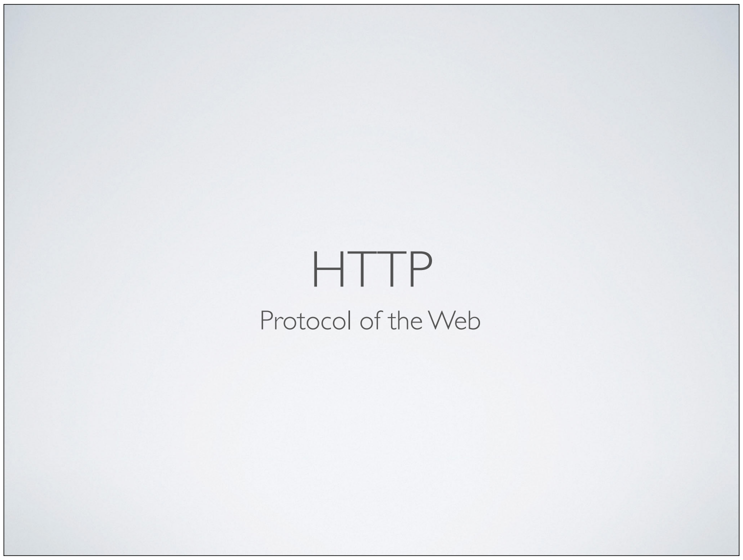
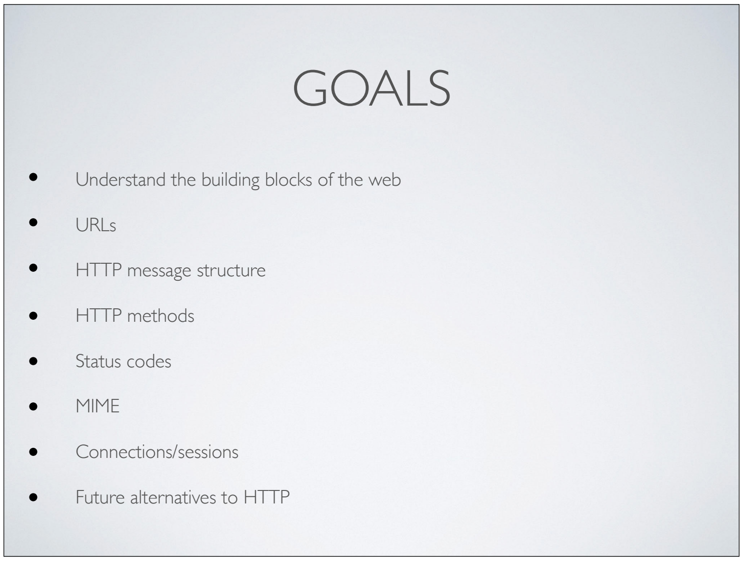


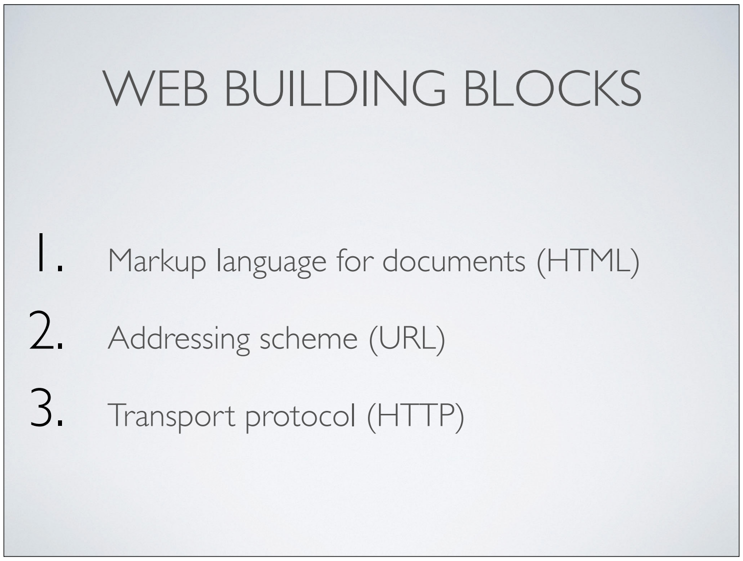
HTTP

Protocol of the Web



GOALS

- Understand the building blocks of the web
- URLs
- HTTP message structure
- HTTP methods
- Status codes
- MIME
- Connections/sessions
- Future alternatives to HTTP



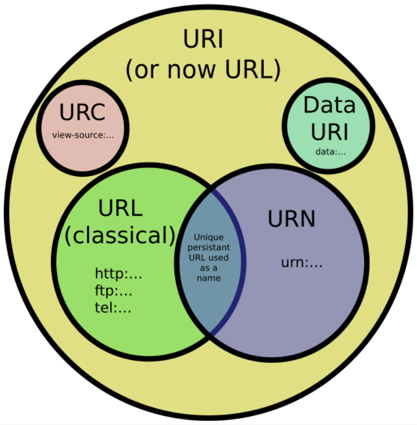
WEB BUILDING BLOCKS

1. Markup language for documents (HTML)
2. Addressing scheme (URL)
3. Transport protocol (HTTP)

URLS, URNS, AND URIS

- Need a way to name resources
- Names allow us to talk about things
- Uniform Resource Identifier (URI)
  - Uniform Resource Name (URN)
  - Uniform Resource Locator (URL)

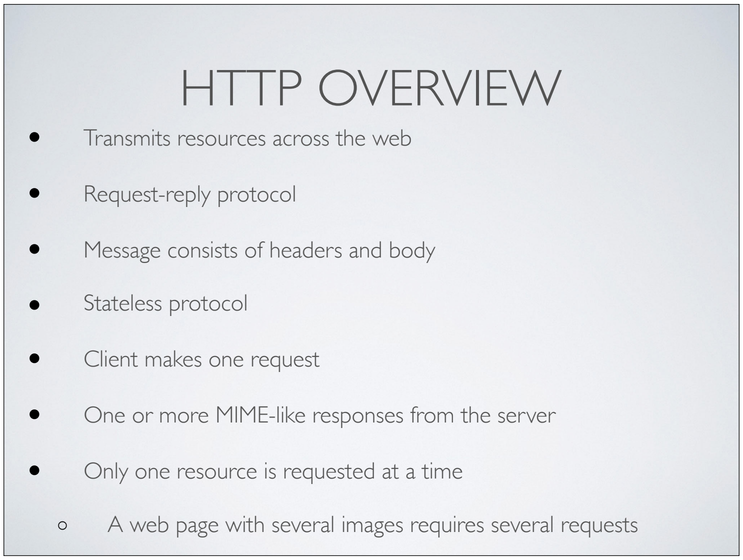
Venn diagram of URIs
as defined by the W3C



URLS

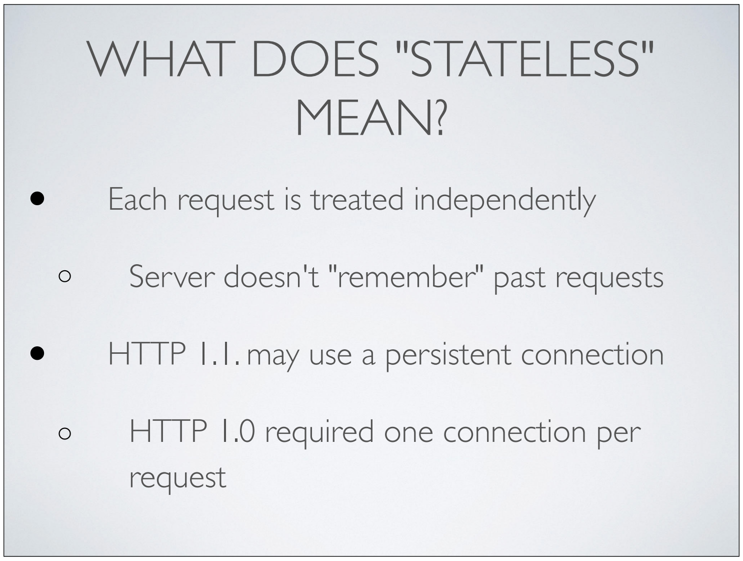
scheme://host[:port]/path/...[?query-string][#anchor]

- scheme = protocol, e.g, "http"
- host = HTTP server
- port = TCP port number (default is 80)
- path = logical filesystem path to a document
- query-string = key-value pairs, e.g., “id=10"
- anchor = position within the document



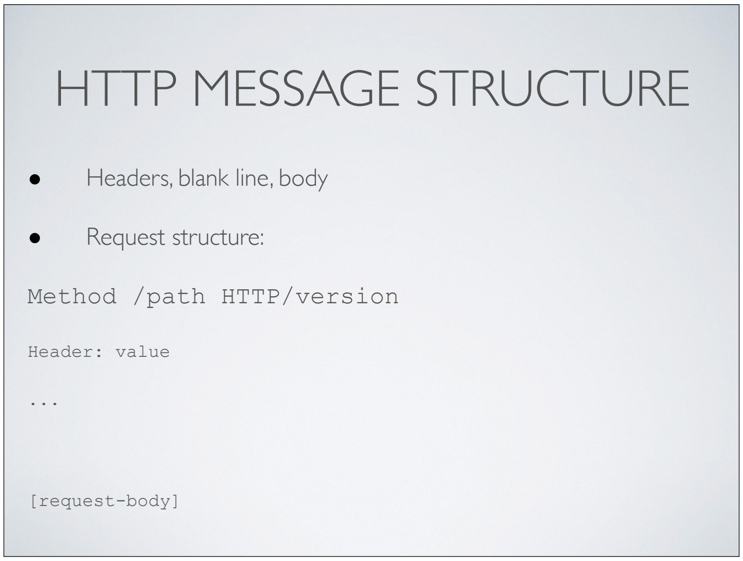
HTTP OVERVIEW

- Transmits resources across the web
- Request-reply protocol
- Message consists of headers and body
- Stateless protocol
- Client makes one request
- One or more MIME-like responses from the server
- Only one resource is requested at a time
  - A web page with several images requires several requests



WHAT DOES "STATELESS" MEAN?

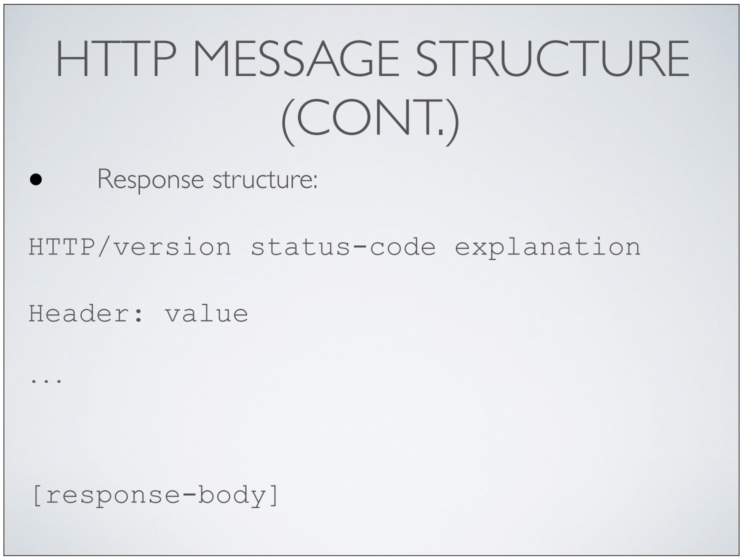
- Each request is treated independently
  - Server doesn't "remember" past requests
- HTTP 1.1. may use a persistent connection
  - HTTP 1.0 required one connection per request



HTTP MESSAGE STRUCTURE

- Headers, blank line, body
- Request structure:

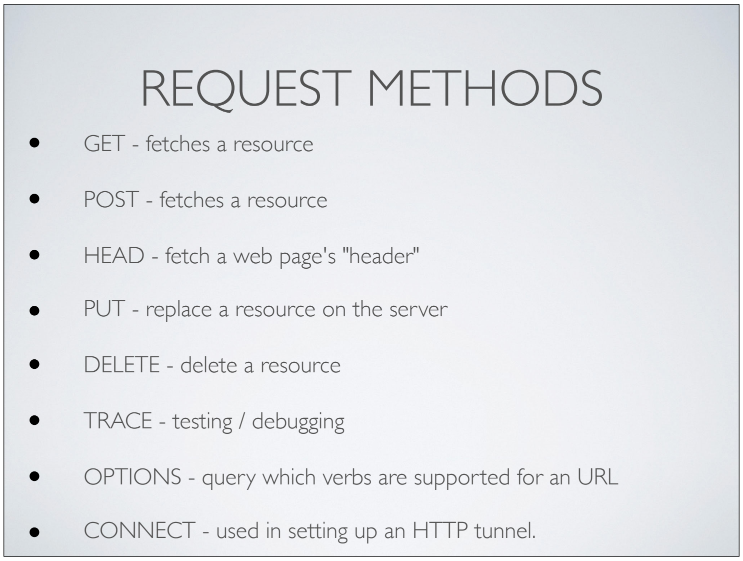
```
Method /path HTTP/version

Header: value

...


[request-body]
```



HTTP MESSAGE STRUCTURE (CONT.)

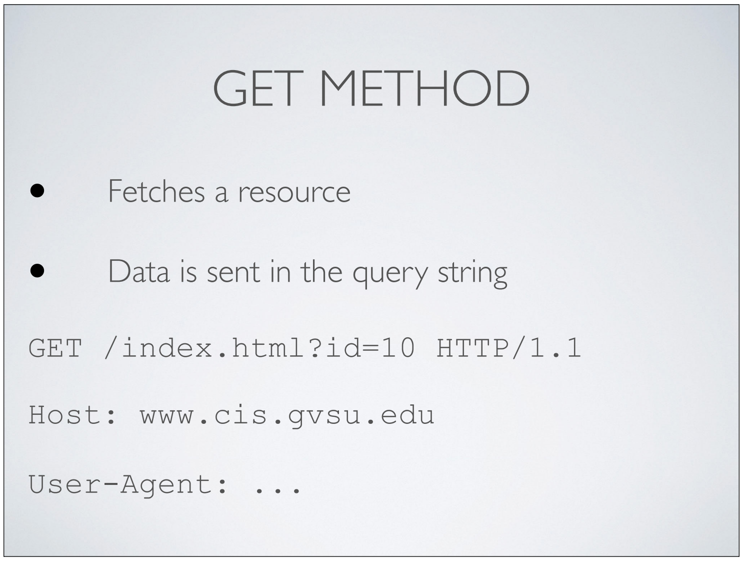
- Response structure:

```
HTTP/version status-code explanation

Header: value

...


[response-body]
```



REQUEST METHODS

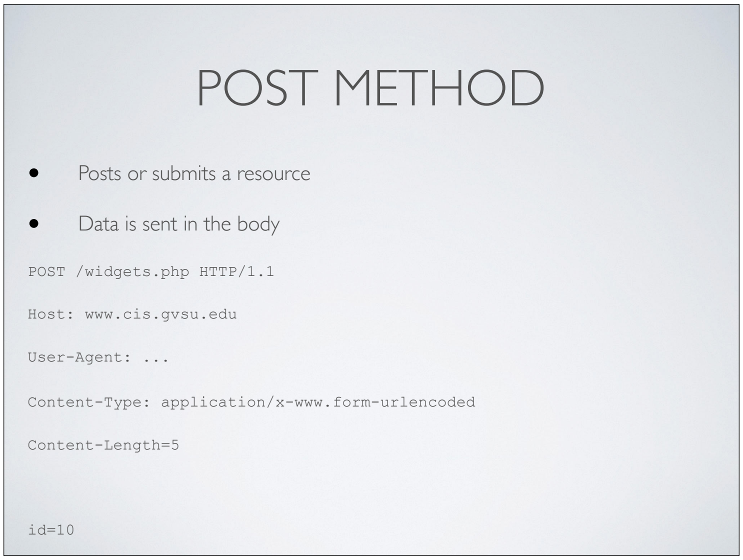
- GET - fetches a resource
- POST - fetches a resource
- HEAD - fetch a web page's "header"
- PUT - replace a resource on the server
- DELETE - delete a resource
- TRACE - testing / debugging
- OPTIONS - query which verbs are supported for an URL
- CONNECT - used in setting up an HTTP tunnel.



GET METHOD

- Fetches a resource
- Data is sent in the query string

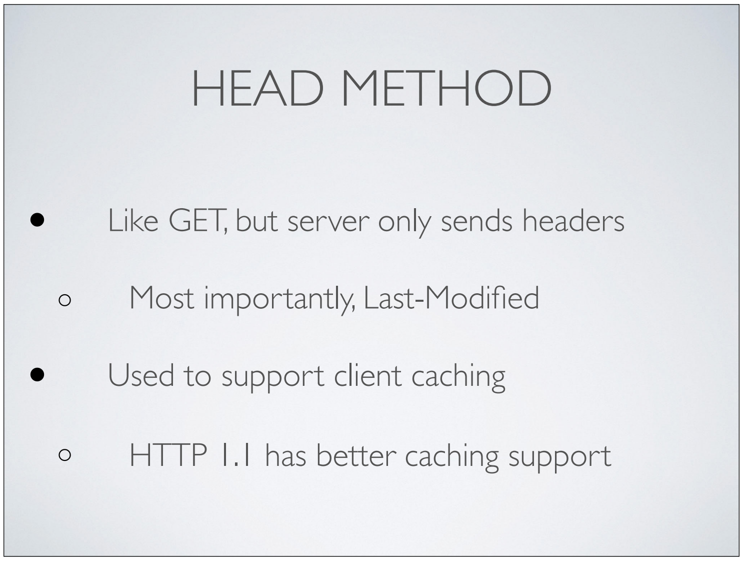
```
GET /index.html?id=10 HTTP/1.1

Host: www.cis.gvsu.edu

User-Agent: ...
```



POST METHOD

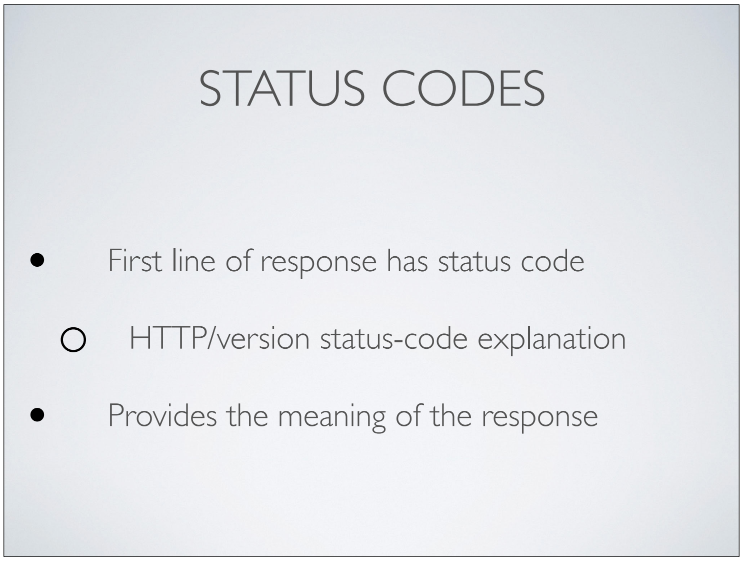
- Posts or submits a resource
- Data is sent in the body

```
POST /widgets.php HTTP/1.1

Host: www.cis.gvsu.edu

User-Agent: ...

Content-Type: application/x-www.form-urlencoded

Content-Length=5


id=10
```



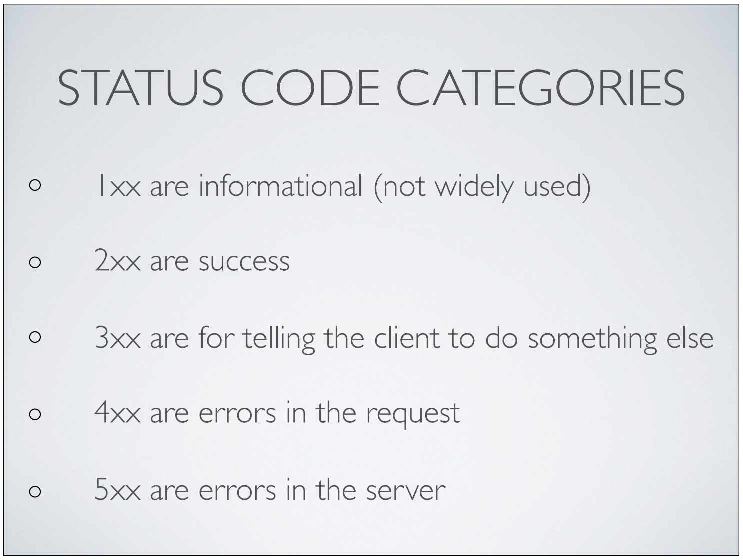
HEAD METHOD

- Like GET, but server only sends headers
  - Most importantly, Last-Modified
- Used to support client caching
  - HTTP 1.1 has better caching support



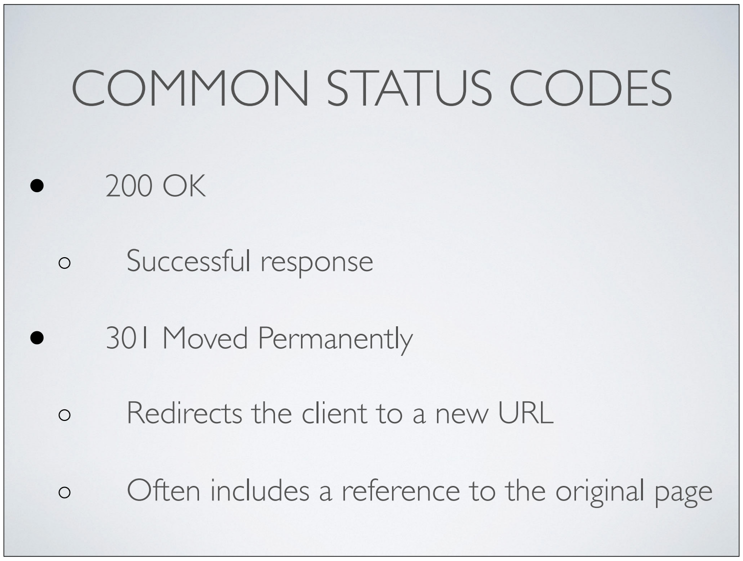
STATUS CODES

- First line of response has status code
  - HTTP/version status-code explanation
- Provides the meaning of the response



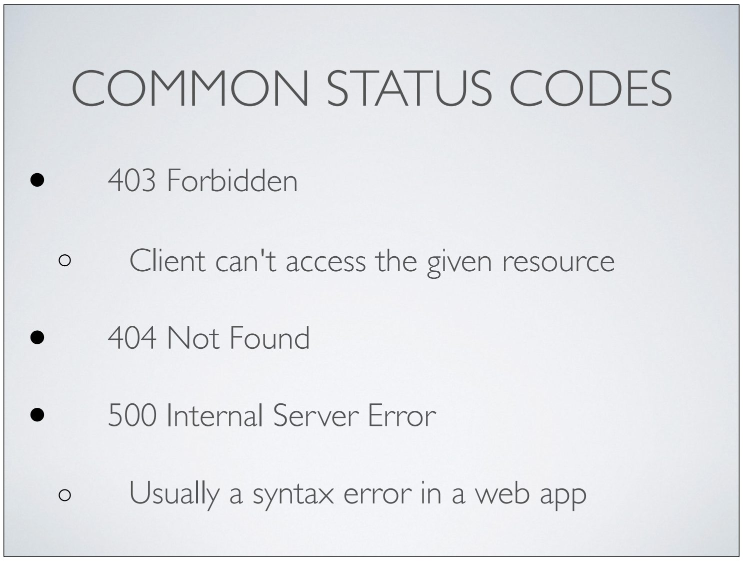
STATUS CODE CATEGORIES

- 1xx are informational (not widely used)
- 2xx are success
- 3xx are for telling the client to do something else
- 4xx are errors in the request
- 5xx are errors in the server



COMMON STATUS CODES

- 200 OK
  - Successful response
- 301 Moved Permanently
  - Redirects the client to a new URL
  - Often includes a reference to the original page



COMMON STATUS CODES

- 403 Forbidden
  - Client can't access the given resource
- 404 Not Found
- 500 Internal Server Error
  - Usually a syntax error in a web app

HTTP HEADERS

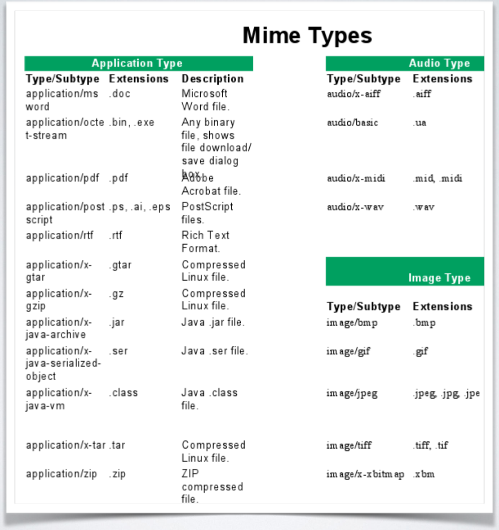
- General headers
  - Apply to both requests and responses
  - E.g., "Date"
- Request headers
  - E.g., "User-Agent", "Host", "Referer"

HTTP HEADERS

- Response headers
  - E.g., "Location", "Server"
- Entity headers
  - Describe message bodies
  - E.g., "Content-Type", "Content-Length"

MULTIMEDIA SUPPORT

- Multipurpose Internet Mail Extensions (MIME)
- Two headers:
  - Content-Encoding
    - Specifies how the body is encoded
    - "gzip" (gnu), "compress", "deflate" (zlib)
  - Content-Type
    - type / subtype
    - E.g., "text/html"
- Can be "multipart"
  - E.g., form data plus file to be uploaded



Mime Types

Application Type		
Type/Subtype	Extensions	Description
application/msword	.doc	Microsoft Word file.
application/octet-stream	.bin, .exe	Any binary file, shows file download/ save dialog box.
application/pdf	.pdf	Adobe Acrobat file.
application/postscript	.ps, .ai, .eps	PostScript files.
application/rtf	.rtf	Rich Text Format.
application/x-gtar	.gtar	Compressed Linux file.
application/x-gzip	.gz	Compressed Linux file.
application/x-java-archive	.jar	Java .jar file.
application/x-java-serialized-object	.ser	Java .ser file.
application/x-java-vm	.class	Java .class file.
application/x-tar	.tar	Compressed Linux file.
application/zip	.zip	ZIP compressed file.

Audio Type	
Type/Subtype	Extensions
audio/x-aiff	.aiff
audio/basic	.ua
audio/x-midi	.mid, .midi
audio/x-wav	.wav

Image Type	
Type/Subtype	Extensions
image/bmp	.bmp
image/gif	.gif
image/jpeg	.jpeg, .jpg, .jpe
image/tiff	.tiff, .tif
image/x-xbitmap	.xbm

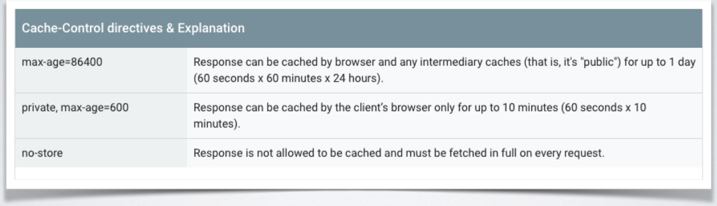
CACHING

- Caching - temporary storage
  - Why?
- Three types
  - Server-side
  - Browser-side
  - Proxy-side
- What to cache?
- When to use the cache?

CACHING HEADERS

- Cache-Control (HTTP/1.1)
  - public - resource can be cached
  - private - cannot be stored in a shared cache
  - no-cache - don't cache the resource
- Pragma (HTTP/1.0)
  - no-cache

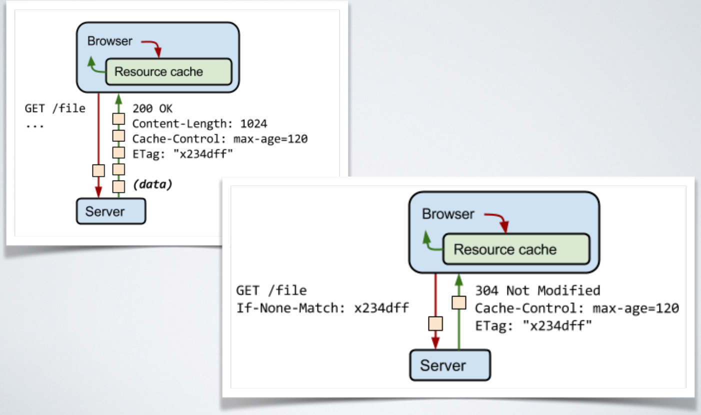
CACHE-CONTROL EXAMPLES



Cache-Control directives & Explanation	
max-age=86400	Response can be cached by browser and any intermediary caches (that is, it's "public") for up to 1 day (60 seconds x 60 minutes x 24 hours).
private, max-age=600	Response can be cached by the client's browser only for up to 10 minutes (60 seconds x 10 minutes).
no-store	Response is not allowed to be cached and must be fetched in full on every request.

https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/http-caching

ETAG VALIDATION



```
GET /file
...
```

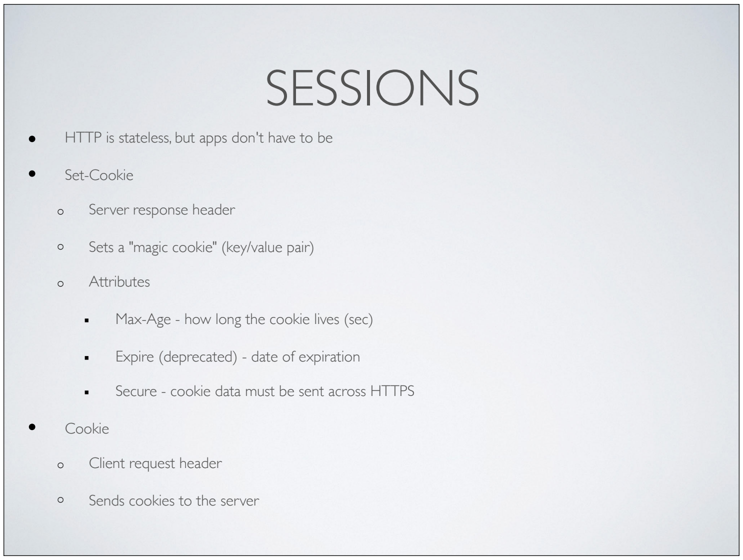
```
200 OK
Content-Length: 1024
Cache-Control: max-age=120
ETag: "x234dff"
(data)
```

```
GET /file
If-None-Match: x234dff
```

```
304 Not Modified
Cache-Control: max-age=120
ETag: "x234dff"
```

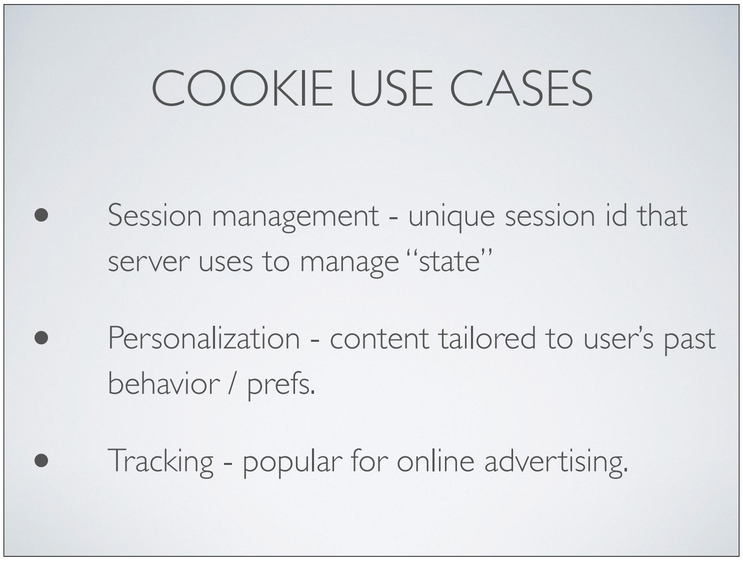
SECURITY

- Controlled via headers
- Not widely used
  - Most web apps implement their own authentication



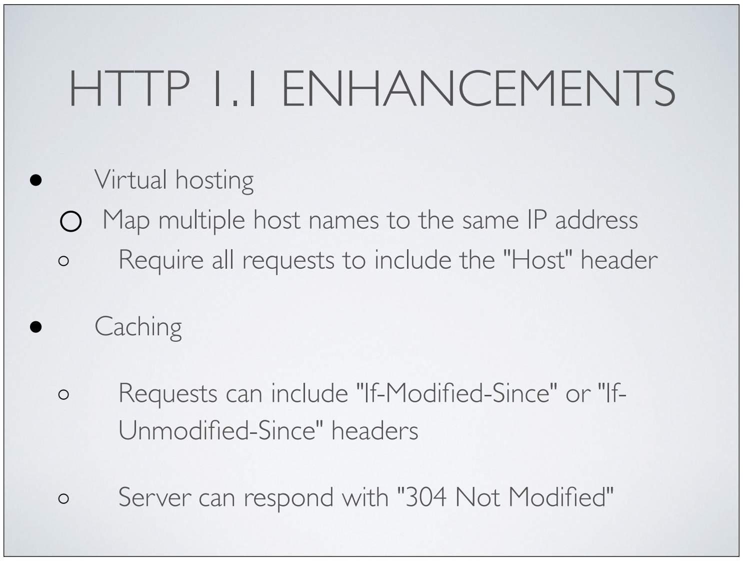
SESSIONS

- HTTP is stateless, but apps don't have to be
- Set-Cookie
  - Server response header
  - Sets a "magic cookie" (key/value pair)
  - Attributes
    - Max-Age - how long the cookie lives (sec)
    - Expire (deprecated) - date of expiration
    - Secure - cookie data must be sent across HTTPS
- Cookie
  - Client request header
  - Sends cookies to the server



COOKIE USE CASES

- Session management - unique session id that server uses to manage "state"
- Personalization - content tailored to user's past behavior / prefs.
- Tracking - popular for online advertising.



HTTP 1.1 ENHANCEMENTS

- Virtual hosting
  - Map multiple host names to the same IP address
  - Require all requests to include the "Host" header
- Caching
  - Requests can include "If-Modified-Since" or "If-Unmodified-Since" headers
  - Server can respond with "304 Not Modified"

PERSISTENT CONNECTIONS

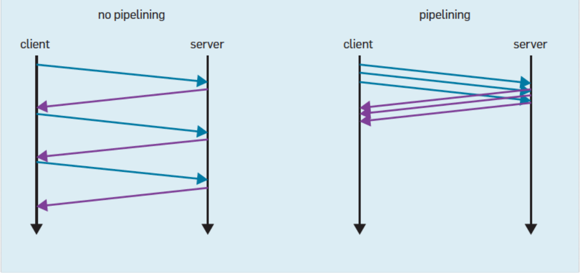
- TCP connections are expensive
- HTTP/1.0 required a new connection per request
  - High overhead
- HTTP/1.1 connections are persistent until closed with "Connection: close"
- Browsers can queue requests and pipeline them to servers

RECENT ENHANCEMENTS TO HTTP

- SPDY - Google interim implementation
  - Request multiplexing and prioritization
  - Header compression
  - Server push
  - Barriers to adoption?

HTTP 1.1 PIPELINING

Figure 4. HTTP pipelining allows multiple concurrent requests, reducing the number of round trips to the server.



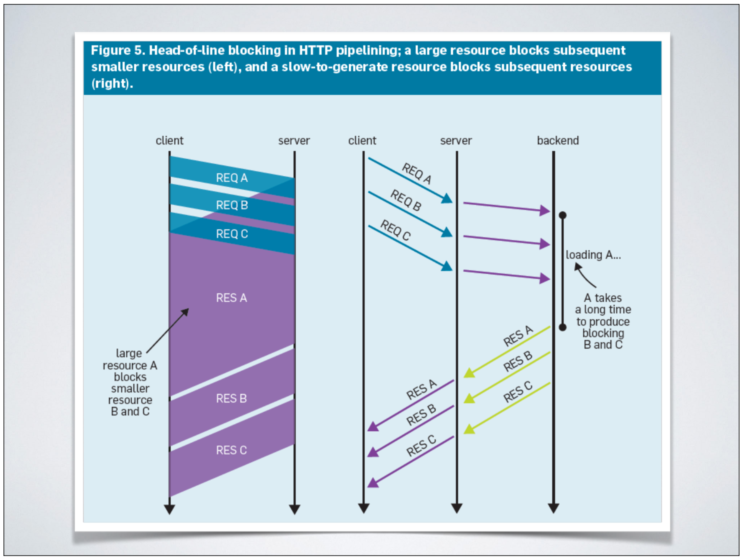


Figure 5. Head-of-line blocking in HTTP pipelining; a large resource blocks subsequent smaller resources (left), and a slow-to-generate resource blocks subsequent resources (right).

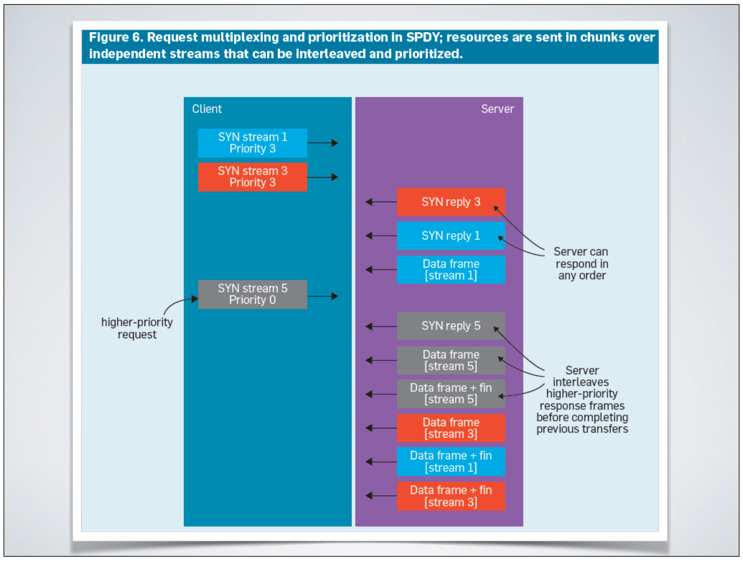
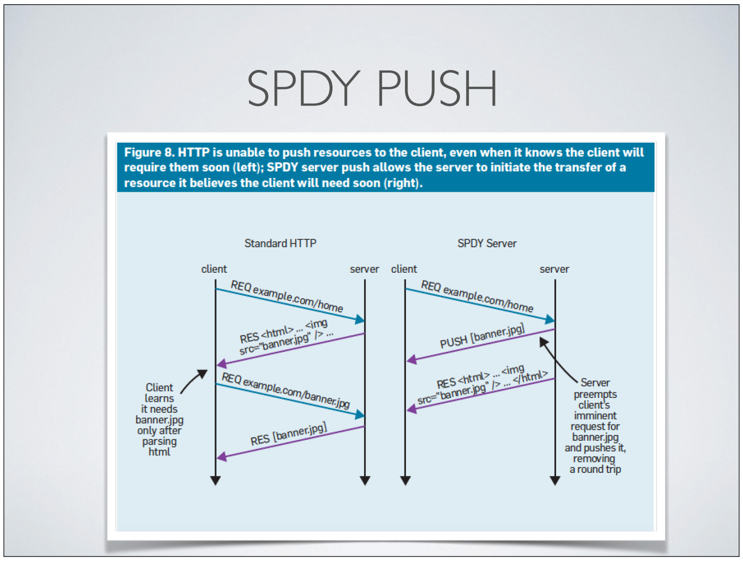


Figure 6. Request multiplexing and prioritization in SPDY; resources are sent in chunks over independent streams that can be interleaved and prioritized.



SPDY PUSH

Figure 8. HTTP is unable to push resources to the client, even when it knows the client will require them soon (left); SPDY server push allows the server to initiate the transfer of a resource it believes the client will need soon (right).

Standard HTTP
SPDY Server
client
server
client
server
REQ example.com/home
RES <html> ... <img src="banner.jpg" /> ...
REQ example.com/banner.jpg
RES [banner.jpg]
Client learns it needs banner.jpg only after parsing html
REQ example.com/home
PUSH [banner.jpg]
RES <html> ... <img src="banner.jpg" /> ... </html>
Server preempts client's imminent request for banner.jpg and pushes it, removing a round trip

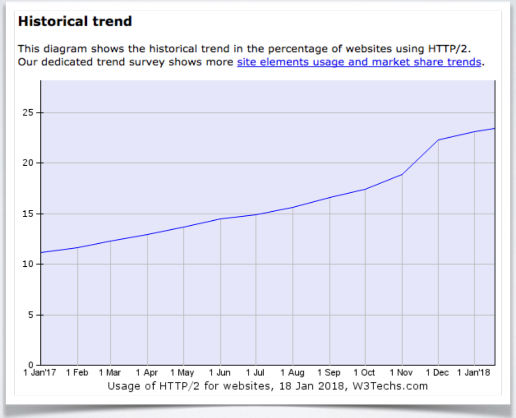
CURRENT STATUS

- HTTP/2 is now RFC 7540 at IETF
- An alternative to HTTP 1.1, does not obsolete.
- Based on SPDY.
- Google has embraced HTTP/2 and support for SPDY was deprecated in May 2016.

HTTP/2

- Less latency sensitive
- Fixes pipelining and the head of line blocking problem
- Eliminates the need to keep increasing the number of connections to each host
- Keeps all existing interfaces, all content, the URI formats and schemes

HTTP/2 ADOPTION

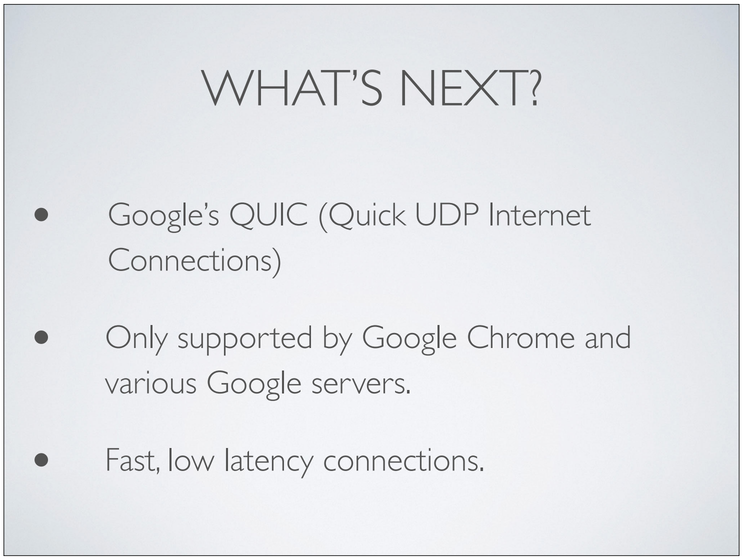


Historical trend

This diagram shows the historical trend in the percentage of websites using HTTP/2.
Our dedicated trend survey shows more site elements usage and market share trends.

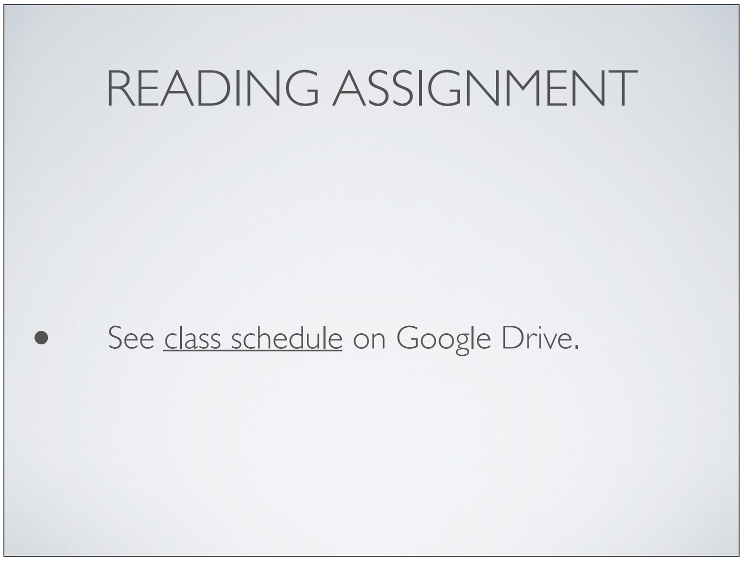
Usage of HTTP/2 for websites, 18 Jan 2018, W3Techs.com

https://w3techs.com/technologies/details/ce-http2/all/all



WHAT'S NEXT?

- Google's QUIC (Quick UDP Internet Connections)
- Only supported by Google Chrome and various Google servers.
- Fast, low latency connections.



READING ASSIGNMENT

- See class schedule on Google Drive.