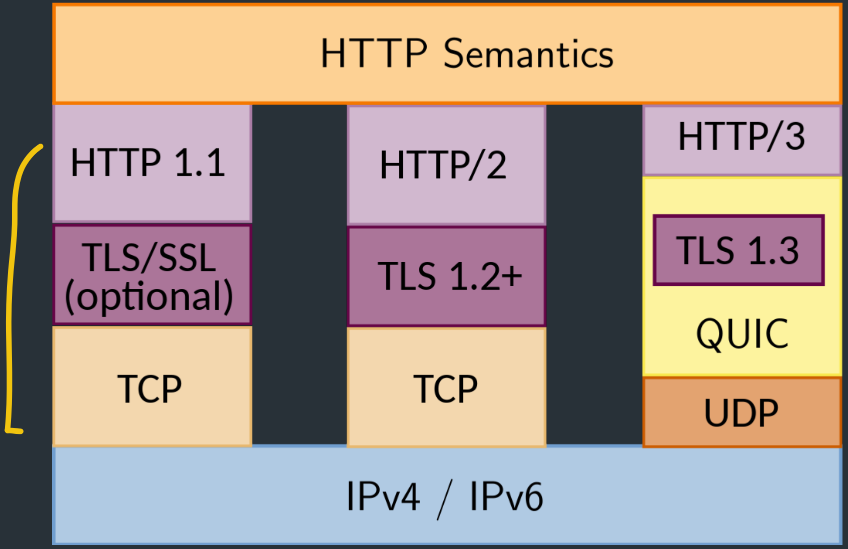
- 1990: First HTTP implementation
  - Tim Berners-Lee, CERN
- 1991: HTTP/0.9: Fetching pages
- 1992: HTTP/1.0:
  Client/server information, simple caching
- 1996: HTTP/1.1
  - Extensive caching support
  - Host identification
  - Pipelined, persistent connections, …

STILL MOST WIDELY SUPPORTED!

The first webserver

- 2015: HTTP/2
  - Main goal: reduce latency

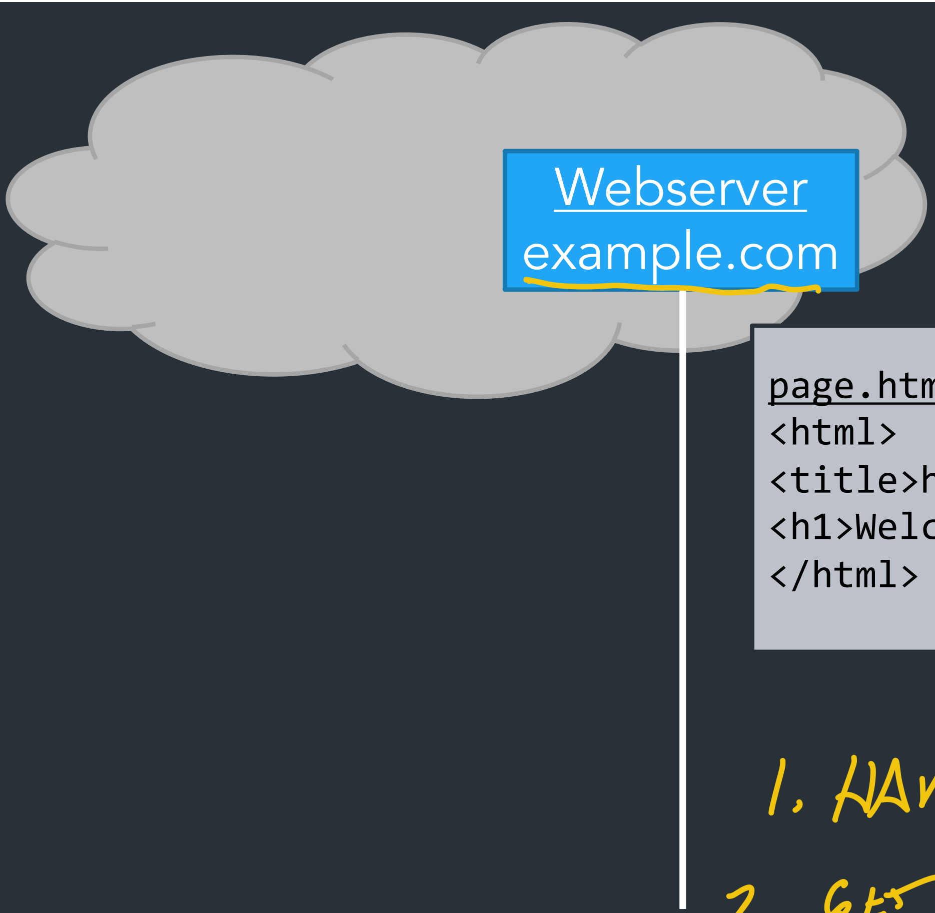
- 2022: HTTP/3
  - Still: reduce latency
  - Integrates security via TLS
  - Replace transport layer with QUIC
  - Already supported in >94% of browsers



http://httpwg.org/specs/rfc7540.html

– MORE ON THIS LATER!

How does “the web” work?

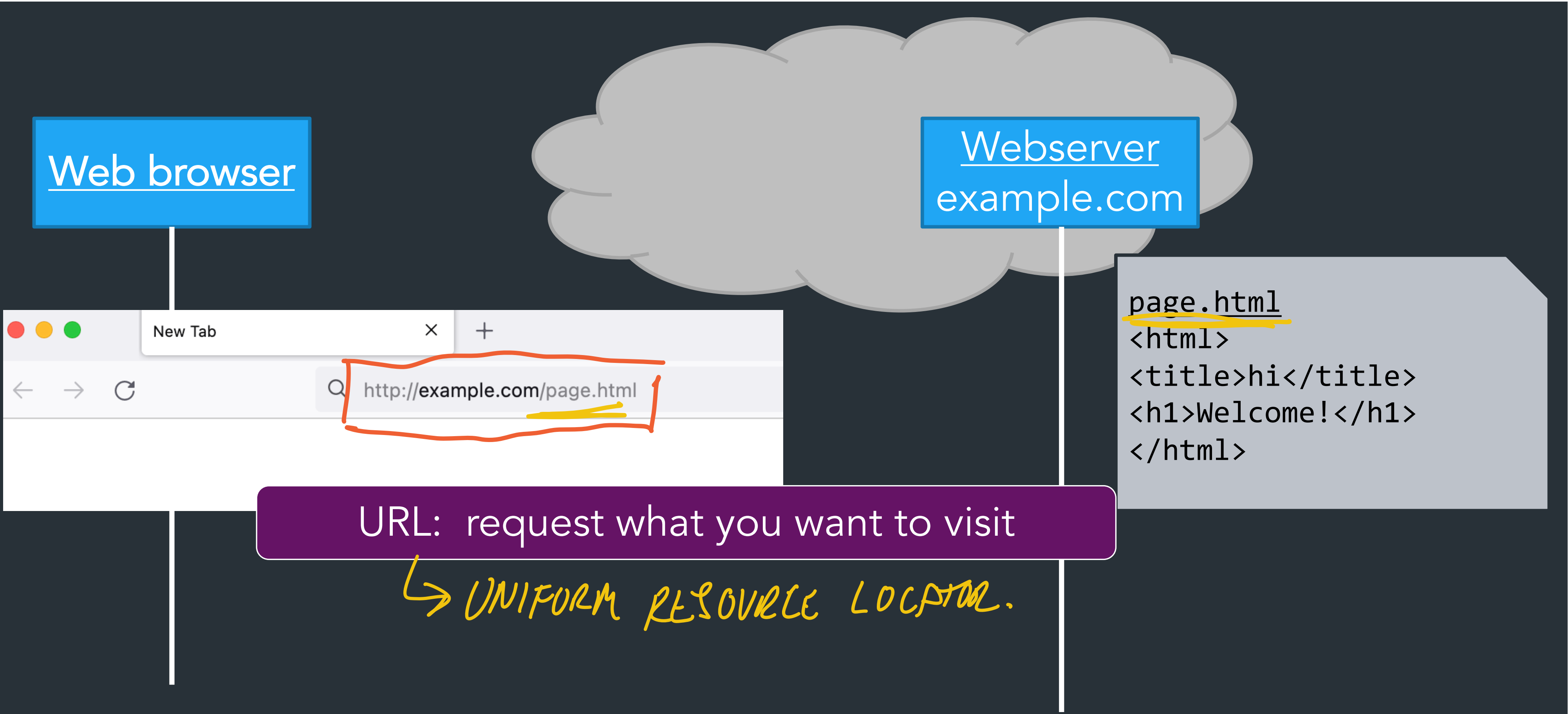


Webserver
example.com

```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```

1. HAVE SERVER

2. GET DOMAIN.

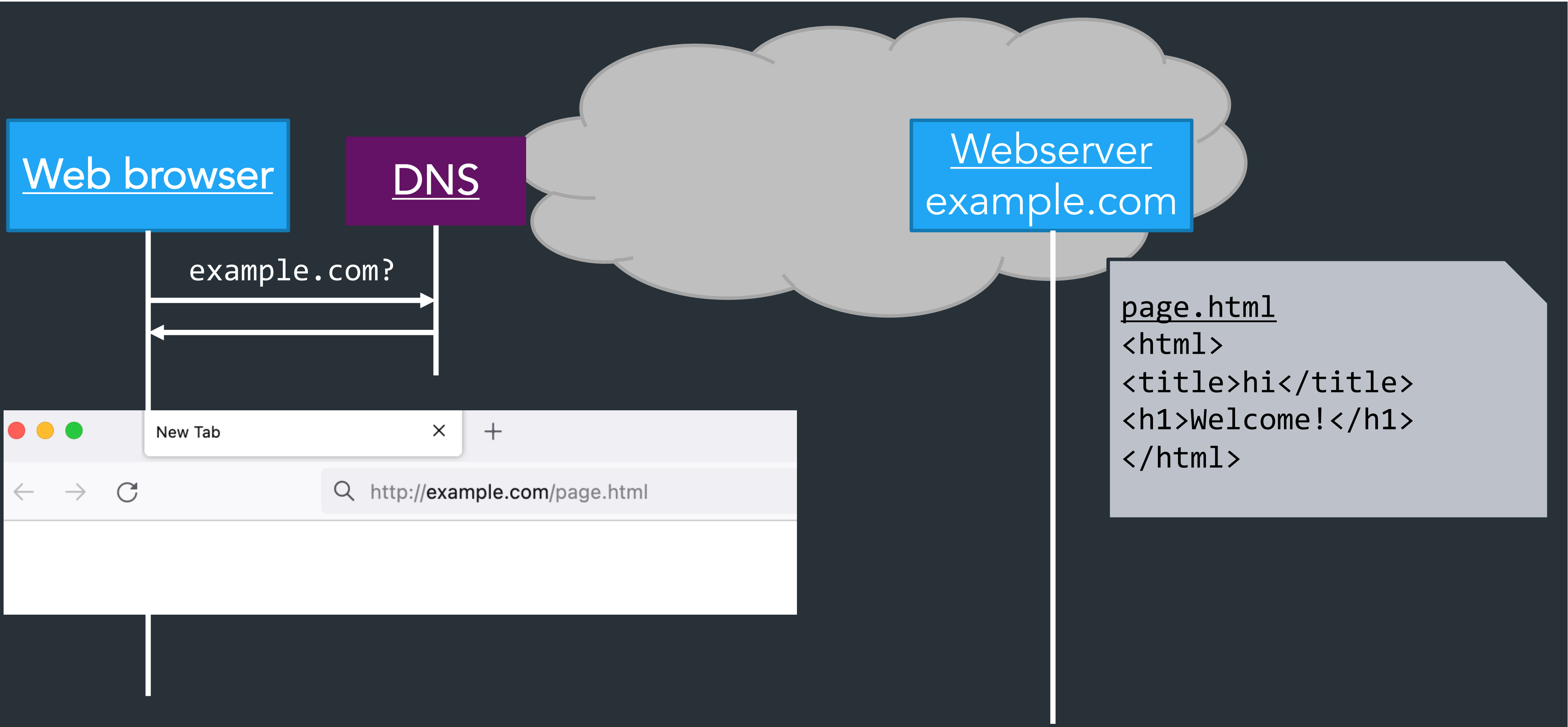


```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```

URL: request what you want to visit

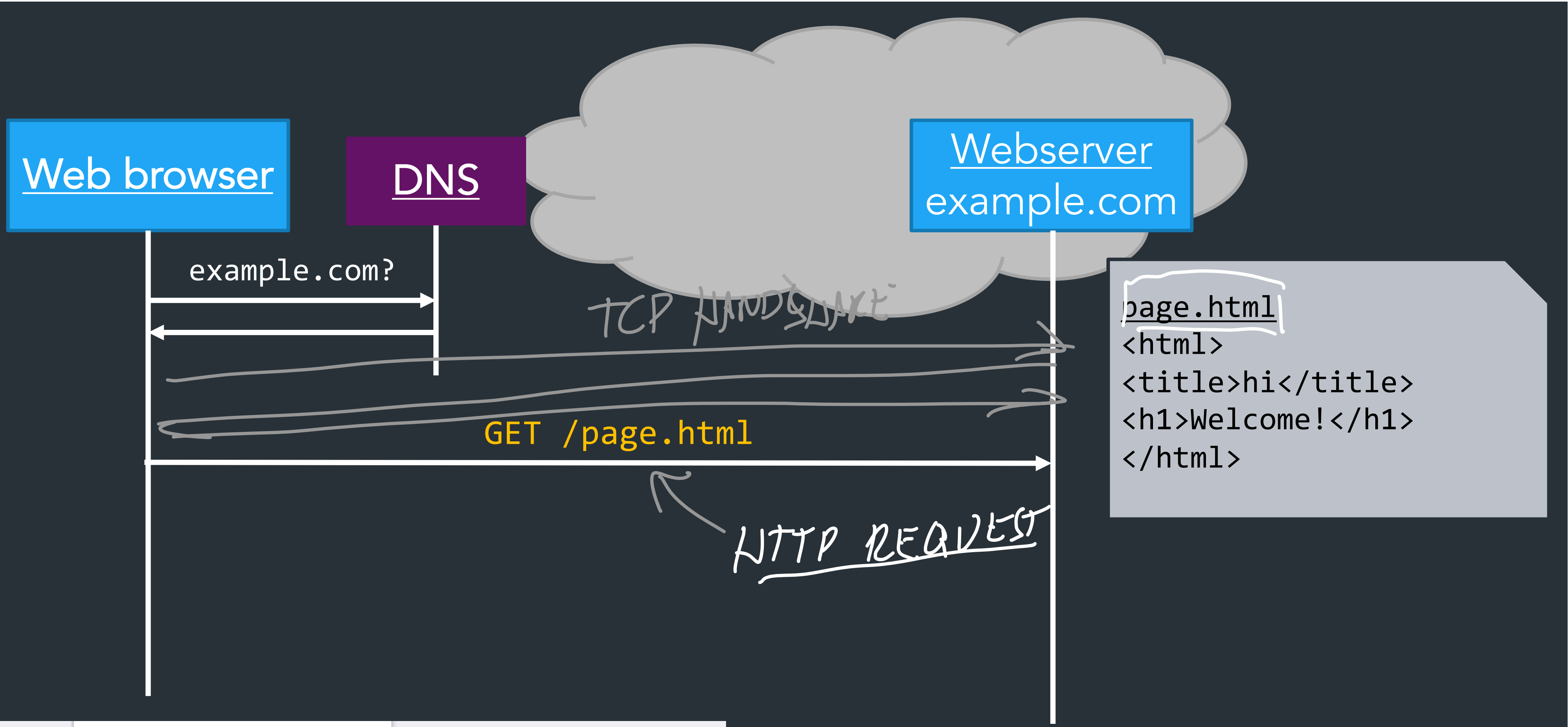
↳ UNIFORM RESOURCE LOCATOR.

⇒ REFERS TO SPECIFIC RESOURCES ON A SERVICE/SITE.

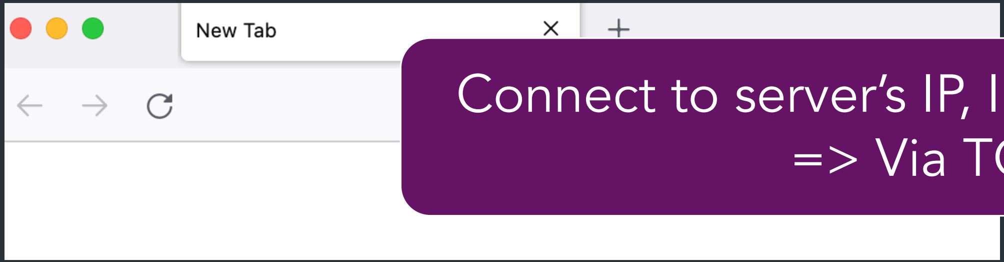


```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```

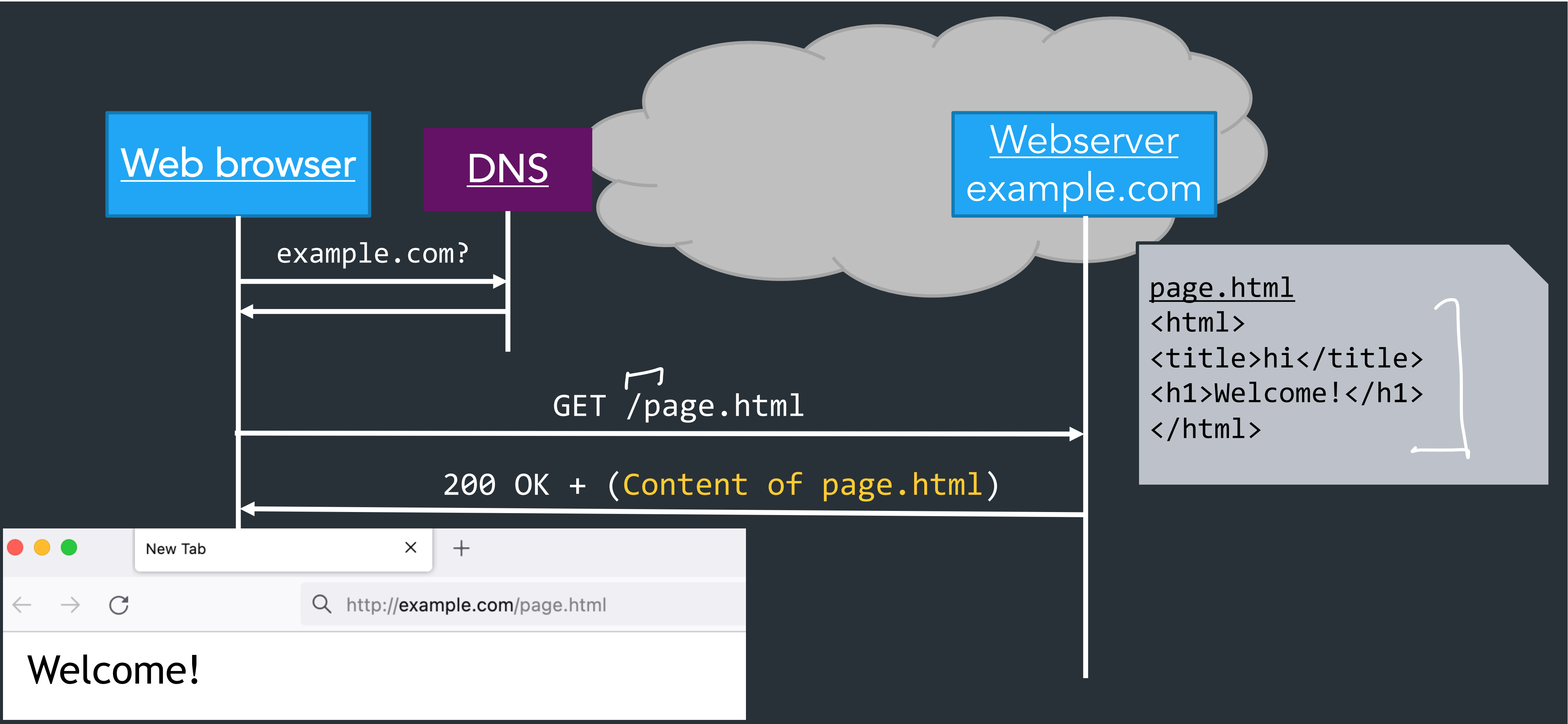
First step: look up where server is on Internet
=> Usually, via DNS



```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```



Connect to server’s IP, Initiate a request for page
=> Via TCP (usually)



```
page.html
<html>
<title>hi</title>
<h1>Welcome!</h1>
</html>
```

Server returns response (in this case, with HTML)

Why so successful?

Anyone can host a website!

… just need a domain and a server

HOSTING

Clients can easily find arbitrary pages, pages can easily link to others => content can grow very quickly

HTTP components

Content: objects (HTML, images, JSON, …)

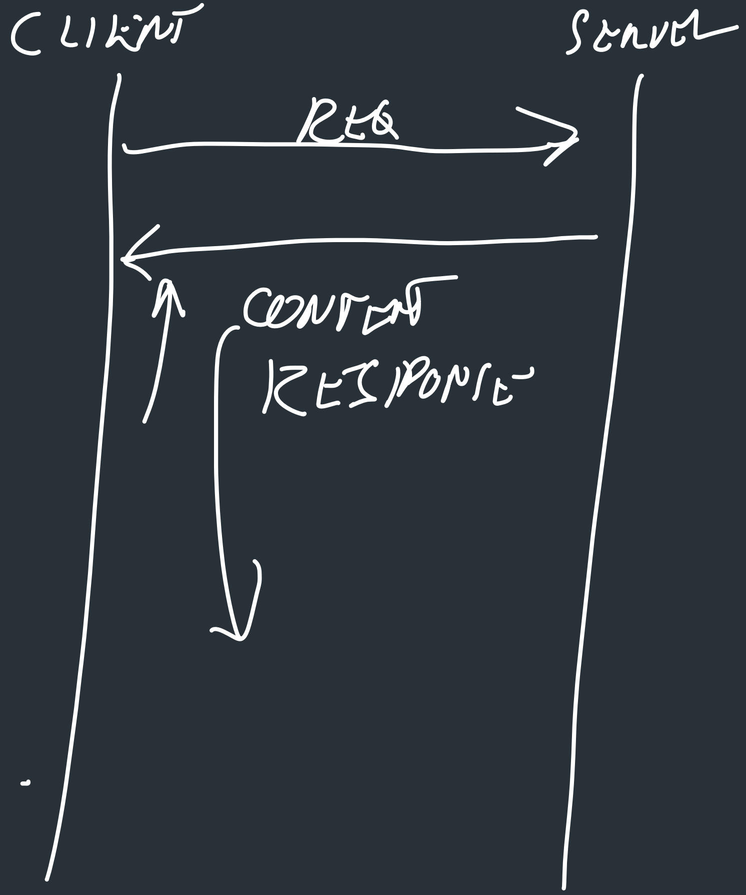
Clients: send requests, receive response

Servers: store content, or generate it

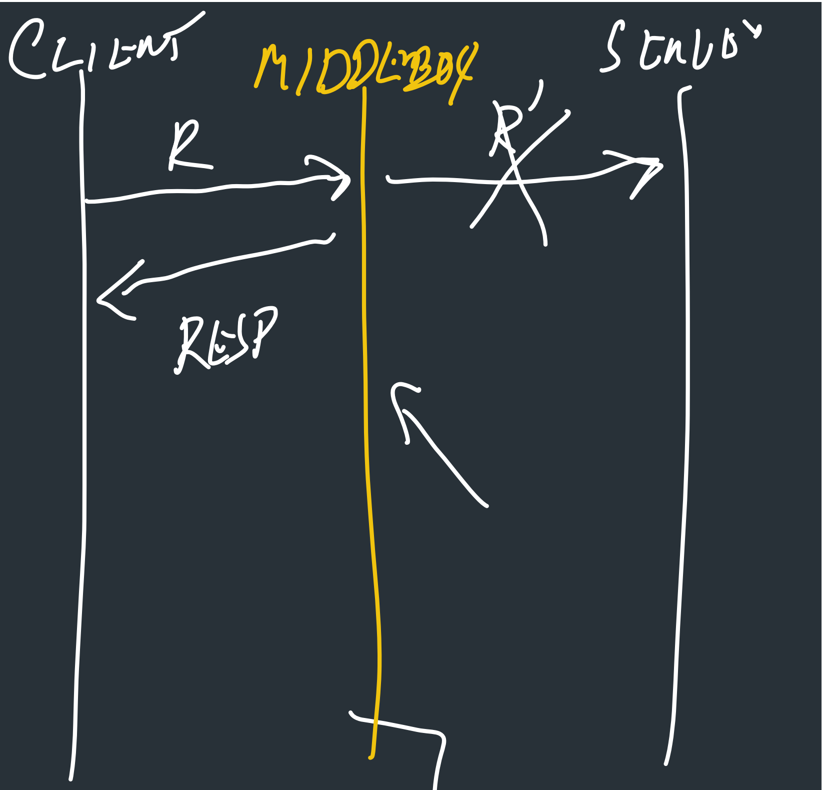
TWO TYPES OF CONTENT

STATIC: SINGLE FILE

DYNAMIC: CONTENT GENERATED, PER-REQUEST ON THE FLY.



HTTP components



Content: objects (HTML, images, JSON, …)

Clients: send requests, receive response

Servers: store content, or generate it

Proxies/Middleboxes

- Placed between clients and servers
- Do extra stuff: caching, anonymization, logging, transcoding, filtering access

=> Important for scaling, modern browsing…
more on this later

How to find stuff?

ABSTRACTION NAME → HOST

- So far: DNS: names for one or more hosts
  - eg. cs.brown.edu

How do we ask for a specific *resource* from this host?

NAME → RESOURCE.

URL: Uniform Resource Locator

URLs: how we find stuff

DOMAIN. (OS USES TO FIND SERVER)

RESOURCE ON PAGE.

https://cs.brown.edu/courses/csci1680/f23/policies/#late-policy

↳ DIRECTORY (MIGHT BE A FILE PATH, MIGHT NOT)
=> SERVER USES TO FIND PAGE

PROTOCOL (HTTP, SSH, ZOOM, ...)
=> CLIENT USES TO FIND APP TO RUN

How to find stuff: URLs

protocol://[name@]hostname[:port]/directory/resource?k1=v1&k2=v2#tag

MORE PARAMETERS FOR SERVER (CUSTOM FORMAT USED BY SERVER)

- Name: can identify a client
- *Hostname:* FQDN or IP address
- *Port number: defaults to common protocol port (eg. 80, 22)*
- *Directory:* path to the resource
- *Resource:* name of the object
- After that, various delimiters to specify further, common examples:
  - ?*parameters* are passed to the server for execution
  - #*tag* allows jumps to named tags within document

HTTP: the protocol

- Client-server protocol
- Protocol (but not data) in ASCII (before HTTP/2)
- Stateless
- Server typically listens on port 80 (or 443, with TLS)

- Server sends response, may close connection (client may ask it to say open)

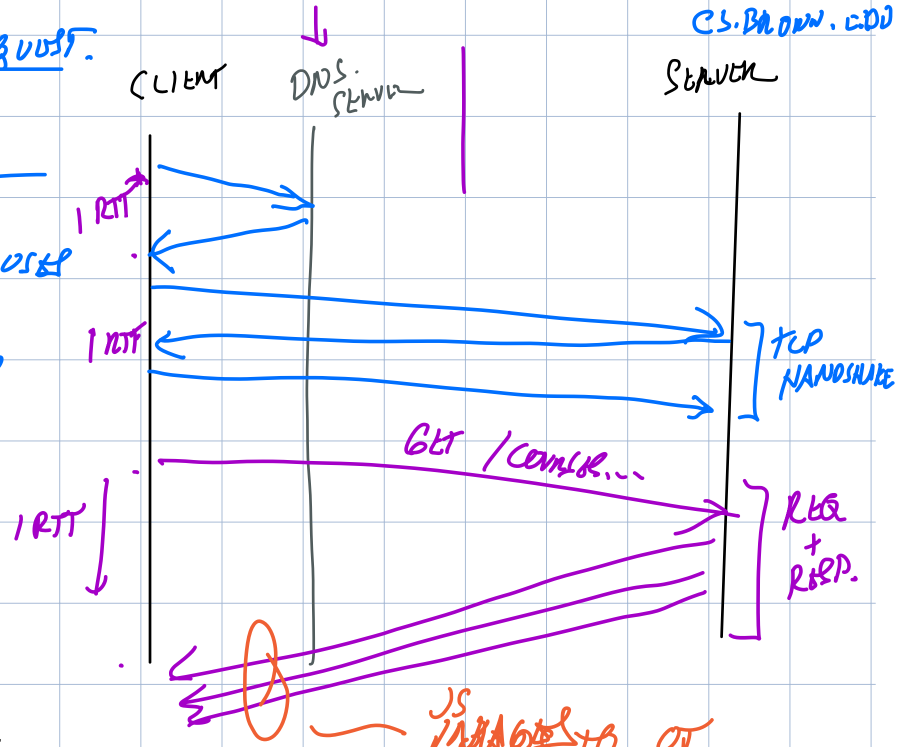
Steps in HTTP$^{(1.0)}$ Request

- Open TCP connection to server
- Send request
- Receive response
- TCP connection terminates
  - How many RTTs for a single request?
- You may also need to do a DNS lookup first!

STEPS TO A REQUEST.

1, DNS REQUEST

BACKGROUND:
- HTTP TYPICALLY USES TCP, PORT 80.
HTTPS = 443



Pieces of a request
- Method: GET, POST, PUT, …
=> What operation you are trying to do
- GET tries to fetch, POST/POST write to server state somehow

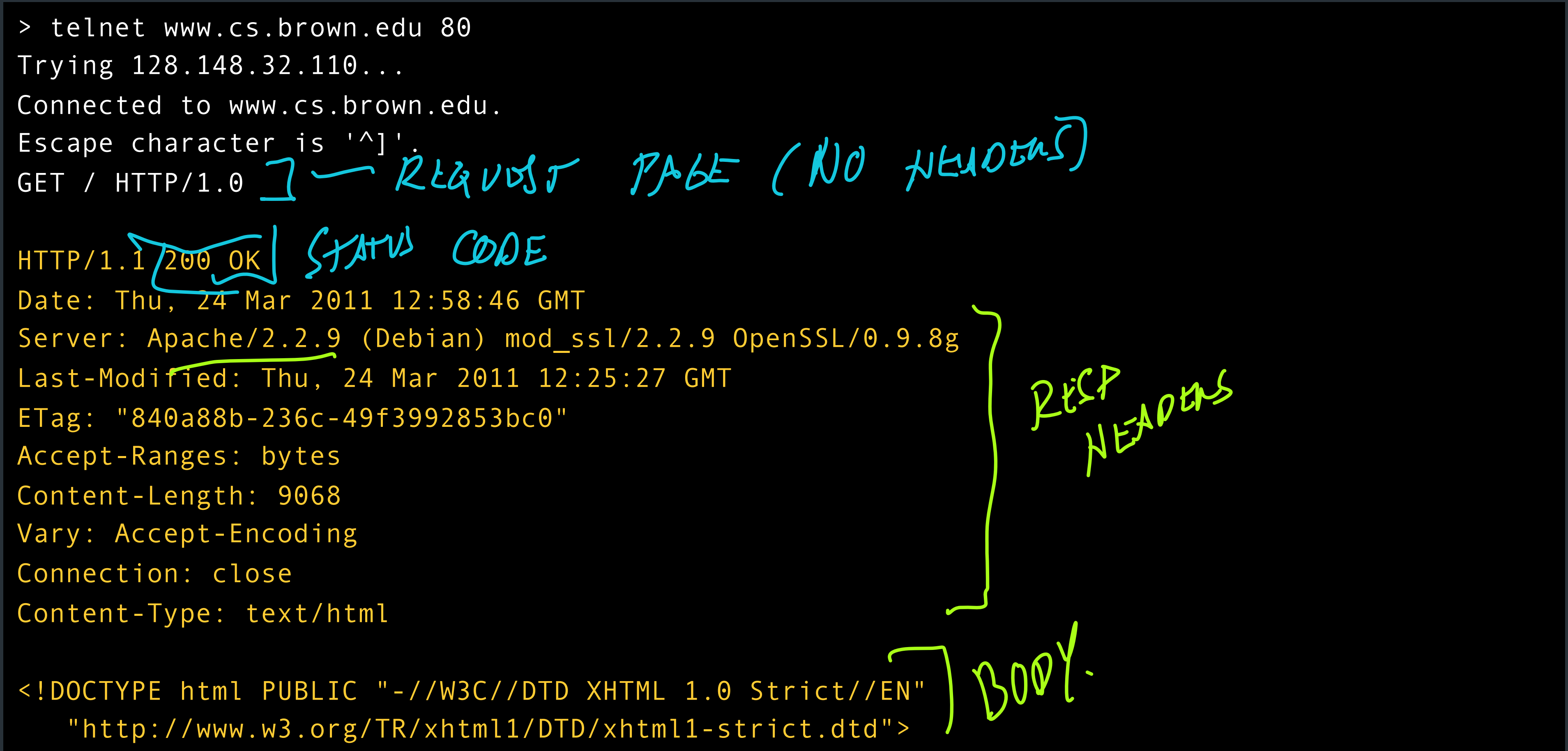
- Headers: metadata about request or client
- User-Agent: info about browser, app doing the request
- Language, content-type, cookies (user info)

- (Sometimes) Body: PUT, POST (+others) have content client sends to server (eg. Uploaded files, form responses, etc.)

URL may have extra parameters, which are interpreted by the server

What goes in an HTTP response?

- Status code (200 OK, 404 not found, 403 forbidden, 500 server error)
- HTTP Headers information (metadata about what the response looks like)
    - Content-Type: text/html
- Response body (HTML, image, JSON, …)



```
> telnet www.cs.brown.edu 80
Trying 128.148.32.110...
Connected to www.cs.brown.edu.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 24 Mar 2011 12:58:46 GMT
Server: Apache/2.2.9 (Debian) mod_ssl/2.2.9 OpenSSL/0.9.8g
Last-Modified: Thu, 24 Mar 2011 12:25:27 GMT
ETag: "840a88b-236c-49f3992853bc0"
Accept-Ranges: bytes
Content-Length: 9068
Vary: Accept-Encoding
Connection: close
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

REQUEST PAGE (NO HEADERS)

STATUS CODE

RESP HEADERS

BODY.

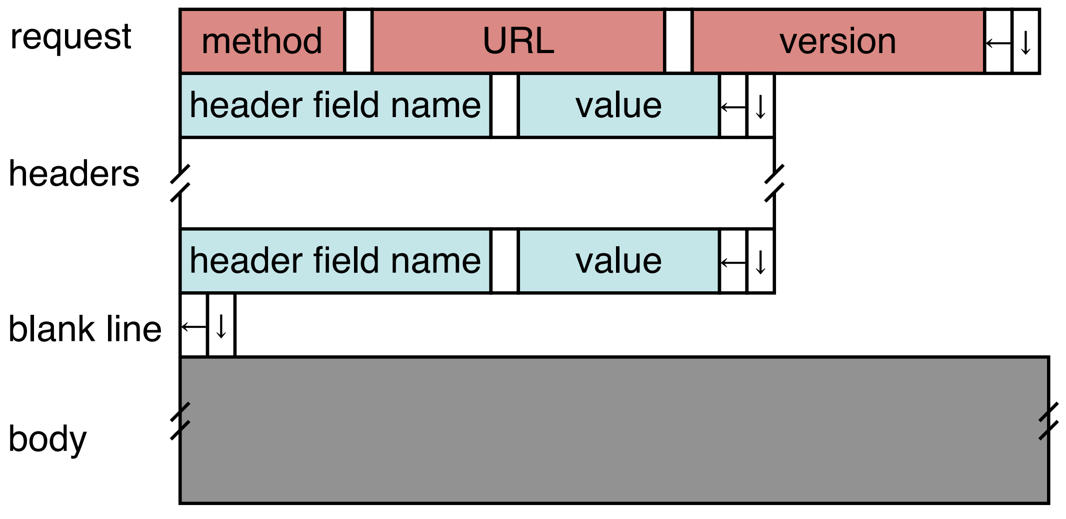
HTTP Request

Method:

- GET: current value of resource, run program
- POST: update a resource, provide input for a program. . .

Headers: useful info about request

- E.g., desired language, text encoding



Sample Browser Request

```
GET / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macinto ...
Accept: text/xml,application/xm ...
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
(empty line)
```

In your browser: right click => Inspect element => Network

HTTP is Stateless

- Each request/response treated independently
- Servers not required to maintain state

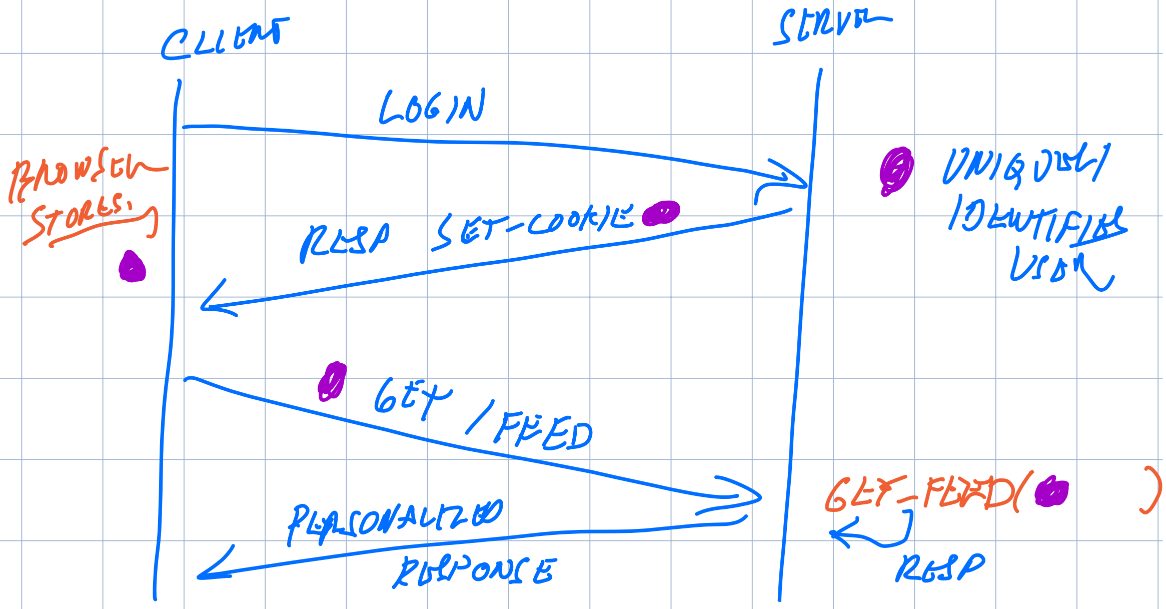
But…

- – Most applications need persistent state
- – E.g., shopping cart, web-mail, usage tracking, (most sites today!)

HTTP is a STATELESS protocol

- Don’t want server to need to remember information about the client
  => When server receives a response, not guaranteed that the server knows about the client ahead of time (not required to have any server side state)

- Applications may give state to client based on header info— specifically data called <u>cookies</u>
  => Cookie: some piece of information (usually an ID number) that tells the server how to look up state for the client
  => One single server doesn’t need to remember the client’s state,

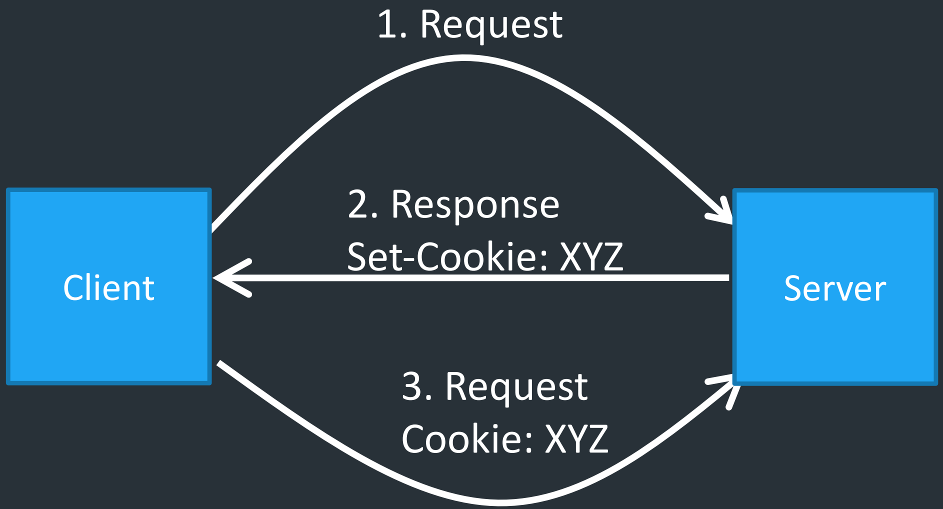


Modern web pages
- Dynamically served (server is an application that produces content specifically for client)
- Rely on state from client with cookies
- Contain lots of resources from many locations

Browser needs to …
- store client state
- Fetch all of the content on the page (recursive process to look everything up)
- Make asynchronous requests as you load the page

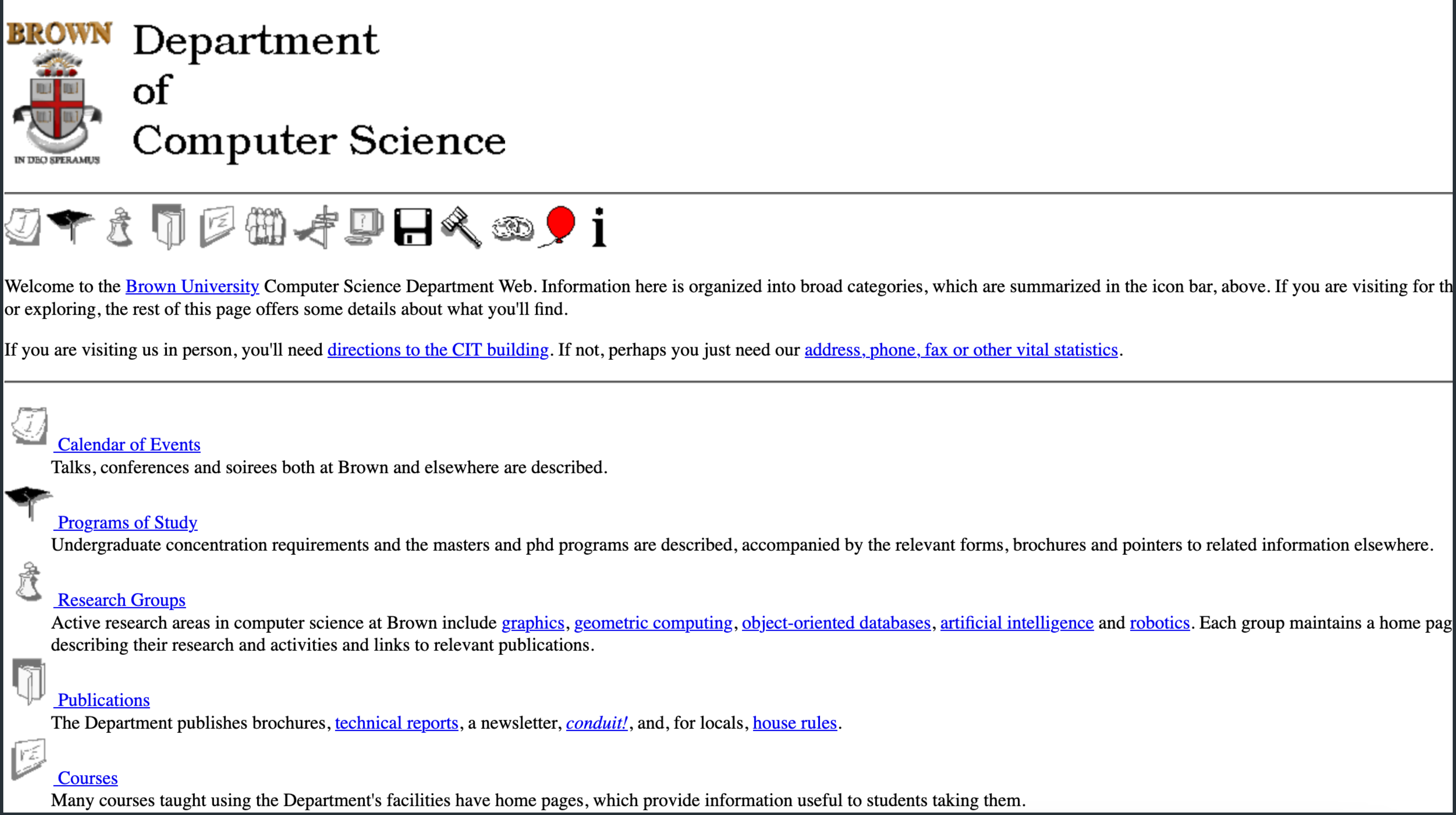
HTTP Cookies

- Client-side state maintenance
  - Client stores small state on behalf of server
  - Sends request in future requests to the server
  - Cookie value is meaningful to the server (e.g., session id)
- Can provide authentication



Anatomy of a Web Page

- HTML content
- A number of additional resources
  - Images
  - Scripts
  - Frames
- Browser makes one HTTP request for each object
  - Course web page: 14 objects
  - Modern web pages: hundreds of objects



Department of Computer Science

Welcome to the Brown University Computer Science Department Web. Information here is organized into broad categories, which are summarized in the icon bar, above. If you are visiting for th
or exploring, the rest of this page offers some details about what you'll find.

If you are visiting us in person, you'll need directions to the CIT building. If not, perhaps you just need our address, phone, fax or other vital statistics.

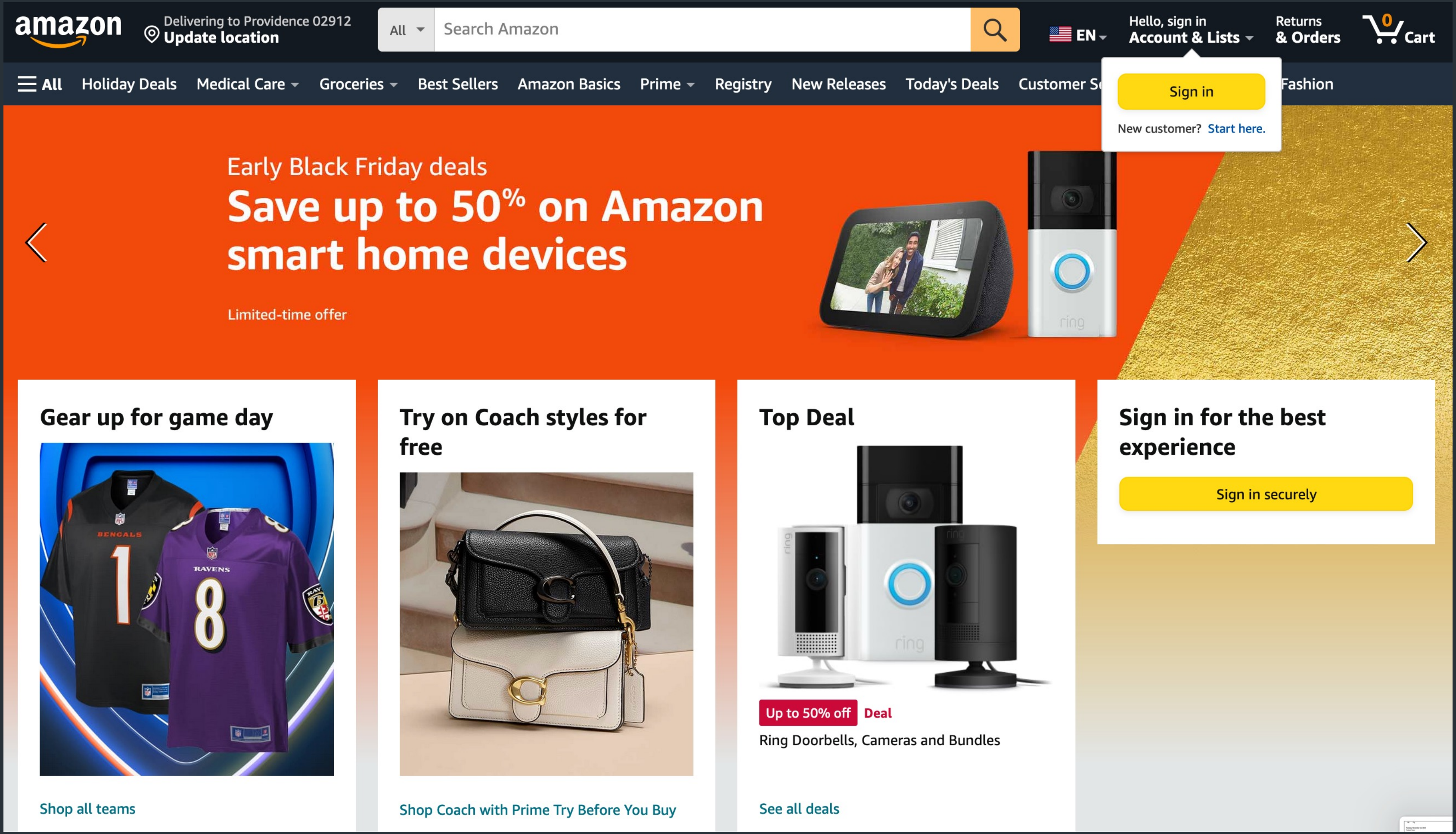
Calendar of Events
Talks, conferences and soirees both at Brown and elsewhere are described.

Programs of Study
Undergraduate concentration requirements and the masters and phd programs are described, accompanied by the relevant forms, brochures and pointers to related information elsewhere.

Research Groups
Active research areas in computer science at Brown include graphics, geometric computing, object-oriented databases, artificial intelligence and robotics. Each group maintains a home pag
describing their research and activities and links to relevant publications.

Publications
The Department publishes brochures, technical reports, a newsletter, *conduit!*, and, for locals, house rules.

Courses
Many courses taught using the Department's facilities have home pages, which provide information useful to students taking them.



Early Black Friday deals

Save up to 50% on Amazon smart home devices

Limited-time offer

Gear up for game day

Try on Coach styles for free

Top Deal

Ring Doorbells, Cameras and Bundles

Sign in for the best experience