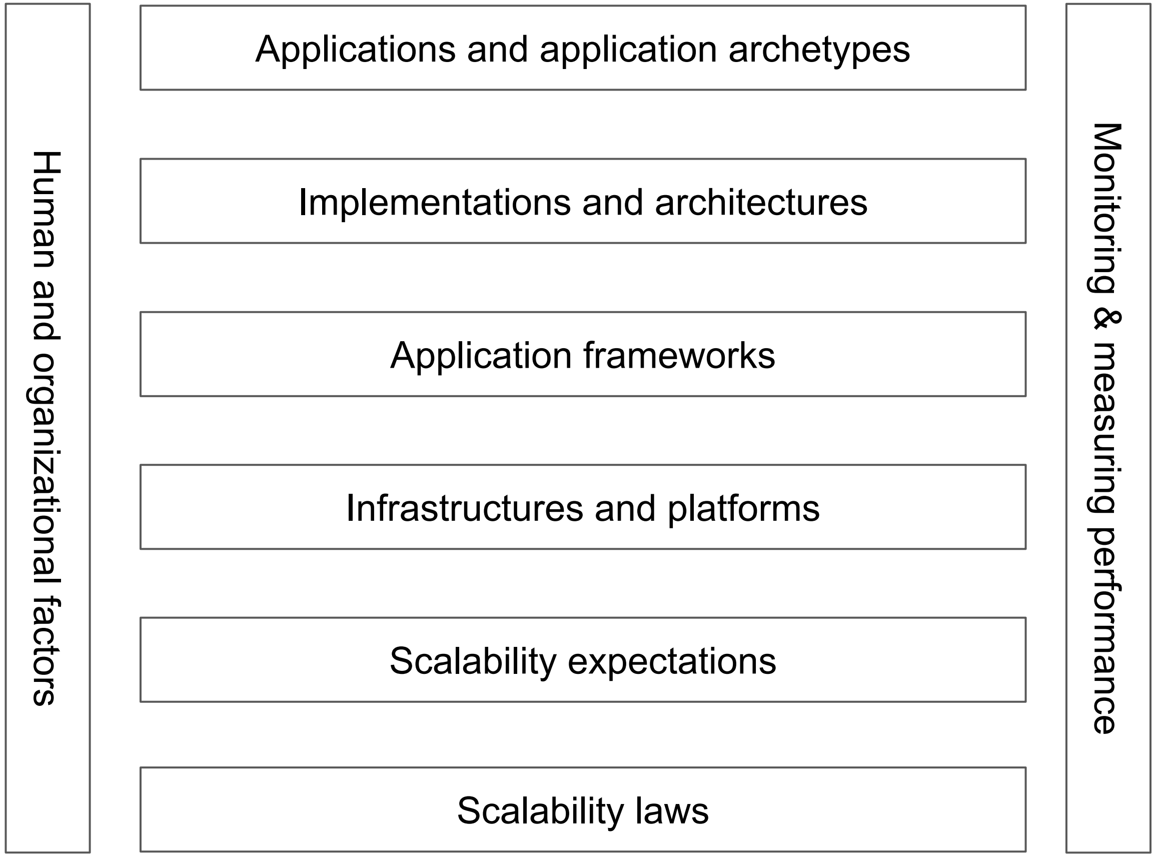
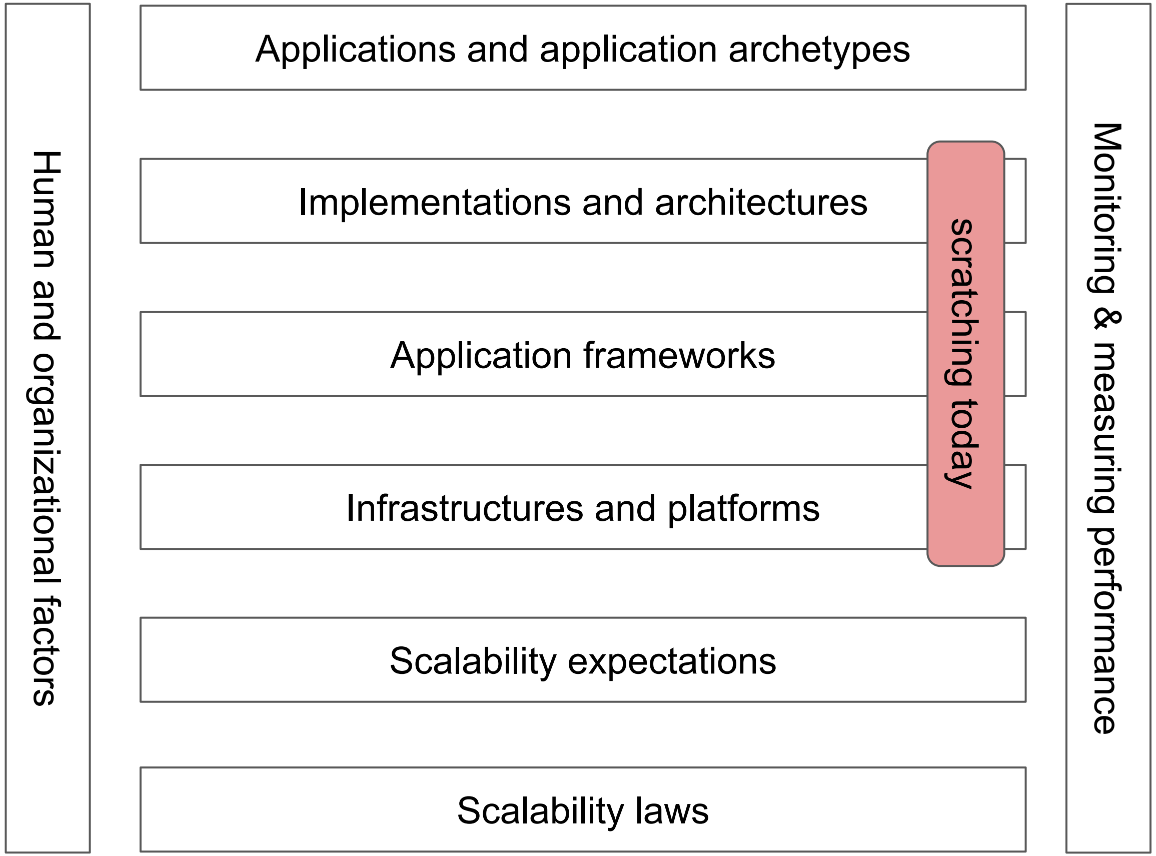
Designing and Building Scalable Web Applications

Lecture 2 / 31.10.2022

The Big Picture


Human and organizational factors
Applications and application archetypes
Implementations and architectures
Application frameworks
Infrastructures and platforms
Scalability expectations
Scalability laws
Monitoring & measuring performance

The Big Picture


Human and organizational factors
Applications and application archetypes
Implementations and architectures
Application frameworks
Infrastructures and platforms
Scalability expectations
Scalability laws
scratching today
Monitoring & measuring performance

Agenda

- Brief HTTP Refresher
- Caching and Content Delivery Networks
- Starting with Client-side Web Development

HTTP Refresher

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.

https://www.rfc-editor.org/rfc/rfc9110.html

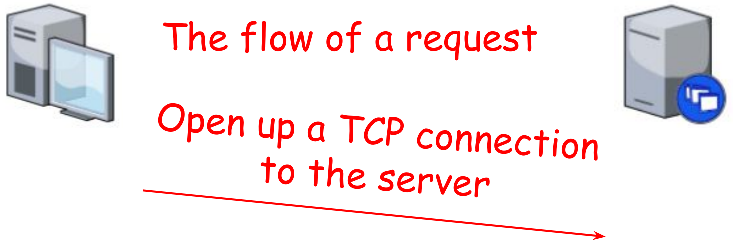
HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.

https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

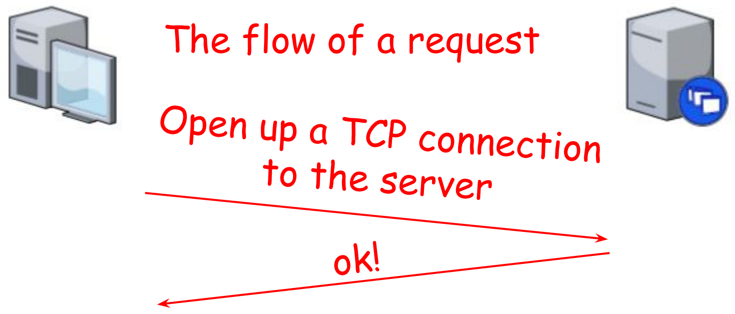
- HTTP is used for communicating with a server through exchanging messages.



https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

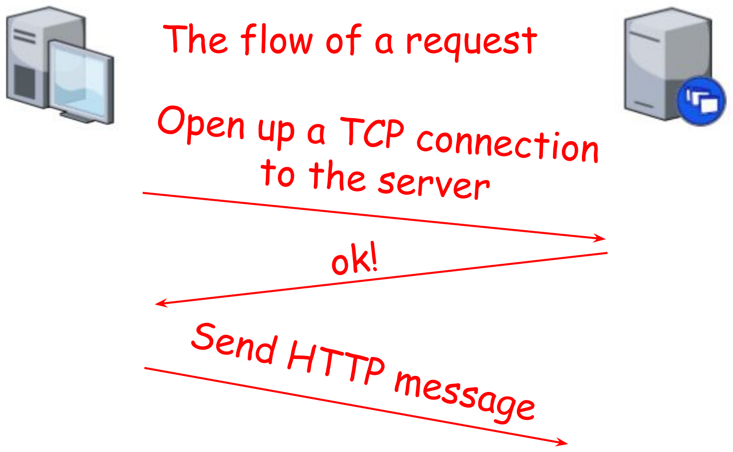
- HTTP is used for communicating with a server through exchanging messages.



https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

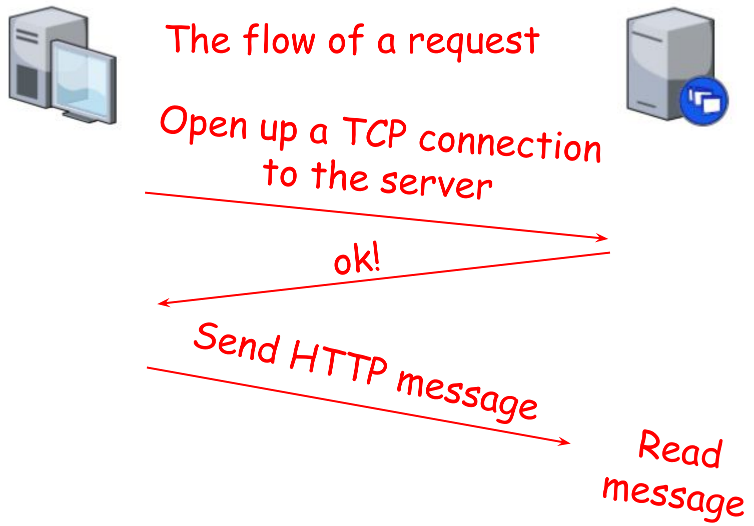
- HTTP is used for communicating with a server through exchanging messages.



https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

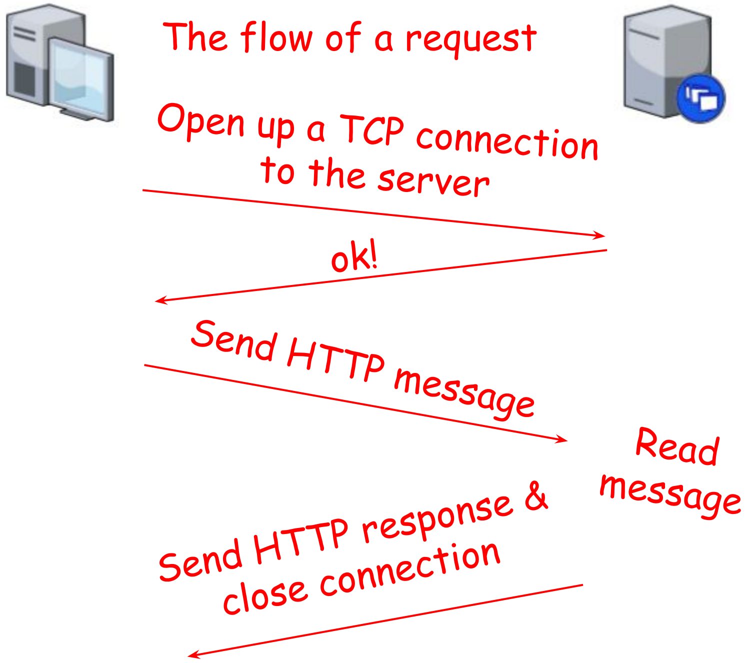
- HTTP is used for communicating with a server through exchanging messages.



https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.

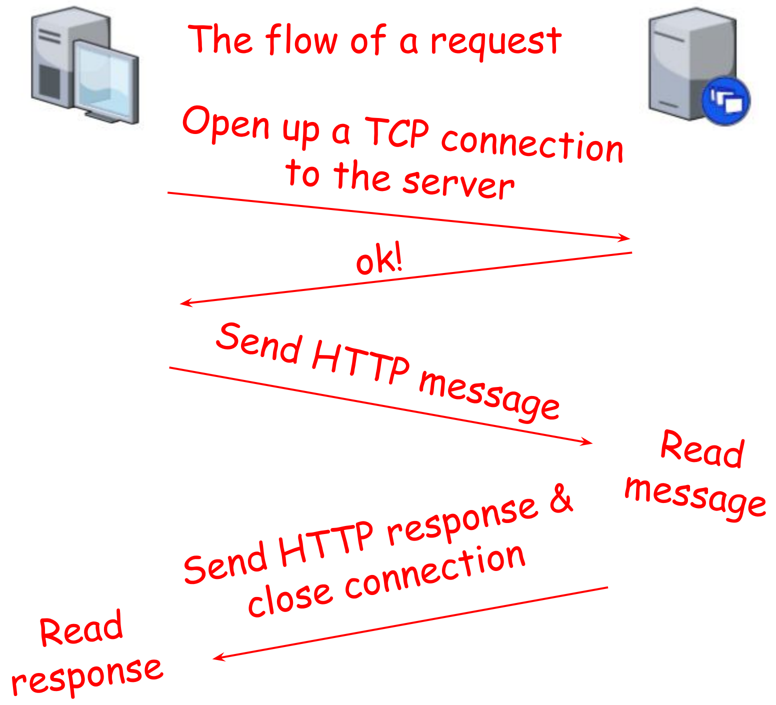


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection

https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.

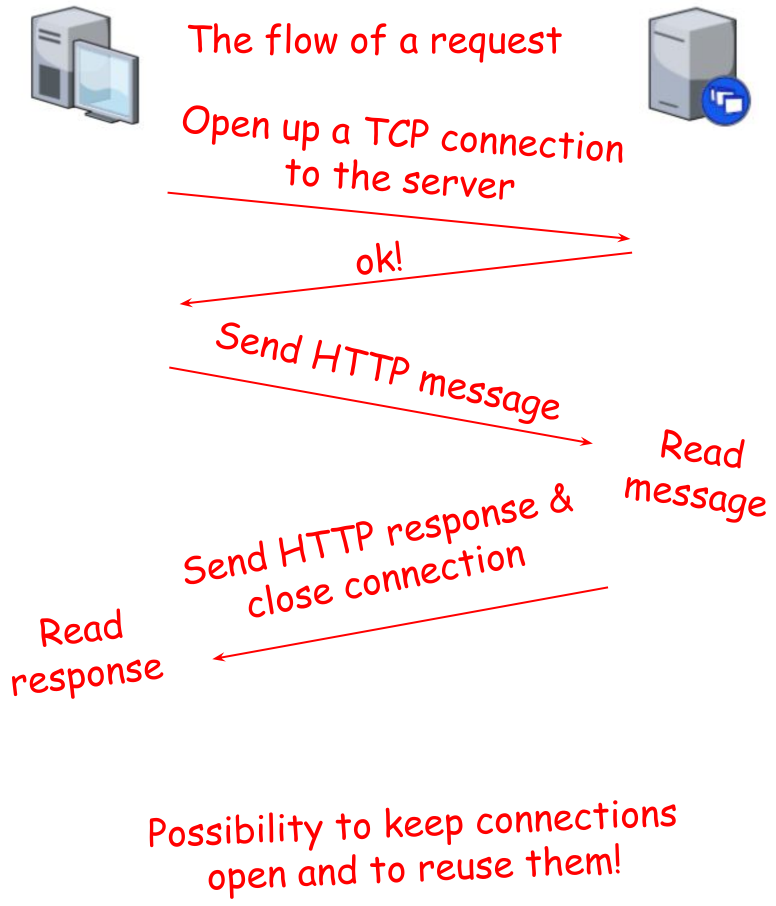


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection
Read response

https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

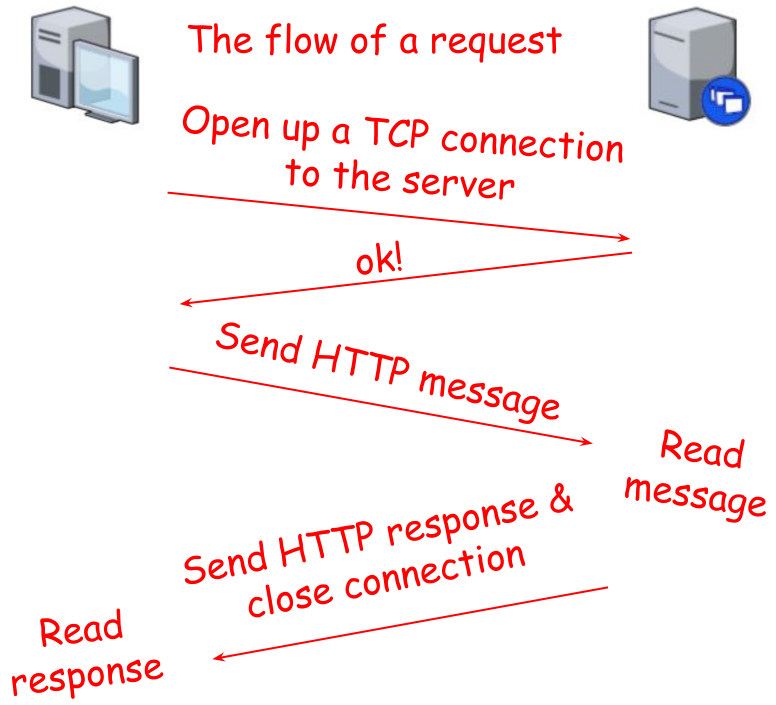
- HTTP is used for communicating with a server through exchanging messages.


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection
Read response
Possibility to keep connections open and to reuse them!

https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.
- HTTP is stateless: every request is independent of other requests.


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection
Read response

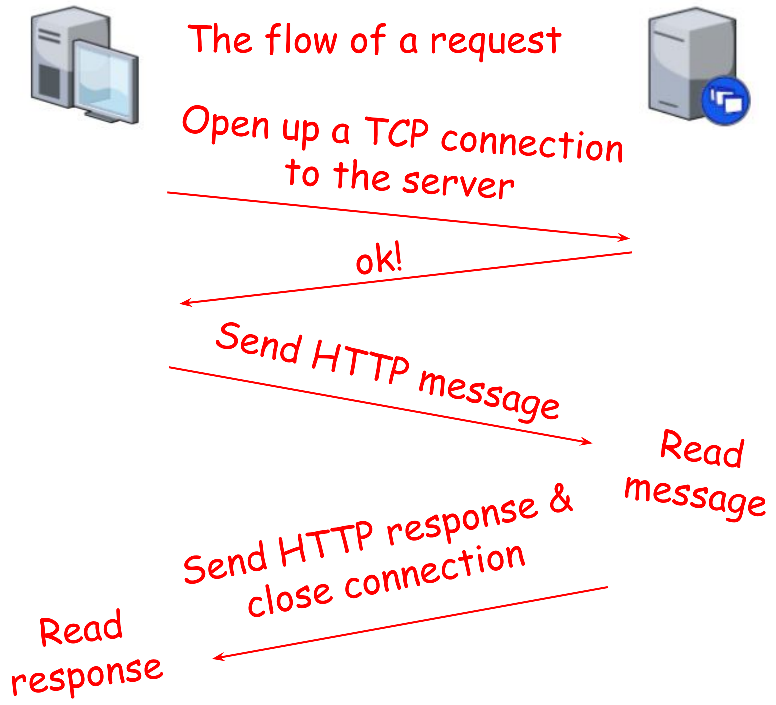
Possibility to keep connections open and to reuse them!

https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.
- HTTP is stateless: every request is independent of other requests.

But… cookies, tokens, etc!


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection
Read response

Possibility to keep connections open and to reuse them!

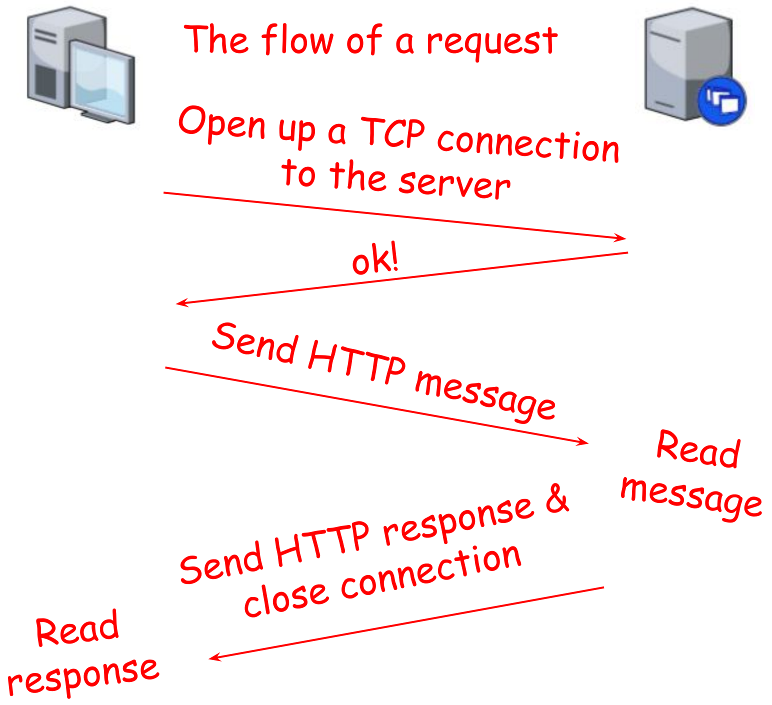
https://www.rfc-editor.org/rfc/rfc9110.html

HTTP Refresher

- HTTP is used for communicating with a server through exchanging messages.
- HTTP is stateless: every request is independent of other requests.

But… cookies, tokens, etc!

- Older RFC: Persistent connections to the server should be limited (no more than 2 connections per server).


The flow of a request
Open up a TCP connection to the server
ok!
Send HTTP message
Read message
Send HTTP response & close connection
Read response

Possibility to keep connections open and to reuse them!

https://www.rfc-editor.org/rfc/rfc9110.html

https://www.w3.org/Protocols/rfc2616/rfc2616-sec8.html#sec8.1.4

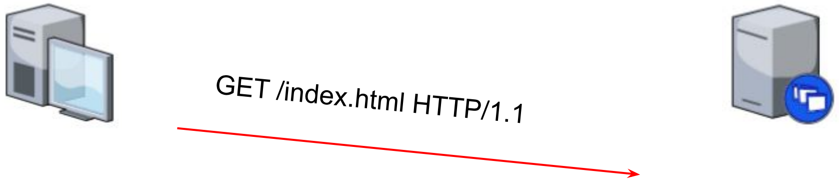
Retrieving a web page

Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.

Retrieving a web page

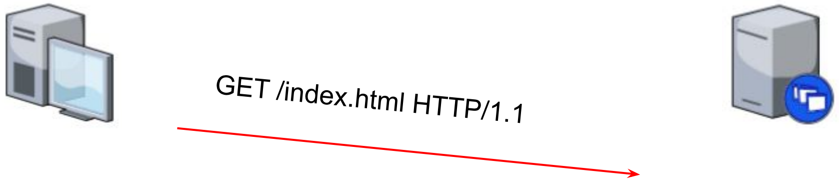
- When retrieving a web page, the client makes a request to the server, asking for a resource.



Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.

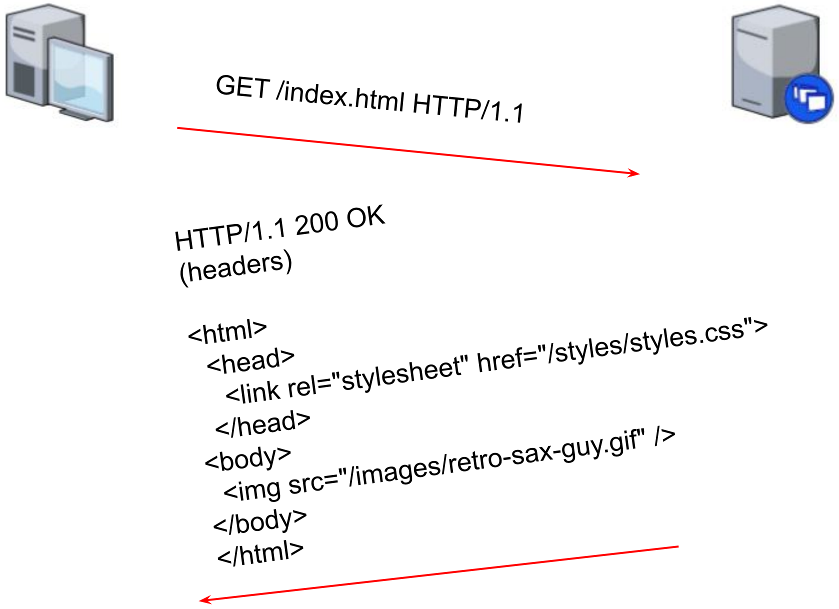
- The server responds with the resource, which is then interpreted by the client.



Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.

- The server responds with the resource, which is then interpreted by the client.



```
GET /index.html HTTP/1.1
```

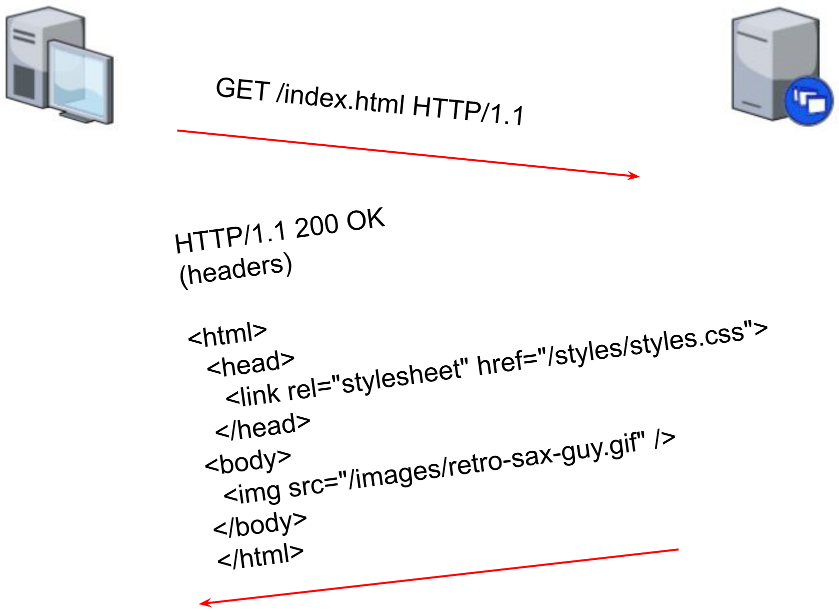
```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.

- The server responds with the resource, which is then interpreted by the client.

- If the response contains resources that the client needs, the client retrieves additional resources.

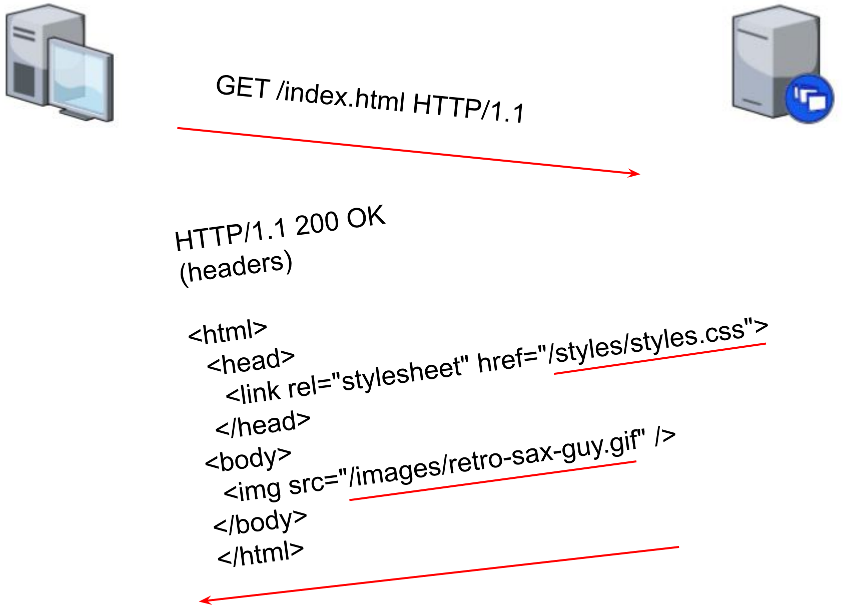


```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

 <html>
   <head>
     <link rel="stylesheet" href="/styles/styles.css">
   </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
  </html>
```

Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.
- The server responds with the resource, which is then interpreted by the client.
- If the response contains resources that the client needs, the client retrieves additional resources.

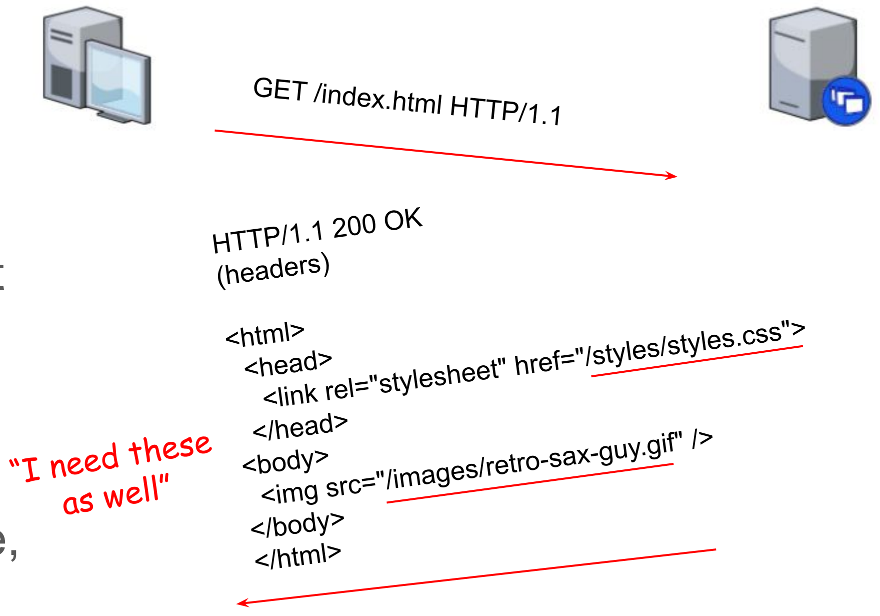


```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.
- The server responds with the resource, which is then interpreted by the client.
- If the response contains resources that the client needs, the client retrieves additional resources.

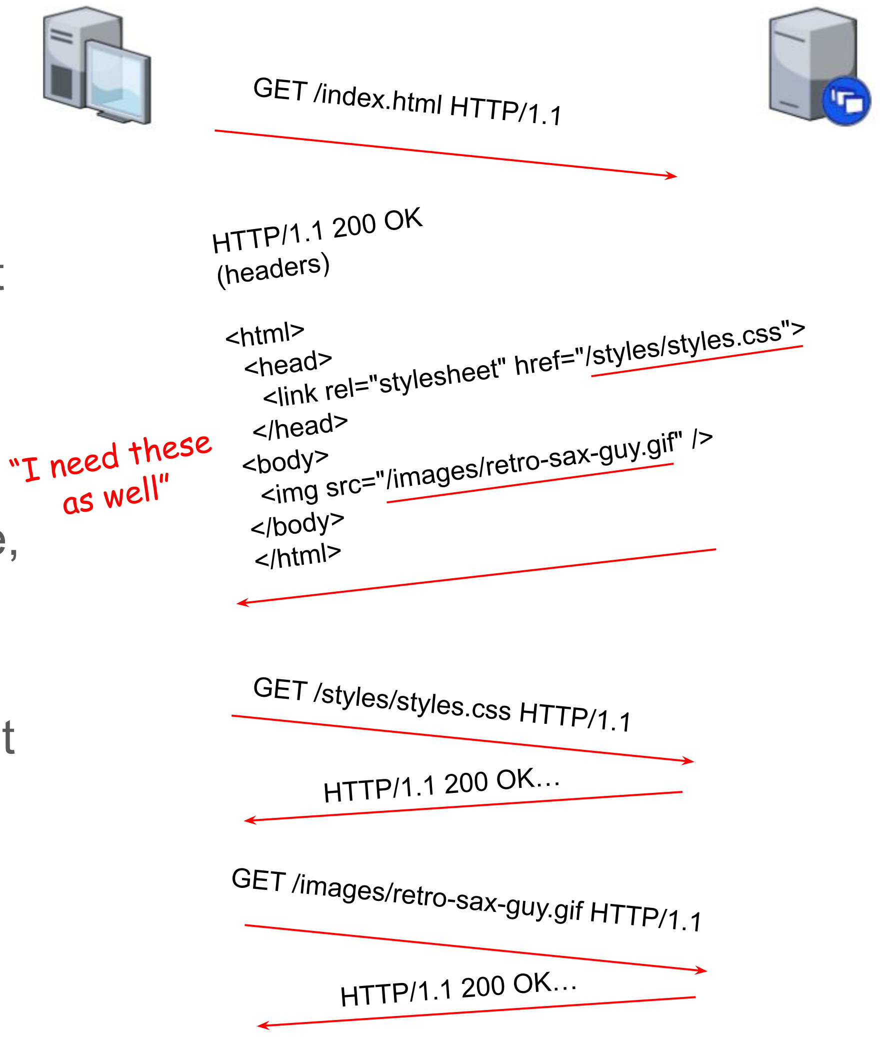


```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

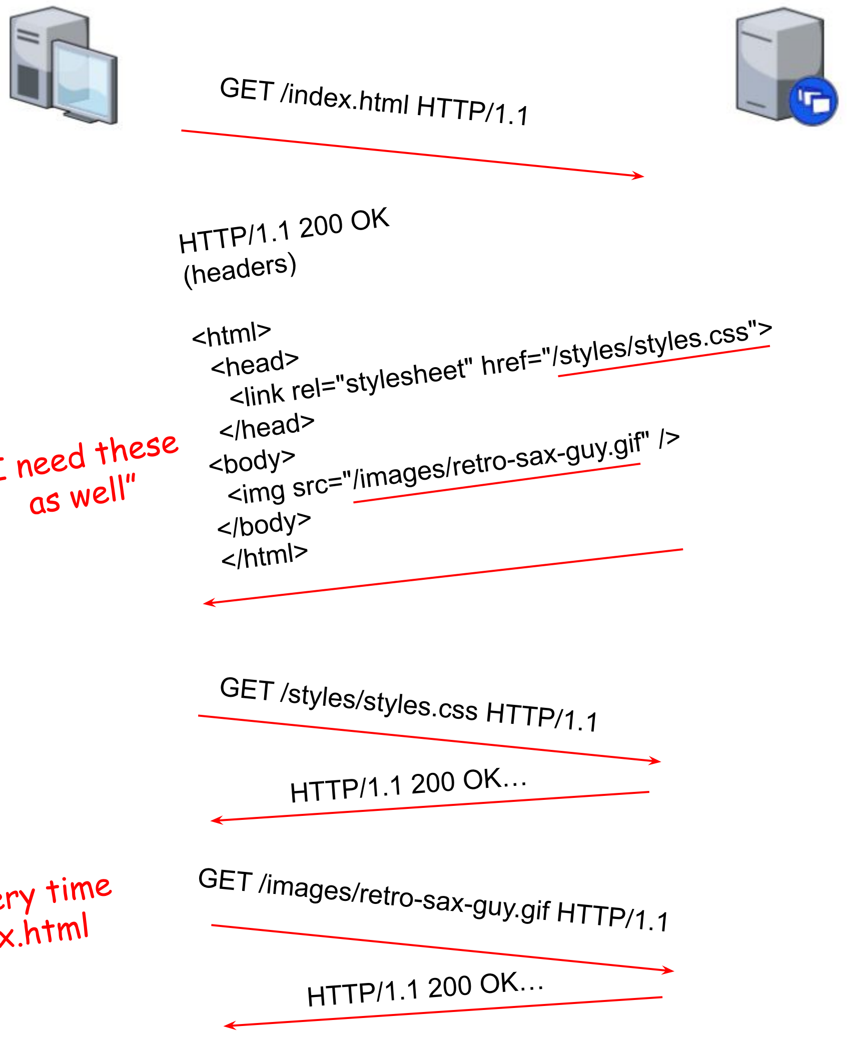
Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.
- The server responds with the resource, which is then interpreted by the client.
- If the response contains resources that the client needs, the client retrieves additional resources.


GET /index.html HTTP/1.1
HTTP/1.1 200 OK
(headers)
<html>
<head>
<link rel="stylesheet" href="/styles/styles.css">
</head>
<body>
<img src="/images/retro-sax-guy.gif" />
</body>
</html>
"I need these as well"
GET /styles/styles.css HTTP/1.1
HTTP/1.1 200 OK...
GET /images/retro-sax-guy.gif HTTP/1.1
HTTP/1.1 200 OK...

Retrieving a web page

- When retrieving a web page, the client makes a request to the server, asking for a resource.
- The server responds with the resource, which is then interpreted by the client.
- If the response contains resources that the client needs, the client retrieves additional resources.



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

"I need these as well"

```
GET /styles/styles.css HTTP/1.1
```

```
HTTP/1.1 200 OK...
```

```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK...
```

By default, this happens every time the client retrieves /index.html

Retrieving a web page

- There may be multiple resources that need to be fetched.

Retrieving a web page

amazon.com is loaded with about 300 requests.

- There may be multiple resources that need to be fetched.

Retrieving a web page

- There may be multiple resources that need to be fetched.

amazon.com is loaded with about 300 requests.

bbc.com is loaded with about 200 requests.

Retrieving a web page

- There may be multiple resources that need to be fetched.

amazon.com is loaded with about 300 requests.

bbc.com is loaded with about 200 requests.

aalto.fi/en is loaded with about 80 requests.

Retrieving a web page

- There may be multiple resources that need to be fetched.

amazon.com is loaded with
about 300 requests.

bbc.com is loaded with about
200 requests.

aalto.fi/en is loaded with
about 80 requests.

fitech101.aalto.fi/web-software-development/
is loaded with about 50 requests.

Retrieving a web page

- There may be multiple resources that need to be fetched.

- The number of concurrent connections from a client is limited.

amazon.com is loaded with about 300 requests.

bbc.com is loaded with about 200 requests.

aalto.fi/en is loaded with about 80 requests.

fitech101.aalto.fi/web-software-development/ is loaded with about 50 requests.

Retrieving a web page

- There may be multiple resources that need to be fetched.
- The number of concurrent connections from a client is limited.
- We may have to wait to load some of the resources.

amazon.com is loaded with about 300 requests.

bbc.com is loaded with about 200 requests.

aalto.fi/en is loaded with about 80 requests.

fitech101.aalto.fi/web-software-development/ is loaded with about 50 requests.

Retrieving a web page

- There may be multiple resources that need to be fetched.
- The number of concurrent connections from a client is limited.
- We may have to wait to load some of the resources.
- How to get around this?
    - Use caches to limit unnecessary loading.
    - Distribute resources over servers.

amazon.com is loaded with about 300 requests.

bbc.com is loaded with about 200 requests.

aalto.fi/en is loaded with about 80 requests.

fitech101.aalto.fi/web-software-development/ is loaded with about 50 requests.

Caching

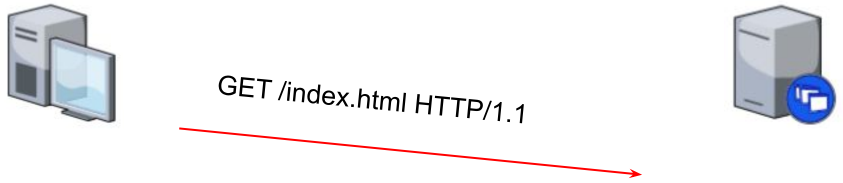
Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.

https://www.rfc-editor.org/rfc/rfc9111.html

Caching

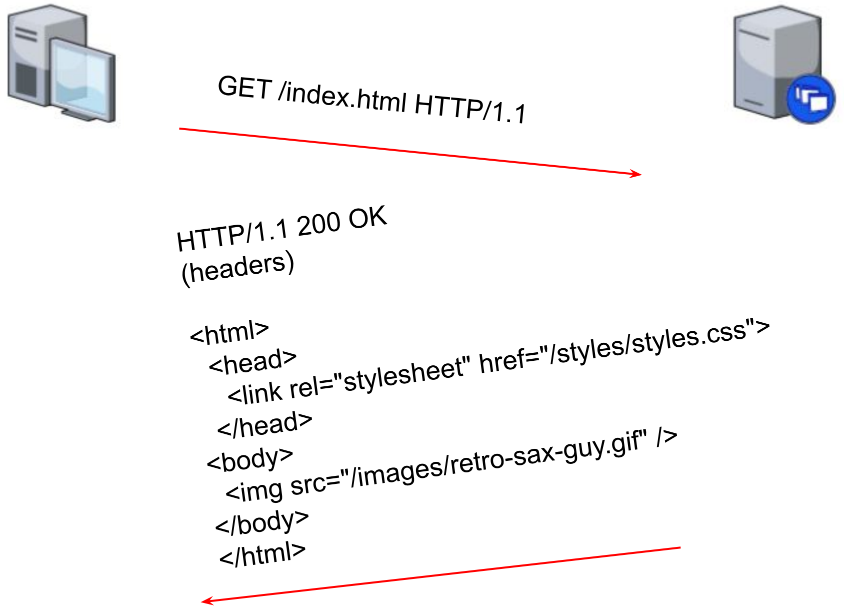
- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.



https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.



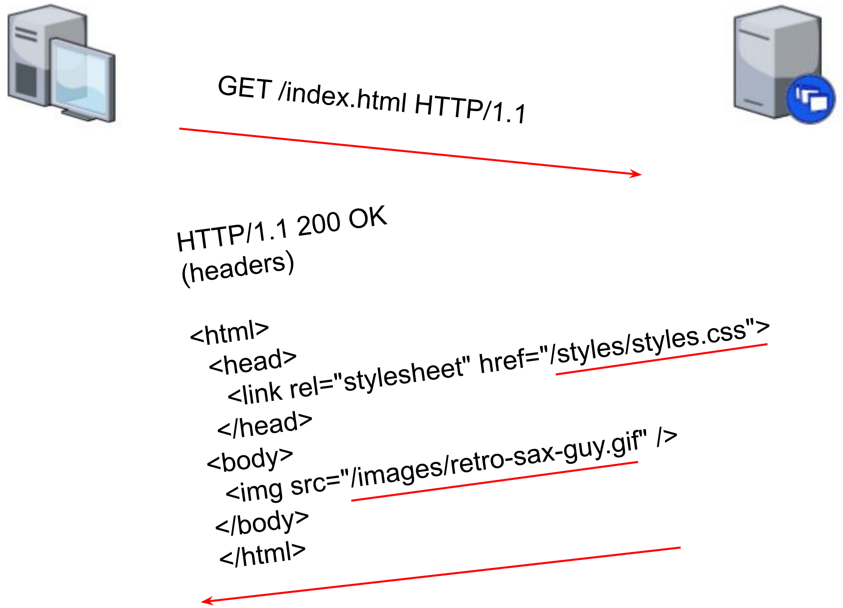
```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.

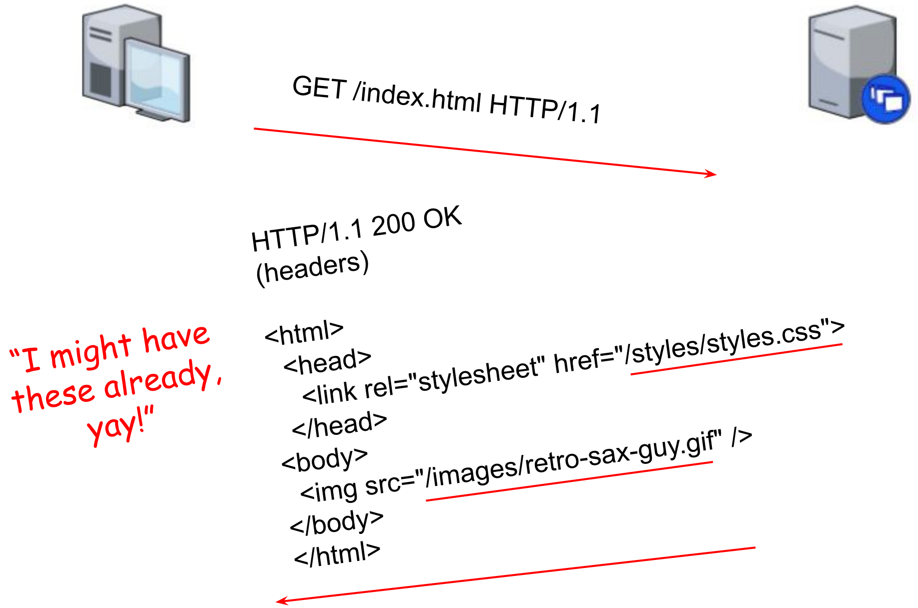


```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.



```
GET /index.html HTTP/1.1
```

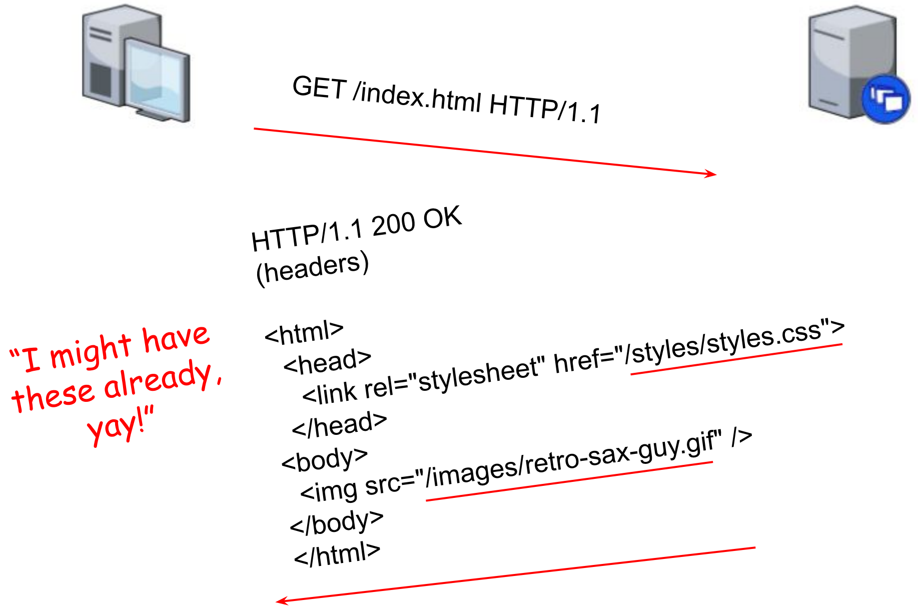
```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.

- Two types of cache:
  - Private cache on the client
  - Shared cache on the server (or *a* server)



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

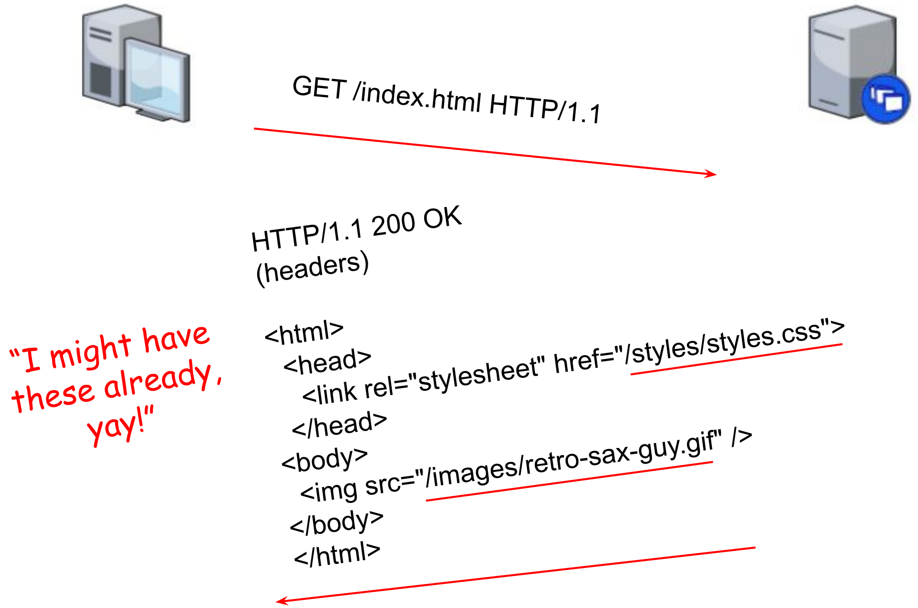
https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.

- Two types of cache:
  - Private cache on the client
  - Shared cache on the server (or *a* server)

- Client-side caching managed using HTTP headers
  - Recently also Web Cache API



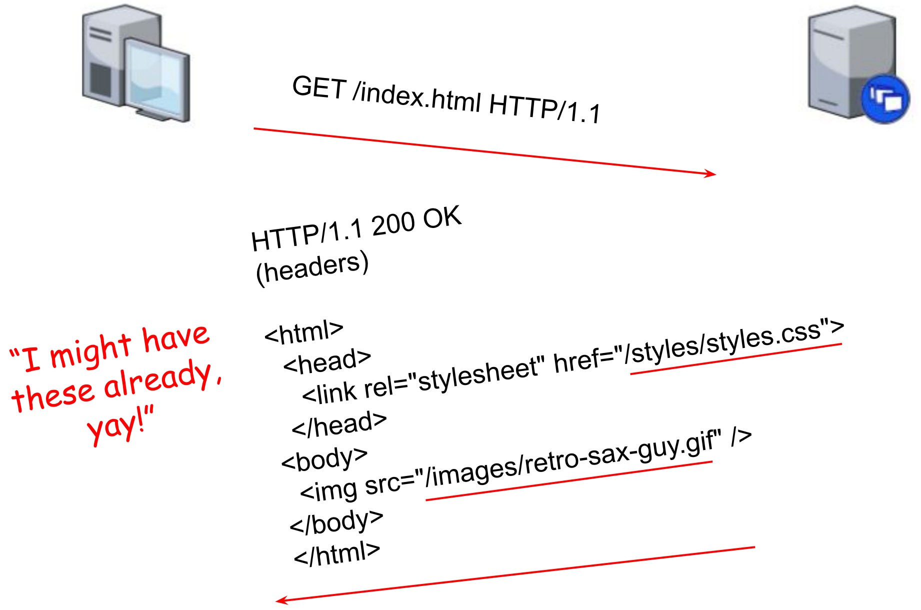
```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

https://www.rfc-editor.org/rfc/rfc9111.html

Caching

- Caching means storing data in a temporary location, reducing the need to re-retrieve the data.
- Two types of cache:
  - Private cache on the client
  - Shared cache on the server (or *a* server)
- Client-side caching managed using HTTP headers
  - Recently also Web Cache API
- Multiple shared (server-side) cache implementations / approaches



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet" href="/styles/styles.css">
  </head>
  <body>
    <img src="/images/retro-sax-guy.gif" />
  </body>
</html>
```

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

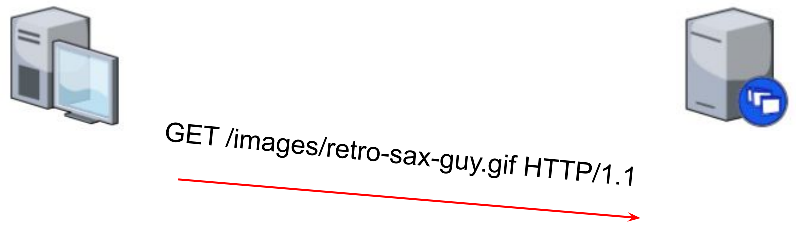
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching



- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

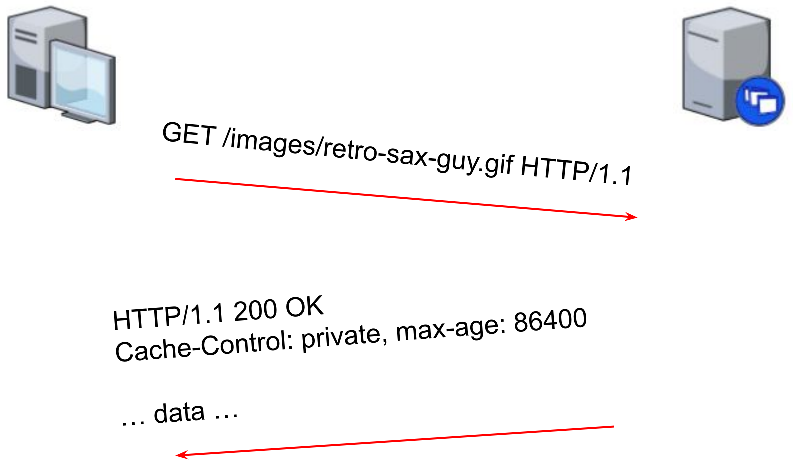
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

- Headers received in HTTP responses and sent in HTTP requests.


GET /images/retro-sax-guy.gif HTTP/1.1
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
… data …

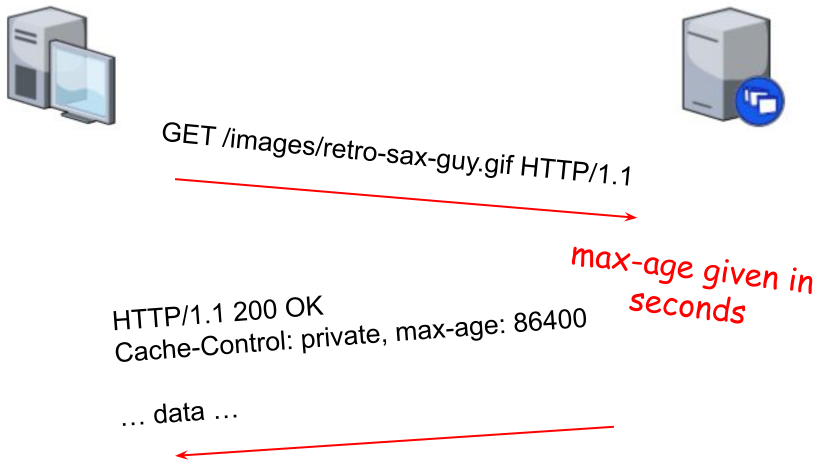
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400

… data …
```

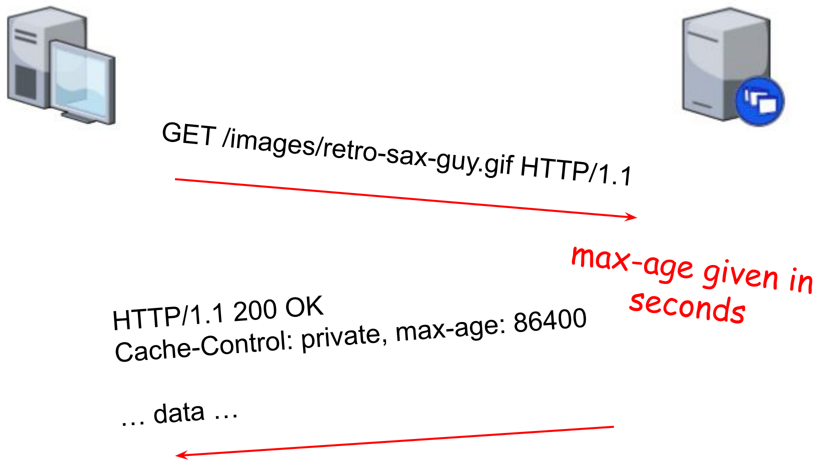
max-age given in seconds

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400

... data ...
```

max-age given in seconds

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

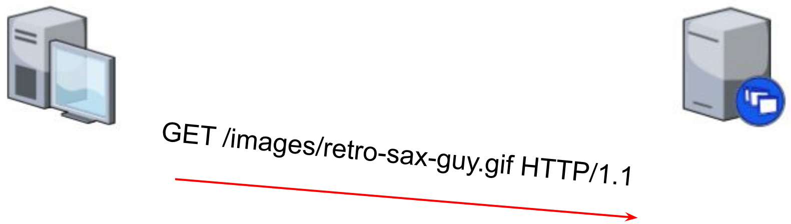
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since Time-based caching
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching



- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since Time-based caching
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

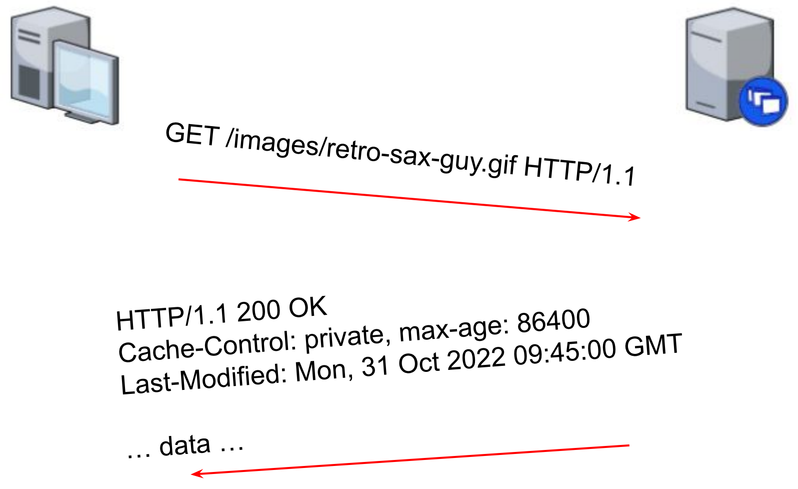
- Information about caching of resources provided in HTTP headers, e.g.

  Used to enable caching for resource and, optionally, to set an age for a resource

  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

  Time-based caching

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT

… data …
```

https://www.rfc-editor.org/rfc/rfc9111.html

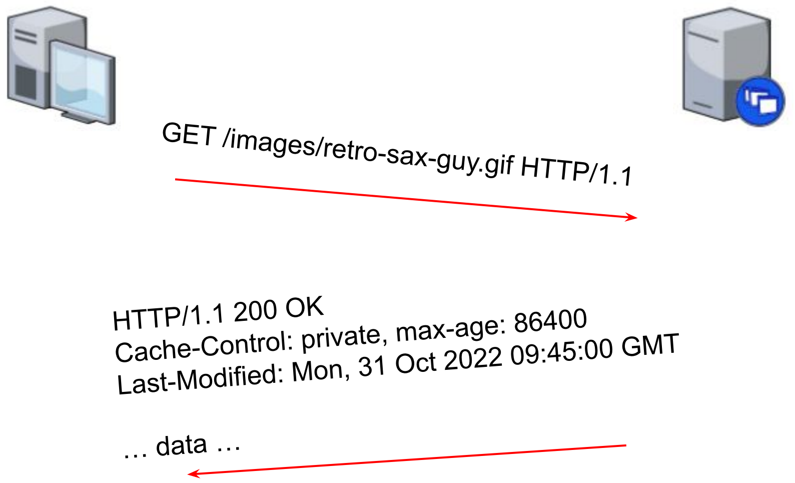
Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT

… data …
```

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

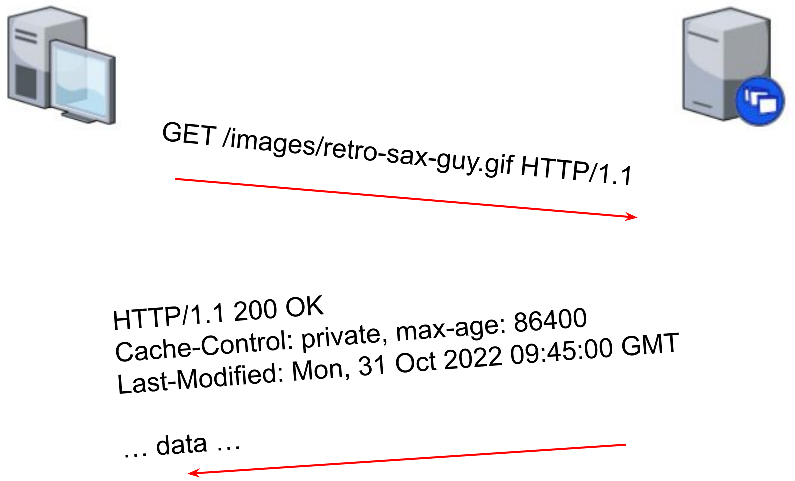
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
    - Cache-Control
    - Last-Modified → If-Modified-Since
    - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT

… data …
```

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

But, it needs to be retrieved after that.

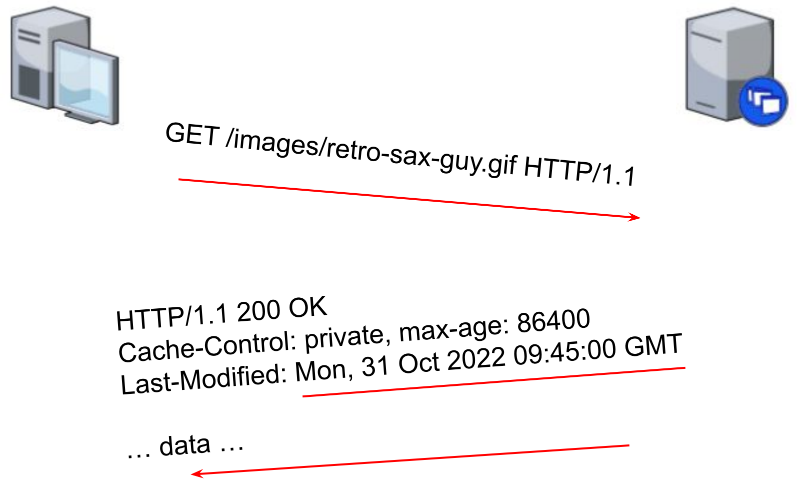
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT
```

```
… data …
```

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

But, it needs to be retrieved after that.

Ask for a new version of the resource only if it has not been modified since Last-Modified.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

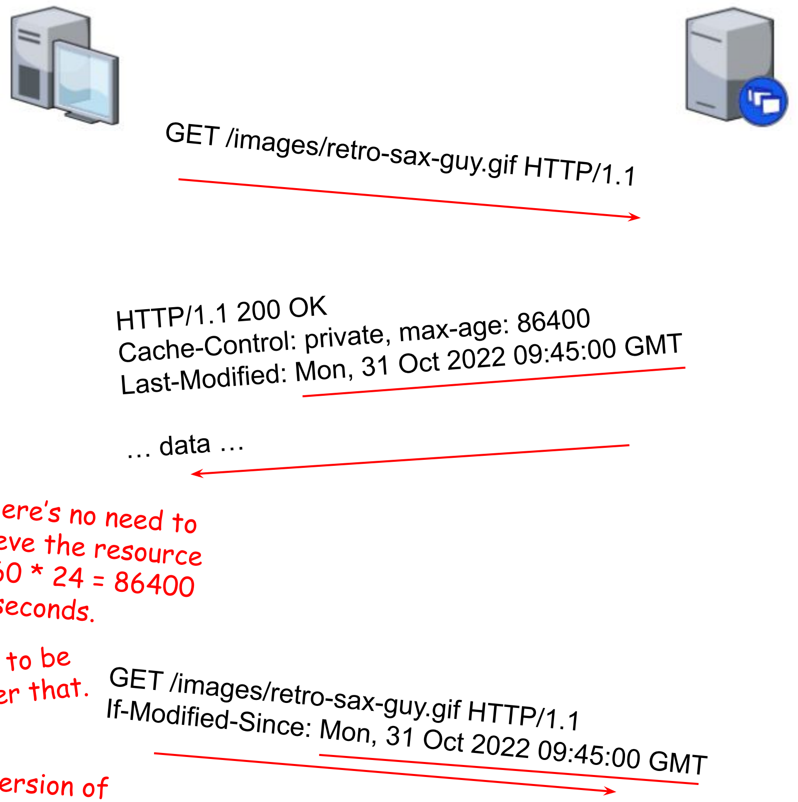
Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

But, it needs to be retrieved after that.

Ask for a new version of the resource only if it has not been modified since Last-Modified.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT

… data …
```

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-Modified-Since: Mon, 31 Oct 2022 09:45:00 GMT
```

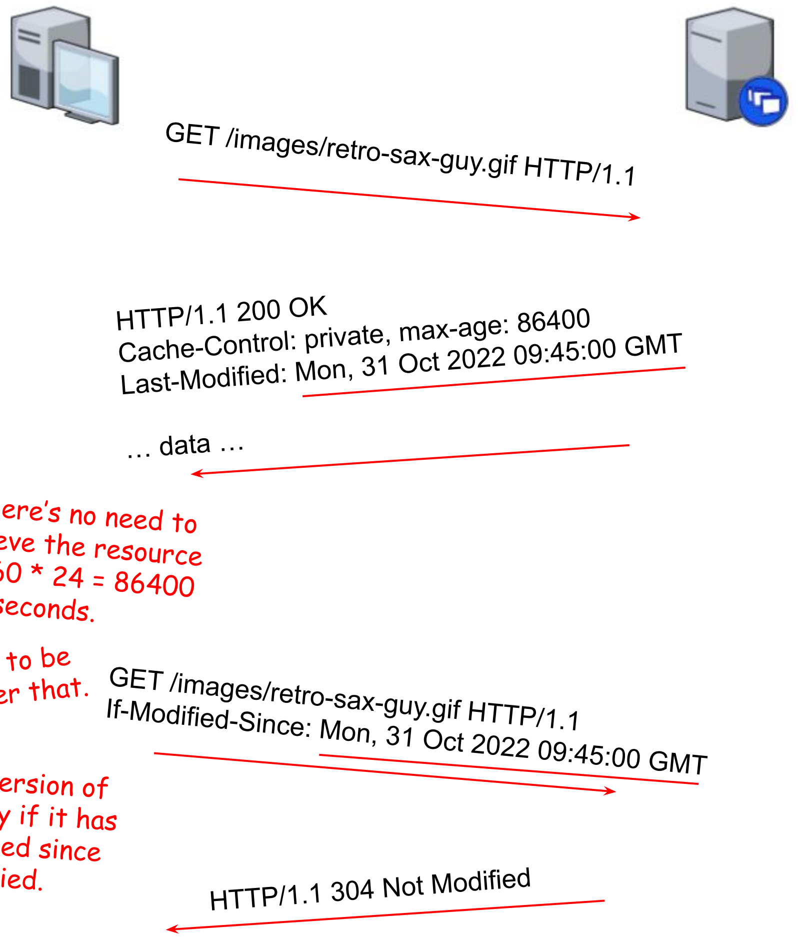
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT
```

```
… data …
```

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

But, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-Modified-Since: Mon, 31 Oct 2022 09:45:00 GMT
```

Ask for a new version of the resource only if it has not been modified since Last-Modified.

```
HTTP/1.1 304 Not Modified
```

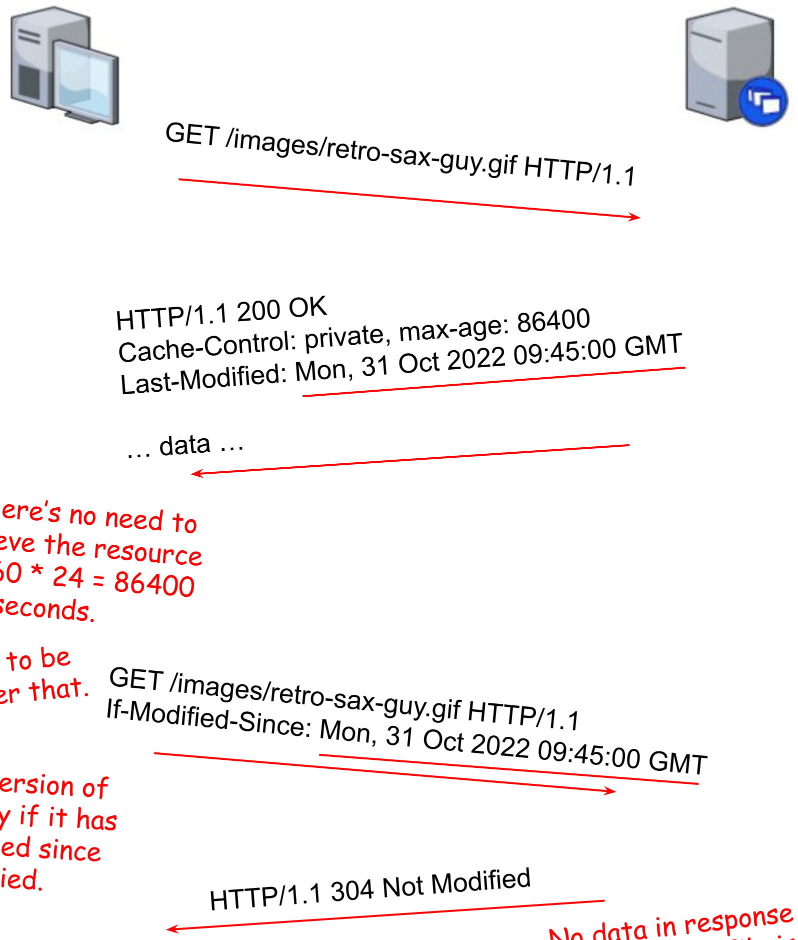
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
Last-Modified: Mon, 31 Oct 2022 09:45:00 GMT
```

```
… data …
```

Now, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

But, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-Modified-Since: Mon, 31 Oct 2022 09:45:00 GMT
```

Ask for a new version of the resource only if it has not been modified since Last-Modified.

```
HTTP/1.1 304 Not Modified
```

No data in response if the cached file is still valid.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.

  Used to enable caching for resource and, optionally, to set an age for a resource

  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

  Time-based caching

- Headers received in HTTP responses and sent in HTTP requests.

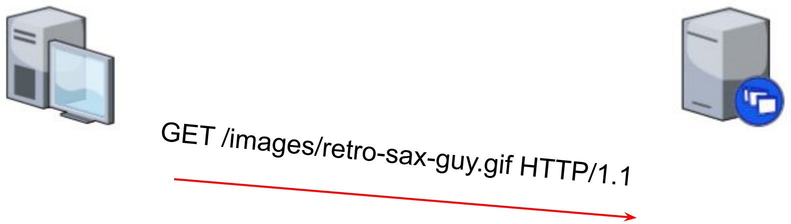
https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since Time-based caching
  - ETag → If-None-Matches Content-based caching
- Headers received in HTTP responses and sent in HTTP requests.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching



- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control Used to enable caching for resource and, optionally, to set an age for a resource
  - Last-Modified → If-Modified-Since Time-based caching
  - ETag → If-None-Matches Content-based caching
- Headers received in HTTP responses and sent in HTTP requests.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

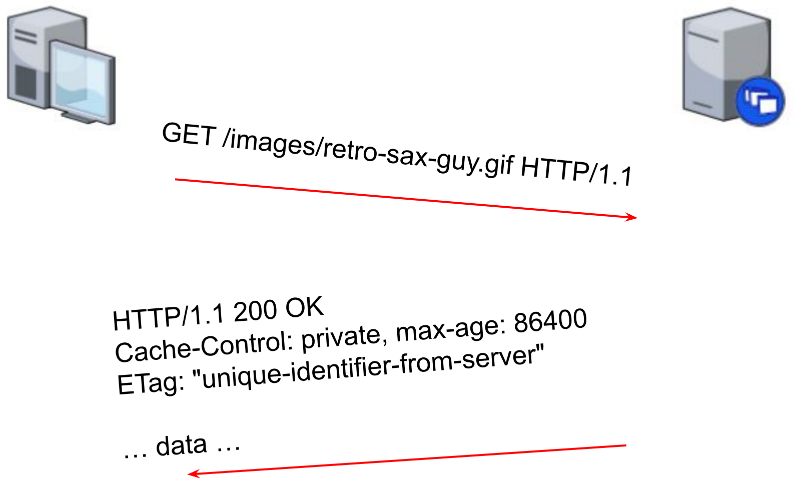
- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"

… data …
```

https://www.rfc-editor.org/rfc/rfc9111.html

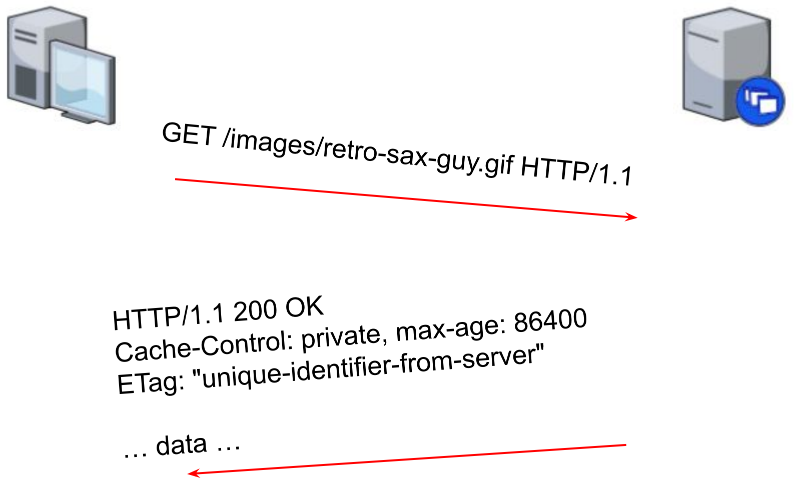
Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching


GET /images/retro-sax-guy.gif HTTP/1.1
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

https://www.rfc-editor.org/rfc/rfc9111.html

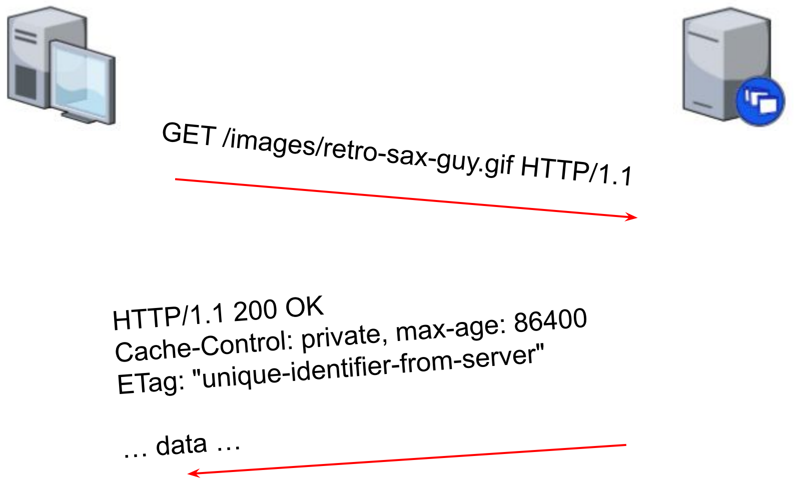
Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
```

… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

https://www.rfc-editor.org/rfc/rfc9111.html

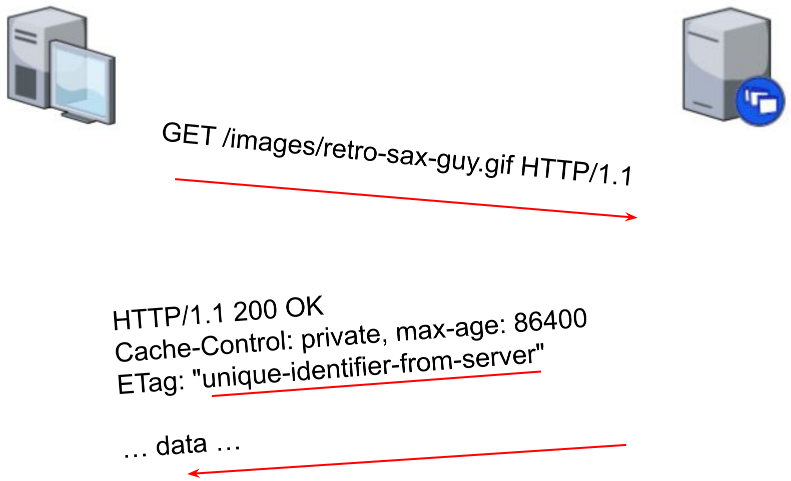
Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching


GET /images/retro-sax-guy.gif HTTP/1.1
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

Ask for a new version of the resource only if its unique identifier does not match ETag.

https://www.rfc-editor.org/rfc/rfc9111.html

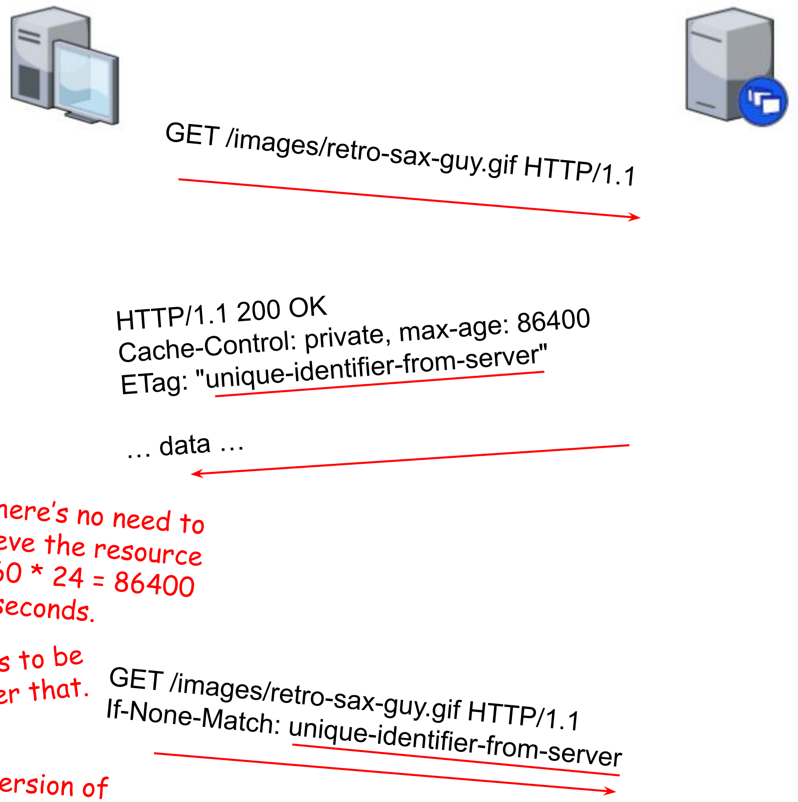
Client-side Caching

- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches
- Headers received in HTTP responses and sent in HTTP requests.

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
```

… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-None-Match: unique-identifier-from-server
```

Ask for a new version of the resource only if its unique identifier does not match ETag.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

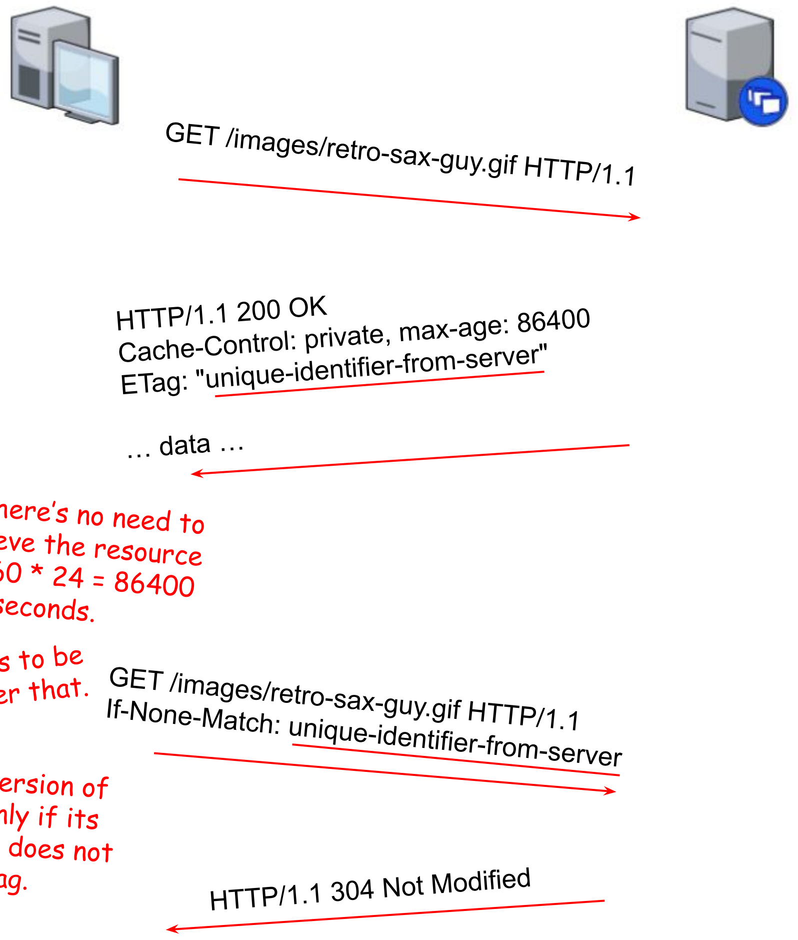
- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
```

… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-None-Match: unique-identifier-from-server
```

Ask for a new version of the resource only if its unique identifier does not match ETag.

```
HTTP/1.1 304 Not Modified
```

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

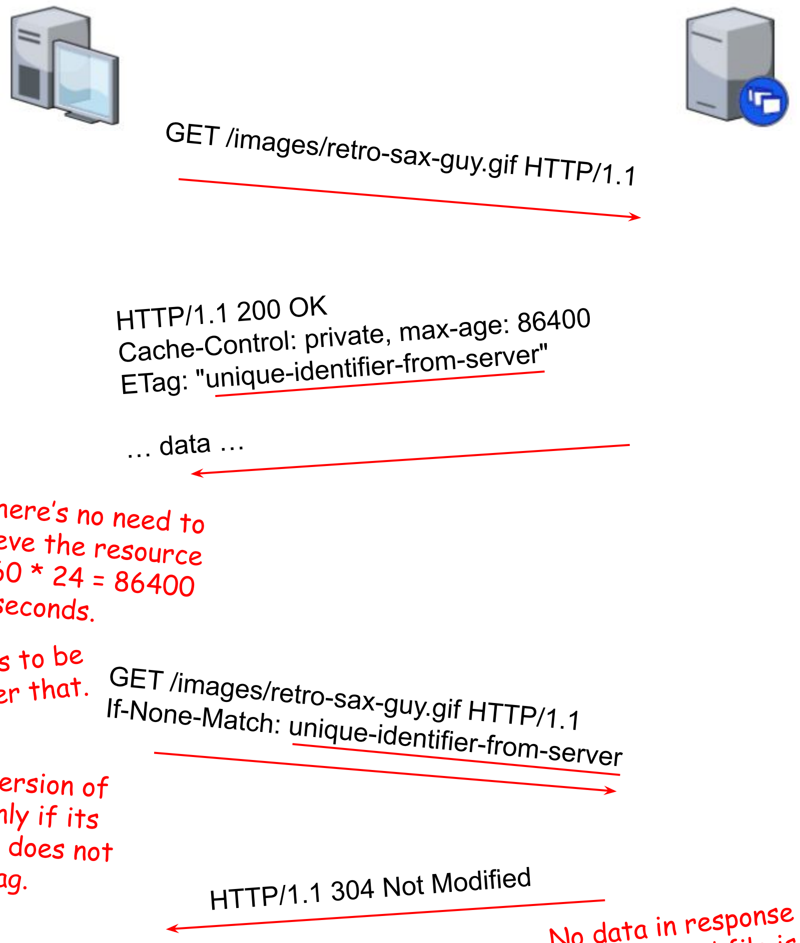
- Information about caching of resources provided in HTTP headers, e.g.
    - Cache-Control
    - Last-Modified → If-Modified-Since
    - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching

- Headers received in HTTP responses and sent in HTTP requests.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
```

… data …

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-None-Match: unique-identifier-from-server
```

Ask for a new version of the resource only if its unique identifier does not match ETag.

```
HTTP/1.1 304 Not Modified
```

No data in response if the cached file is still valid.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching

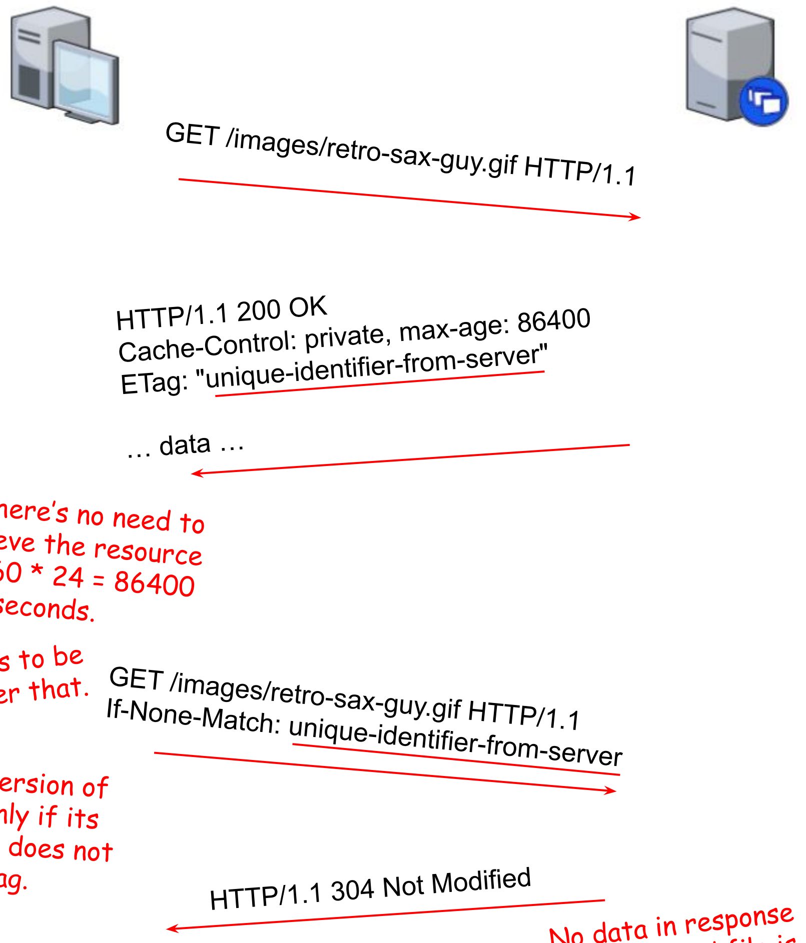
- Information about caching of resources provided in HTTP headers, e.g.
  - Cache-Control
  - Last-Modified → If-Modified-Since
  - ETag → If-None-Matches

Used to enable caching for resource and, optionally, to set an age for a resource

Time-based caching

Content-based caching

- Headers received in HTTP responses and sent in HTTP requests.
- Recently also Web Cache API that can, e.g., be used to cache JavaScript request responses. See RFC9111.



```
GET /images/retro-sax-guy.gif HTTP/1.1
```

```
HTTP/1.1 200 OK
Cache-Control: private, max-age: 86400
ETag: "unique-identifier-from-server"
```

... data ...

Again, there's no need to re-retrieve the resource in 60 * 60 * 24 = 86400 seconds.

Again, it needs to be retrieved after that.

```
GET /images/retro-sax-guy.gif HTTP/1.1
If-None-Match: unique-identifier-from-server
```

Ask for a new version of the resource only if its unique identifier does not match ETag.

```
HTTP/1.1 304 Not Modified
```

No data in response if the cached file is still valid.

https://www.rfc-editor.org/rfc/rfc9111.html

Client-side Caching: Example

Server-side Caching

Server-side Caching

- Many approaches, many ways to implement – typically on a need basis:
  - Caching static resources from disk
  - Caching resources retrieved from database
  - Caching dynamic content
  - Caching resources on the server
  - Caching resources on a separate server, e.g. a dedicated cache server, load balancer, reverse proxy, …

Server-side Caching

- Many approaches, many ways to implement – typically on a need basis:
  - Caching static resources from disk
  - Caching resources retrieved from database
  - Caching dynamic content
  - Caching resources on the server
  - Caching resources on a separate server, e.g. a dedicated cache server, load balancer, reverse proxy, …

- Previously also caching on proxy servers. With increasing use of HTTPS, this is used less often (and is less often useful).

Server-side Caching

- Many approaches, many ways to implement – typically on a need basis:
    - Caching static resources from disk
    - Caching resources retrieved from database
    - Caching dynamic content
    - Caching resources on the server
    - Caching resources on a separate server, e.g. a dedicated cache server, load balancer, reverse proxy, …

Note: proxy servers and reverse proxy are different

- Previously also caching on proxy servers. With increasing use of HTTPS, this is used less often (and is less often useful).

Server-side Caching

- Many approaches, many ways to implement – typically on a need basis:
  - Caching static resources from disk
  - Caching resources retrieved from database
  - Caching dynamic content
  - Caching resources on the server
  - Caching resources on a separate server, e.g. a dedicated cache server, load balancer, reverse proxy, …

Note: proxy servers and reverse proxy are different

- Previously also caching on proxy servers. With increasing use of HTTPS, this is used less often (and is less often useful).

Less often used, well.. except for..

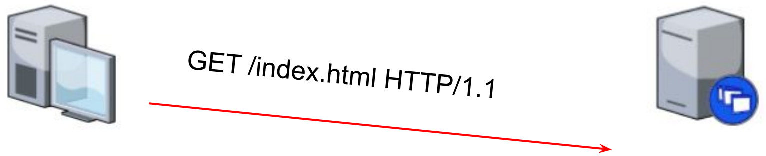
Content-Delivery Networks

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

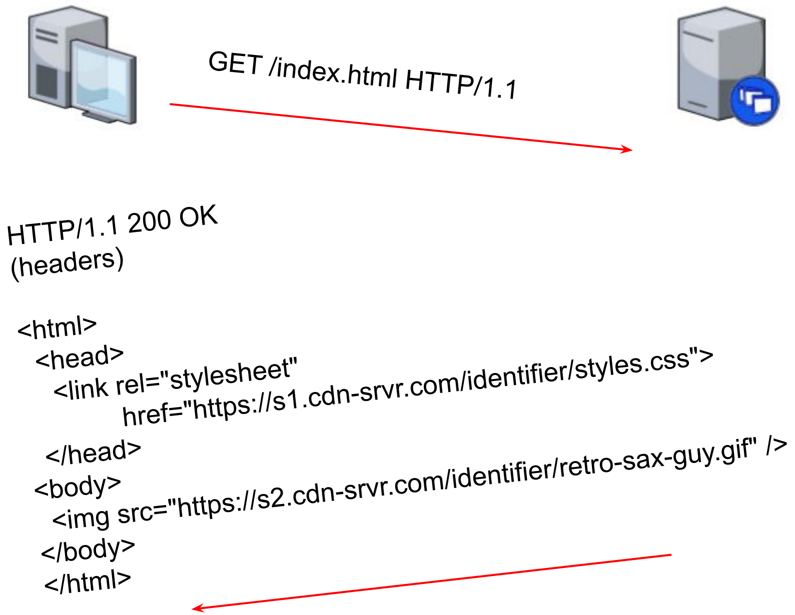
Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).



Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

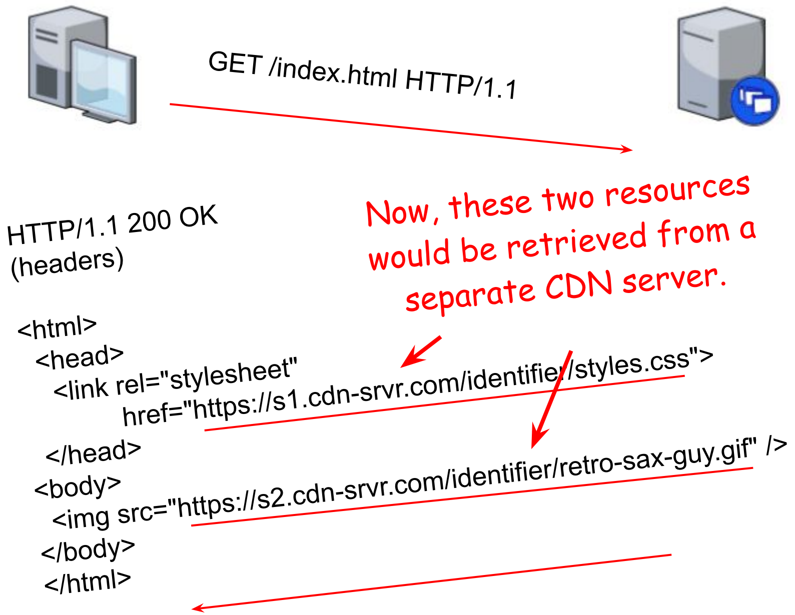


```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).



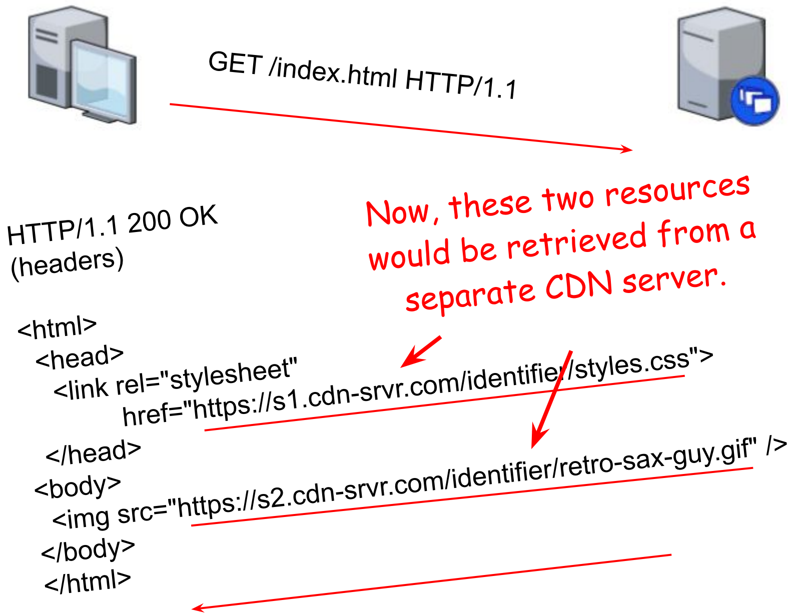
GET /index.html HTTP/1.1

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

Now, these two resources would be retrieved from a separate CDN server.

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).
- Multiple servers with different addresses.



```
GET /index.html HTTP/1.1
```

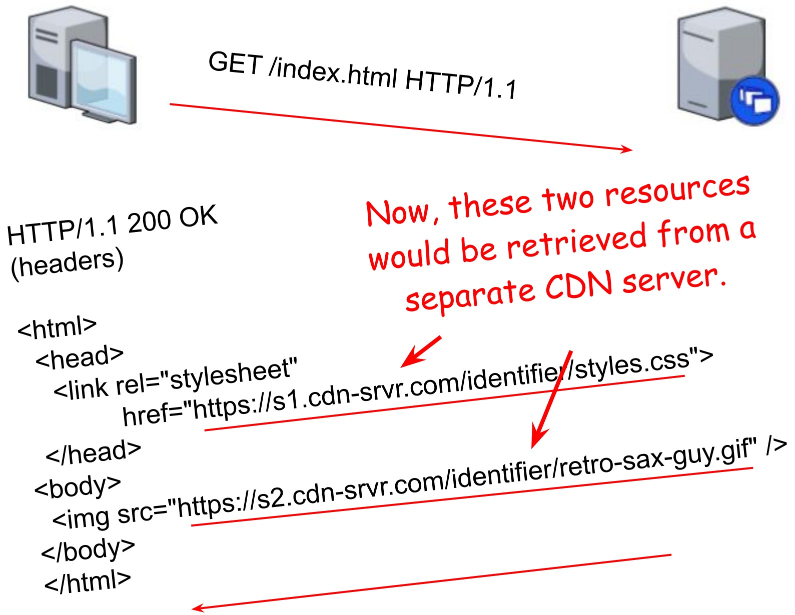
```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

Now, these two resources would be retrieved from a separate CDN server.

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

- Multiple servers with different addresses.
  - Remember that browsers can open up a limited number of concurrent requests per server.
  - With multiple servers, browsers can open up the limited number of concurrent requests to each server.


GET /index.html HTTP/1.1
Now, these two resources would be retrieved from a separate CDN server.

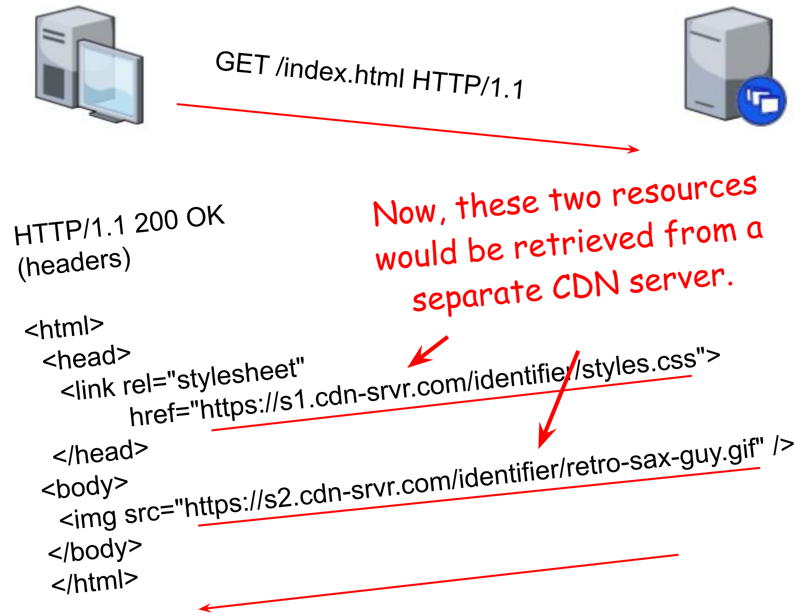
```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

- Multiple servers with different addresses.
  - Remember that browsers can open up a limited number of concurrent requests per server.
  - With multiple servers, browsers can open up the limited number of concurrent requests to each server.

- CDN servers typically geographically distributed.



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

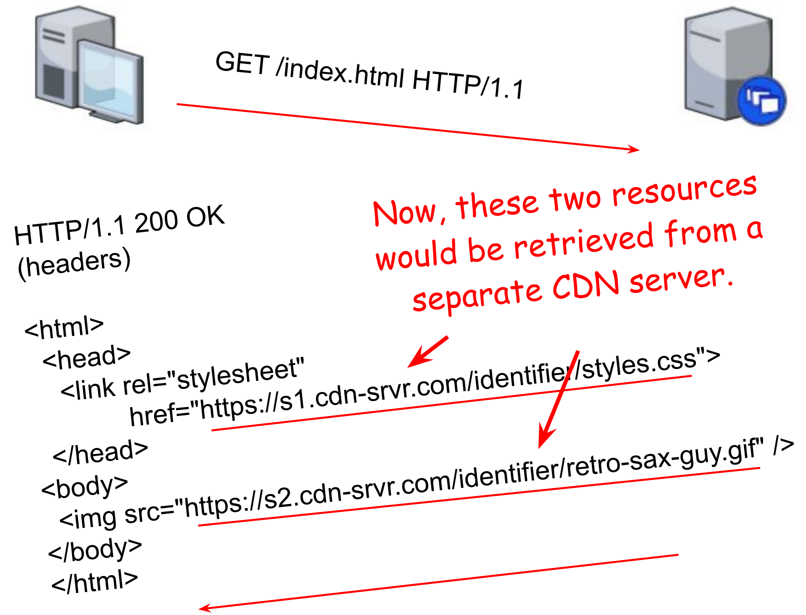
Now, these two resources would be retrieved from a separate CDN server.

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

- Multiple servers with different addresses.
  - Remember that browsers can open up a limited number of concurrent requests per server.
  - With multiple servers, browsers can open up the limited number of concurrent requests to each server.

- CDN servers typically geographically distributed.
  - Requests to CDN routed to the closest servers.



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

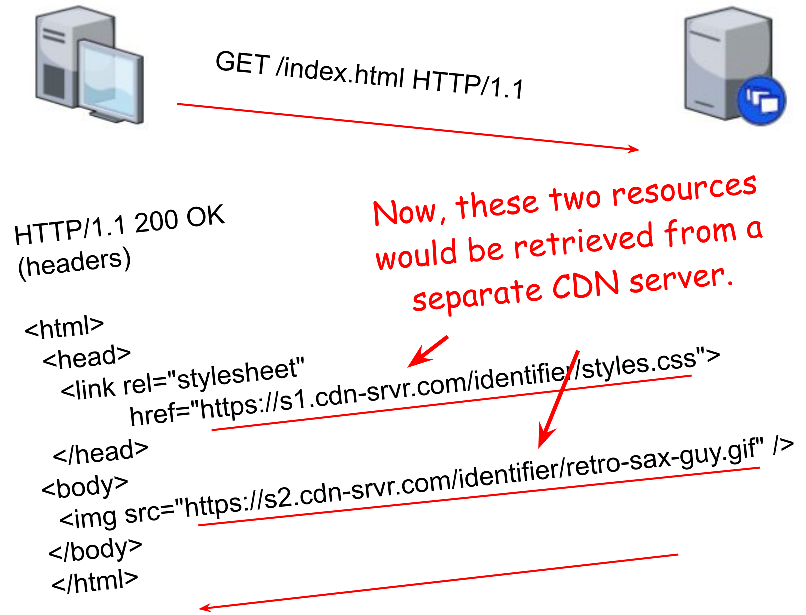
Now, these two resources would be retrieved from a separate CDN server.

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).

- Multiple servers with different addresses.
  - Remember that browsers can open up a limited number of concurrent requests per server.
  - With multiple servers, browsers can open up the limited number of concurrent requests to each server.

- CDN servers typically geographically distributed.
  - Requests to CDN routed to the closest servers.



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

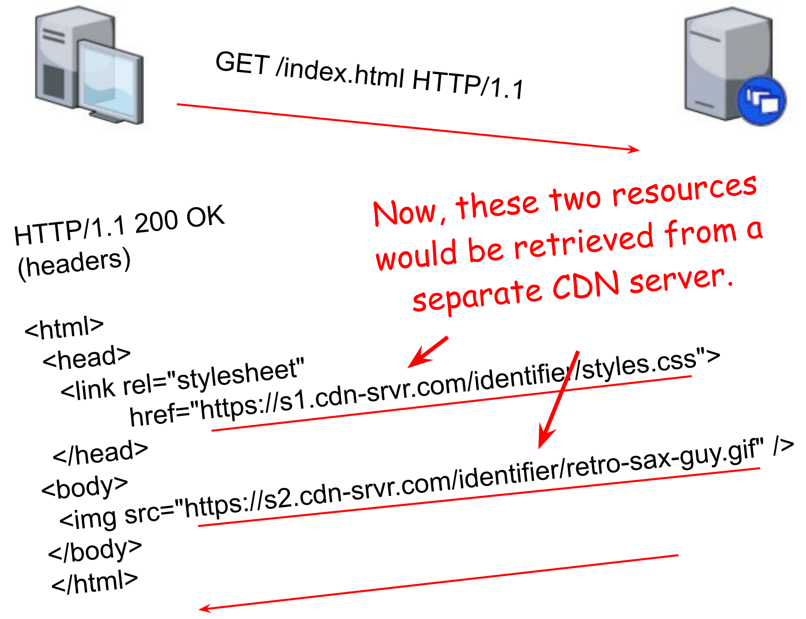
Now, these two resources would be retrieved from a separate CDN server.

"The edge"

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).
- Multiple servers with different addresses.
  - Remember that browsers can open up a limited number of concurrent requests per server.
  - With multiple servers, browsers can open up the limited number of concurrent requests to each server.
- CDN servers typically geographically distributed.
  - Requests to CDN routed to the closest servers.
- Naturally, CDNs also use HTTP headers allowing client-side caching.

"The edge"



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

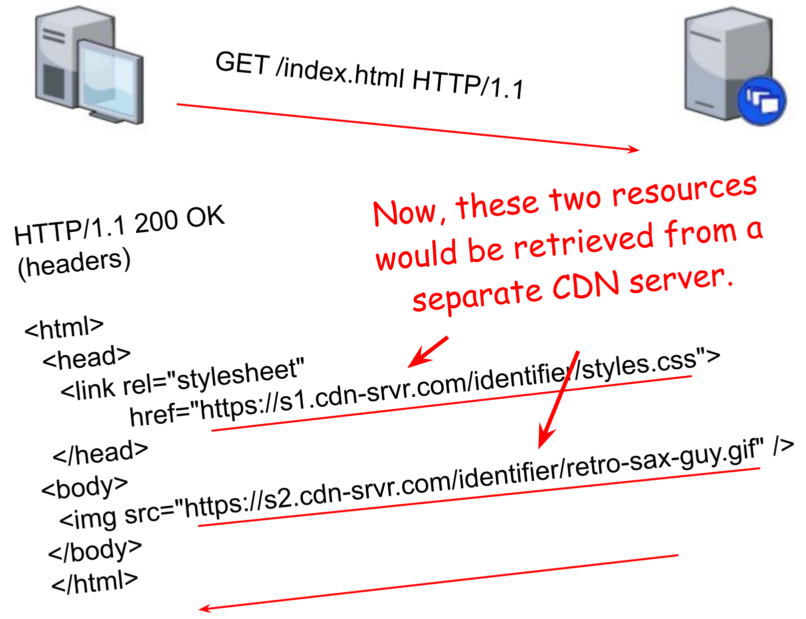
Now, these two resources would be retrieved from a separate CDN server.

Can also be used like a proxy server

Content-Delivery Networks

- A specific shared cache; cacheable data (e.g. static content) is distributed across a body of servers – a Content-Delivery Network (CDN).
- Multiple servers with different addresses.
    - Remember that browsers can open up a limited number of concurrent requests per server.
    - With multiple servers, browsers can open up the limited number of concurrent requests to each server.
- CDN servers typically geographically distributed.
    - Requests to CDN routed to the closest servers.
- Naturally, CDNs also use HTTP headers allowing client-side caching.

"The edge"



```
GET /index.html HTTP/1.1
```

```
HTTP/1.1 200 OK
(headers)

<html>
  <head>
    <link rel="stylesheet"
          href="https://s1.cdn-srvr.com/identifier/styles.css">
  </head>
  <body>
    <img src="https://s2.cdn-srvr.com/identifier/retro-sax-guy.gif" />
  </body>
</html>
```

Now, these two resources would be retrieved from a separate CDN server.

Content-Delivery Networks

Content-Delivery Networks

- Can lead to significantly fewer requests to the “main servers” → Requests for resources distributed over CDN servers.

Content-Delivery Networks

- Can lead to significantly fewer requests to the “main servers” → Requests for resources distributed over CDN servers.
- Can lead to faster web application loading times → Static resources loaded from the edge.

Content-Delivery Networks

- Can lead to significantly fewer requests to the “main servers” → Requests for resources distributed over CDN servers.
- Can lead to faster web application loading times → Static resources loaded from the edge.
- Resources on CDN servers available even if the main servers are not available.

Any downsides with Caching and CDNs?

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.

"There are only two hard problems in Computer Science: cache invalidation and naming things." - Phil Karlton

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.
- Comes with some costs (fees for CDN service providers, set up effort, …).

"There are only two hard problems in Computer Science: cache invalidation and naming things." – Phil Karlton

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.
- Comes with some costs (fees for CDN service providers, set up effort, …).

"There are only two hard problems in Computer Science: cache invalidation and naming things." – Phil Karlton

See e.g. https://cloud.google.com/cdn/pricing

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.

"There are only two hard problems in Computer Science: cache invalidation and naming things." – Phil Karlton

- Comes with some costs (fees for CDN service providers, set up effort, …).

See e.g. https://cloud.google.com/cdn/pricing

- Also potential cyber-security issues, e.g. integrity of second-hand content; using CDNs for DOS attacks (for latter, see RFC8586).

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.
- Comes with some costs (fees for CDN service providers, set up effort, …).
- Also potential cyber-security issues, e.g. integrity of second-hand content; using CDNs for DOS attacks (for latter, see RFC8586).

"There are only two hard problems in Computer Science: cache invalidation and naming things." – Phil Karlton

See e.g. https://cloud.google.com/cdn/pricing

https://www.rfc-editor.org/rfc/rfc8586.html

Any downsides with Caching and CDNs?

- Stale cache → invalid content shown.
- Comes with some costs (fees for CDN service providers, set up effort, …).
- Also potential cyber-security issues, e.g. integrity of second-hand content; using CDNs for DOS attacks (for latter, see RFC8586).

"There are only two hard problems in Computer Science: cache invalidation and naming things." – Phil Karlton

See e.g. https://cloud.google.com/cdn/pricing

https://dmsec.io/hacking-thousands-of-websites-via-third-party-javascript-libraries/

https://www.rfc-editor.org/rfc/rfc8586.html

Example: Cloudflare CDN

https://cloudflare.com

Starting with Client-Side Web Development

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.

- Often realized using HTML, CSS and JavaScript (or with languages that compile to one or more of these).

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.
- Often realized using HTML, CSS and JavaScript (or with languages that compile to one or more of these).

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

E.g. Dart and Flutter (used in CS-E4270 Device-Agnostic Design)

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.
- Often realized using HTML, CSS and JavaScript (or with languages that compile to one or more of these).
- Plenty of tools and frameworks for the purpose, e.g. Angular, Ember, Svelte, React, VueJs…

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

E.g. Dart and Flutter (used in CS-E4270 Device-Agnostic Design)

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.

- Often realized using HTML, CSS and JavaScript (or with languages that compile to one or more of these).

- Plenty of tools and frameworks for the purpose, e.g. Angular, Ember, Svelte, React, VueJs…

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

E.g. Dart and Flutter (used in CS-E4270 Device-Agnostic Design)

New ones popping up every now and then.

Client-Side Web Development

- Client-side applications run on the client (e.g. the browser) and do not necessarily interact with a server.
- Often realized using HTML, CSS and JavaScript (or with languages that compile to one or more of these).
- Plenty of tools and frameworks for the purpose, e.g. Angular, Ember, Svelte, React, VueJs…
- The field has evolved significantly during the last decade and seems to still be evolving.

Full stack development refers to working on both client-side apps (frontend) and server-side apps (backend).

E.g. Dart and Flutter (used in CS-E4270 Device-Agnostic Design)

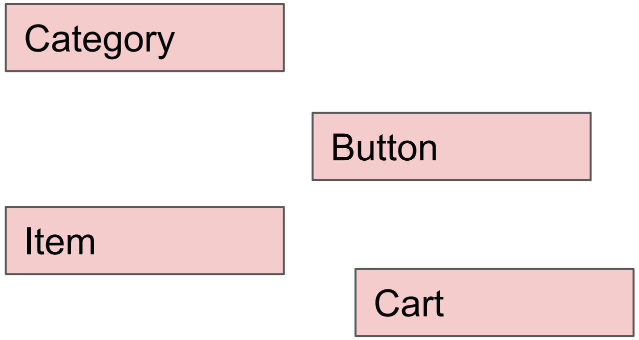
New ones popping up every now and then.

Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.

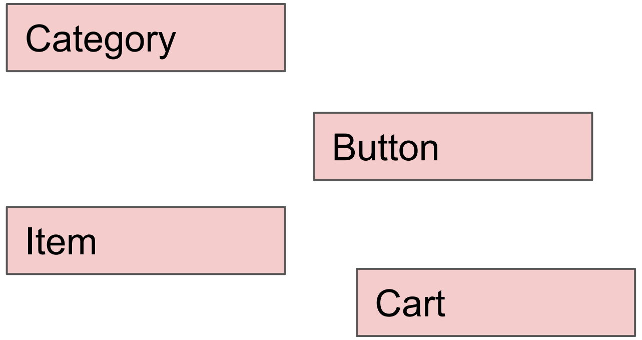
Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.



Client-Side Web Development

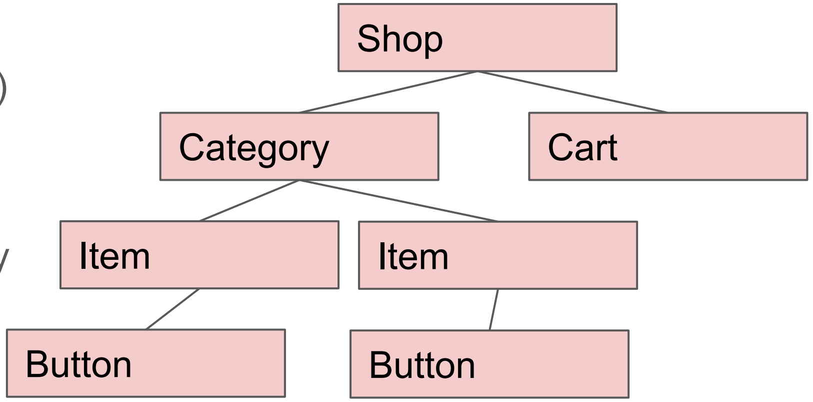
- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.
- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.



Client-Side Web Development

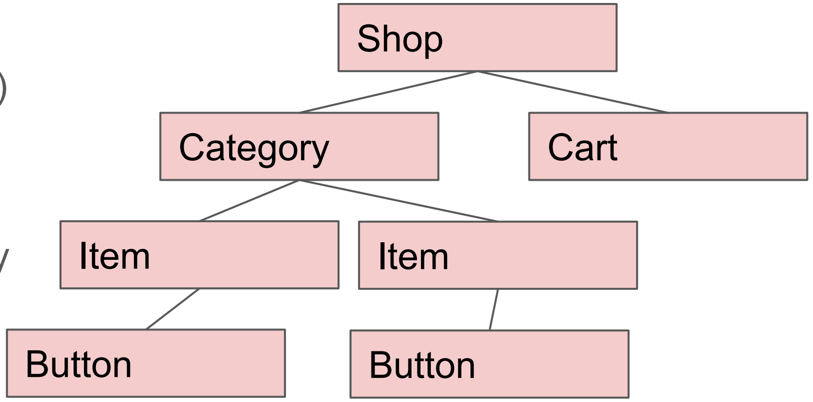
- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.

- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.



Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.
- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.

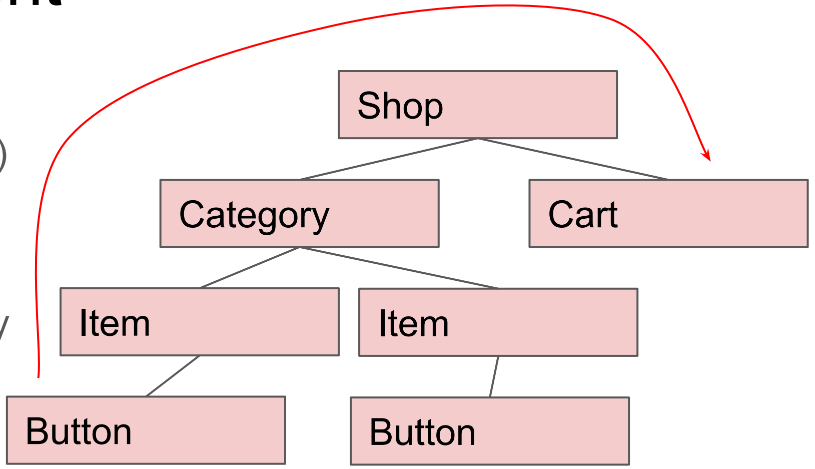


State management libraries another area with continuous development - looking for developer-friendly approaches for maintaining state across the application.

E.g. how to add an item to the cart when the button is clicked.

Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.
- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.

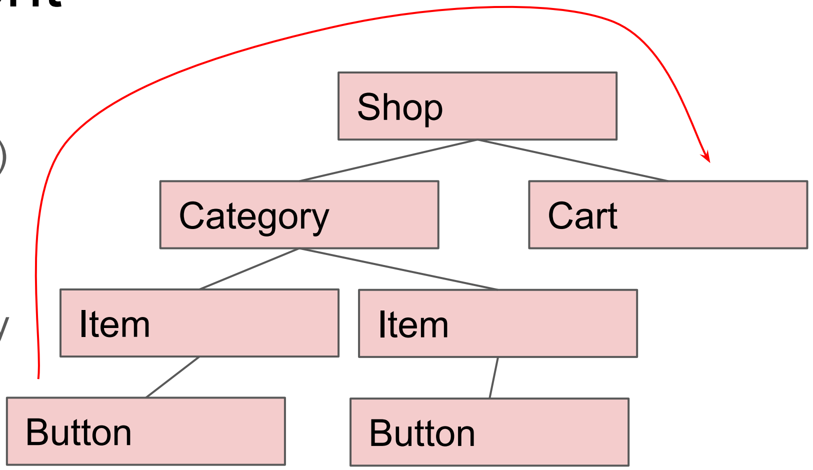


State management libraries another area with continuous development - looking for developer-friendly approaches for maintaining state across the application.

Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.
- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.
- When an application is released, the application is compiled into a bundle consisting of HTML, CSS and JavaScript, which can be loaded by browsers.

E.g. how to add an item to the cart when the button is clicked.

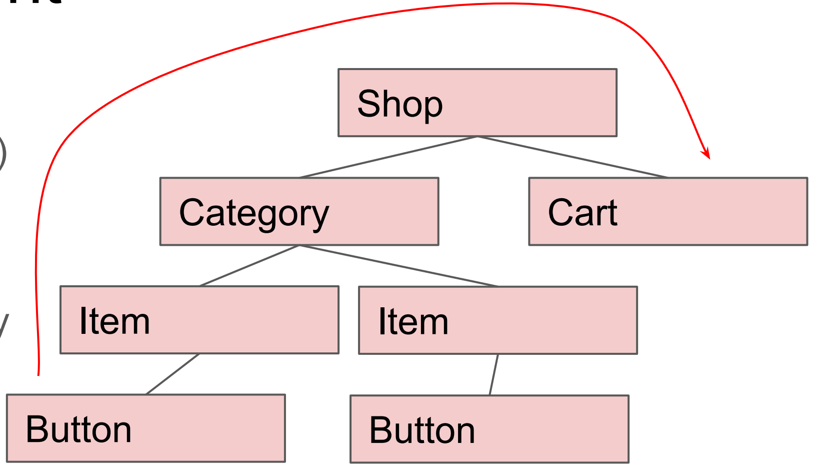


State management libraries another area with continuous development - looking for developer-friendly approaches for maintaining state across the application.

Client-Side Web Development

- Many of the popular client-side frameworks are essentially libraries for building (reusable) components.
- Applications are formed by building and nesting components in a meaningful (typically tree-like) structure, which leads to the client-side application.
- When an application is released, the application is compiled into a bundle consisting of HTML, CSS and JavaScript, which can be loaded by browsers.
- The bundle is then added to a server from where it can be accessed.

E.g. how to add an item to the cart when the button is clicked.



State management libraries another area with continuous development - looking for developer-friendly approaches for maintaining state across the application.

A Brief Peek at Svelte

https://kit.svelte.dev/