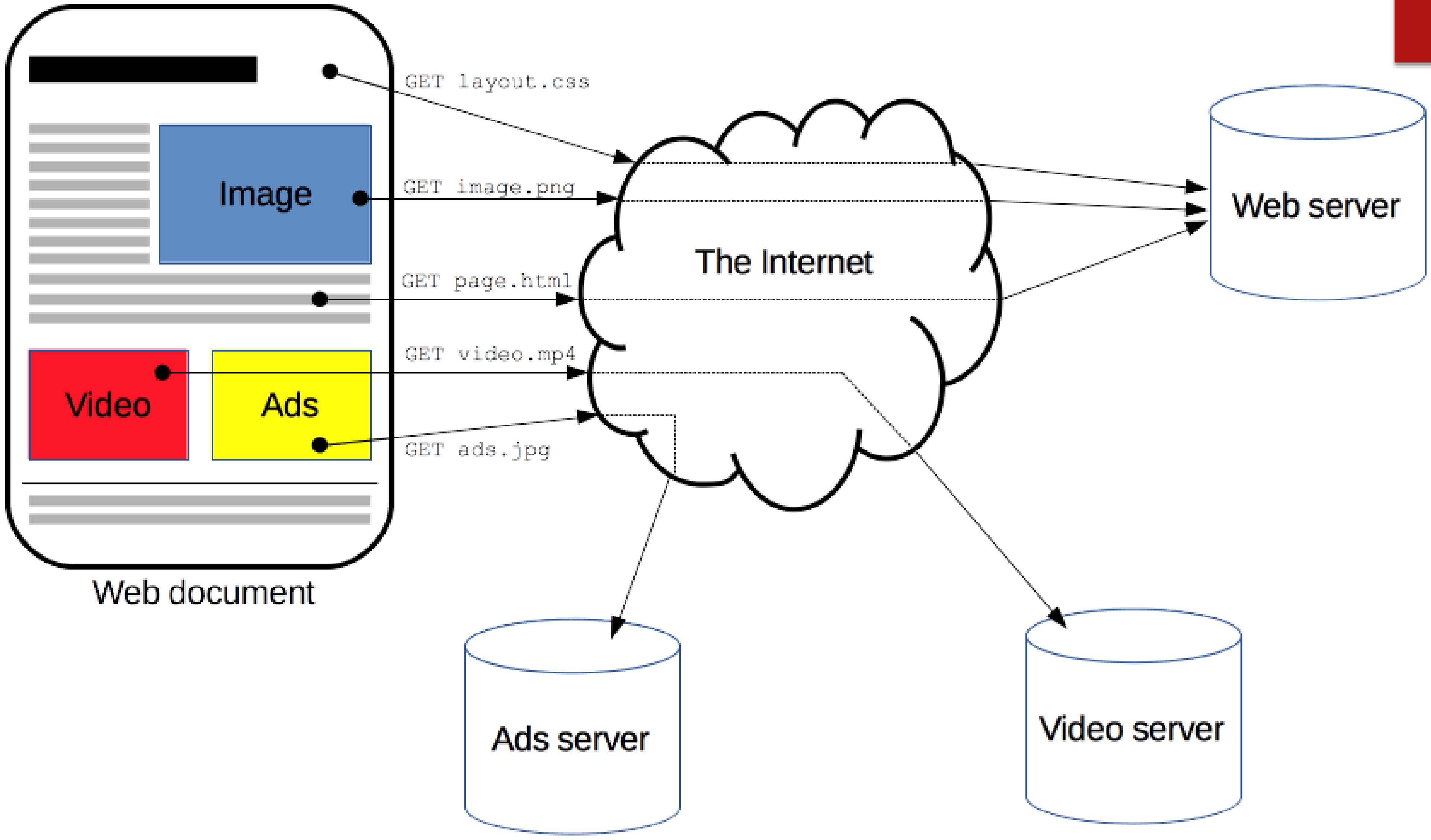
Web and HTTP Fundamentals

MOHAMMED SAFADI

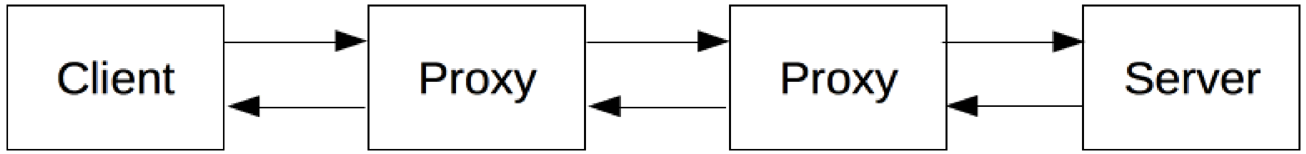
WEB SYSTEMS CONSULTANT

HTTP

- ***Hypertext Transfer Protocol (HTTP)*** is an application-layer protocol for transmitting hypermedia documents, such as HTML.
- Designed for communication between web browsers and web servers, but it can also be used for other purposes.
- HTTP follows a classical client-server model, with a client opening a connection to make a request, then waiting until it receives a response.
- HTTP is a stateless protocol, meaning that the server does not keep any data (state) between two requests.


GET layout.css
Image
GET image.png
The Internet
Web server
GET page.html
GET video.mp4
Video
Ads
GET ads.jpg
Web document
Ads server
Video server

Components of HTTP-based systems



Client: The user-agent

- The *user-agent* is any tool that acts on the behalf of the user. This role is primarily performed by the Web browser; other possibilities are programs used by engineers and Web developers to debug their applications.
- The browser is **always** the entity initiating the request. It is never the server (though some mechanisms have been added over the years to simulate server-initiated messages).

The Web Server

- On the opposite side of the communication channel, is the server, which *serves* the document as requested by the client. A server appears as only a single machine virtually: this is because it may actually be a collection of servers, sharing the load (load balancing) or a complex piece of software interrogating other computers (like cache, a DB server, or e-commerce servers), totally or partially generating the document on demand.

Proxies

- caching (the cache can be public or private, like the browser cache)
- filtering (like an antivirus scan or parental controls)
- load balancing (to allow multiple servers to serve the different requests)
- authentication (to control access to different resources)
- logging (allowing the storage of historical information)

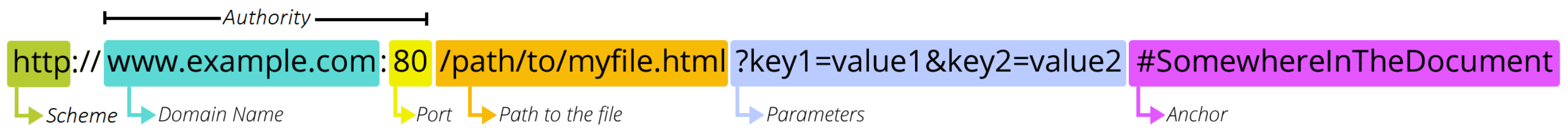
Identifying resources on the Web

- The target of an HTTP request is called a "resource", whose nature isn't defined further; it can be a document, a photo, or anything else. Each resource is identified by a Uniform Resource Identifier (URI) used throughout HTTP for identifying resources.

URLs

- ***URL*** is one of the key concepts of the Web. It is the mechanism used by browsers to retrieve any published resource on the web.
- **URL** stands for *Uniform Resource Locator*.
- A URL is nothing more than the address of a given unique resource on the Web.
- A URL is composed of different parts, some mandatory and others are optional

- `https://developer.mozilla.org`
- `https://developer.mozilla.org/en-US/docs/Learn/`
- `https://developer.mozilla.org/en-US/search?q=URL`

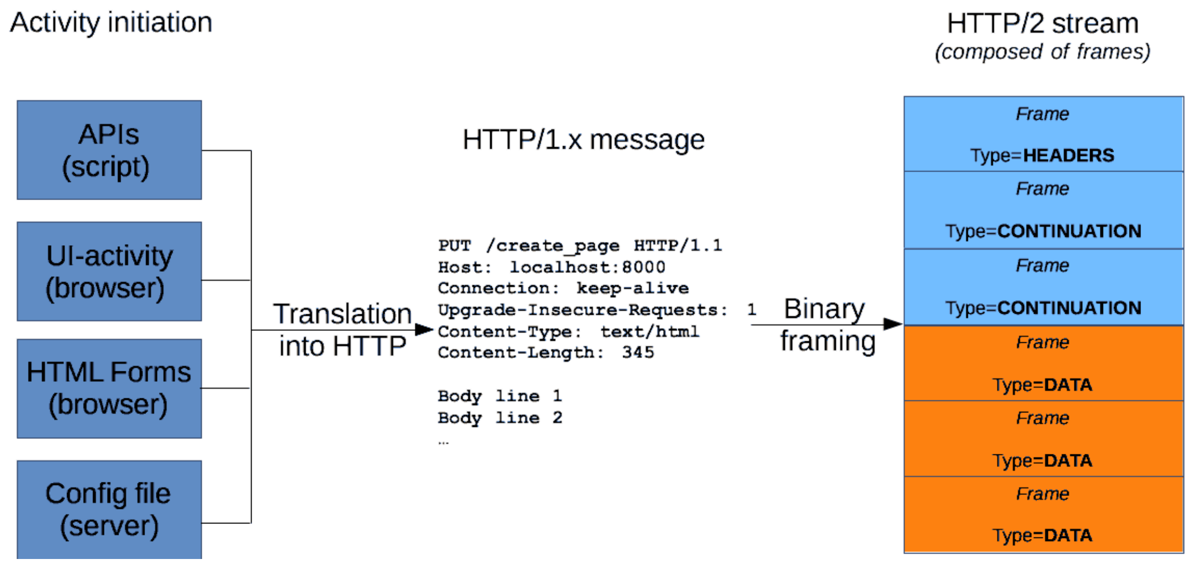

Authority
http://www.example.com:80/path/to/myfile.html?key1=value1&key2=value2#SomewhereInTheDocument
Scheme
Domain Name
Port
Path to the file
Parameters
Anchor

Anatomy of a URL

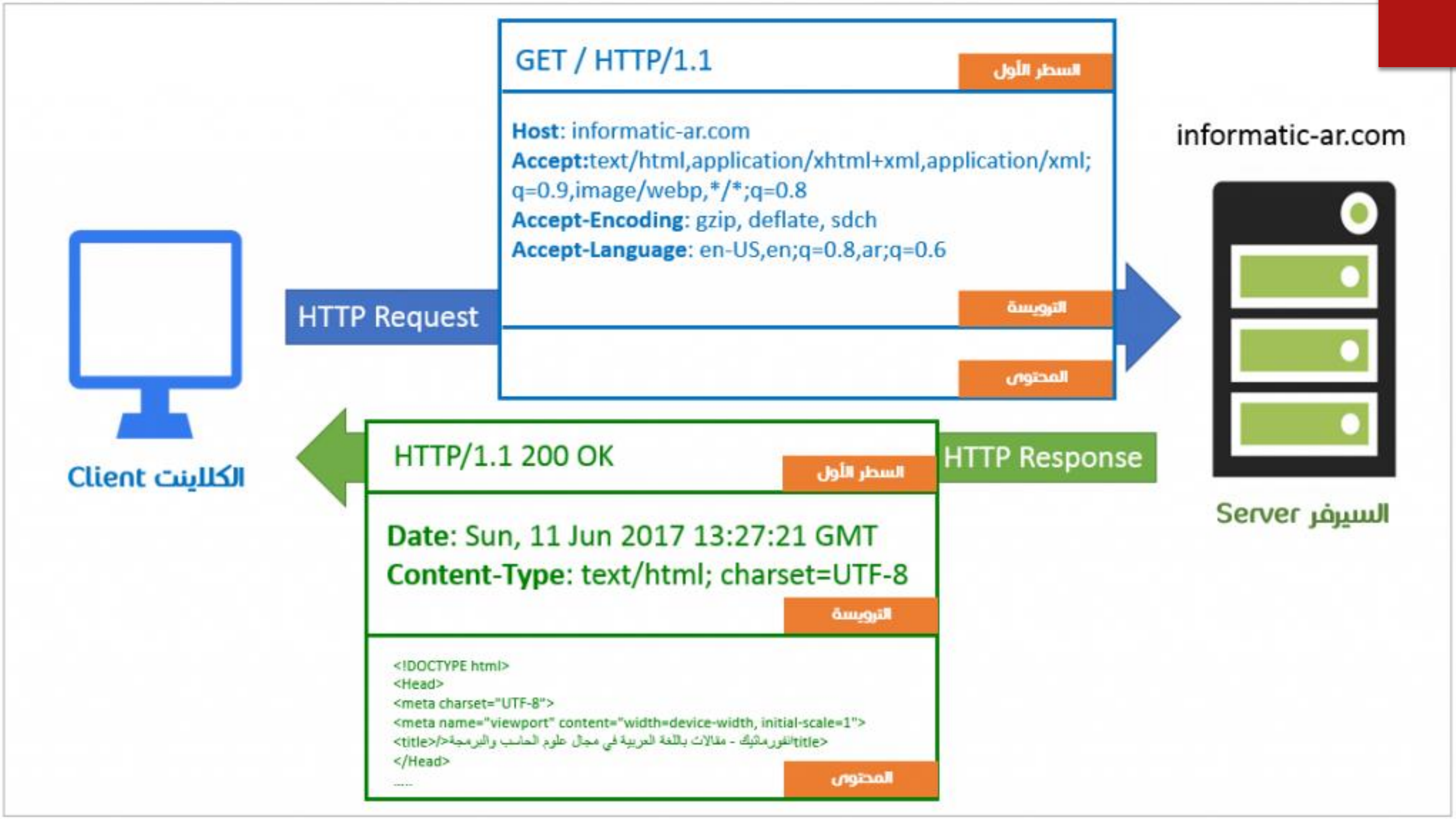
- Scheme: indicates the protocol that the browser must use to request the resource.
- *Authority*: If present the authority includes both the *domain* (e.g. `www.example.com`) and the *port* (`80`), separated by a colon.
- Path to resource: In the early days of the Web, a path like this represented a physical file location on the Web server. Nowadays, it is mostly an abstraction handled by Web servers without any physical reality.
- Parameters: list of key/value pairs separated with the `&` symbol. The Web server can use those parameters to do extra stuff before returning the resource
- Anchor: is an anchor to another part of the resource itself

HTTP Messages

- HTTP messages are how data is exchanged between a server and a client. There are two types of messages: requests sent by the client to trigger an action on the server, and responses, the answer from the server.



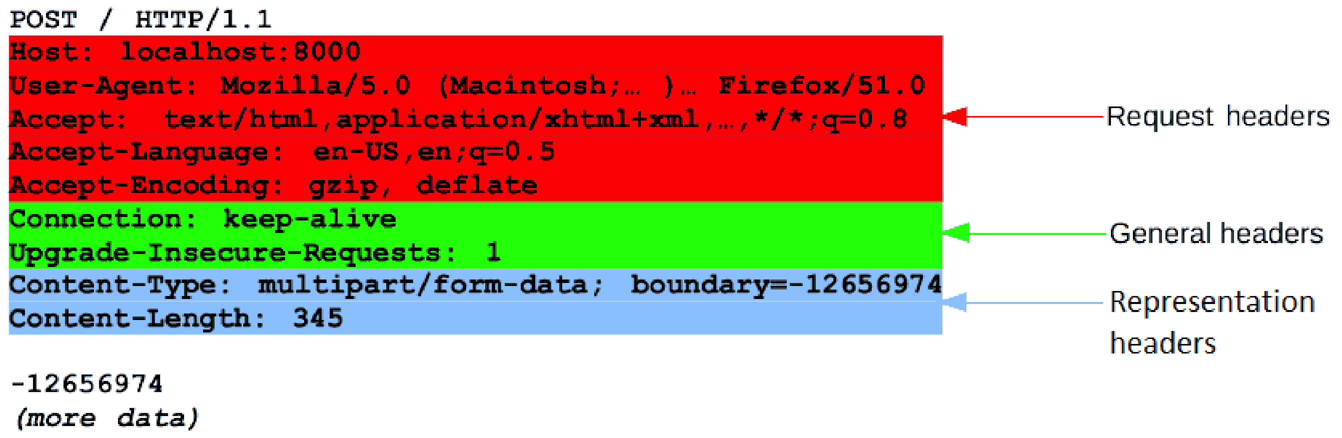
```
PUT /create_page HTTP/1.1
Host: localhost:8000
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Content-Type: text/html
Content-Length: 345

Body line 1
Body line 2
...
```



```
GET / HTTP/1.1

Host: informatic-ar.com
Accept:text/html,application/xhtml+xml,application/xml;
q=0.9,image/webp,*/*;q=0.8
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8,ar;q=0.6
```

```
HTTP/1.1 200 OK

Date: Sun, 11 Jun 2017 13:27:21 GMT
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<Head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>انفورماتيك - مقالات باللغة العربية في مجال علوم الحاسب والبرمجة</title>
</Head>
-----
```

HTTP Request



```
POST / HTTP/1.1
Host: localhost:8000
User-Agent: Mozilla/5.0 (Macintosh;… )… Firefox/51.0
Accept:  text/html,application/xhtml+xml,…,*/*;q=0.8
Accept-Language:  en-US,en;q=0.5
Accept-Encoding:  gzip, deflate
Connection:  keep-alive
Upgrade-Insecure-Requests:  1
Content-Type:  multipart/form-data;  boundary=-12656974
Content-Length:  345

-12656974
(more data)
```

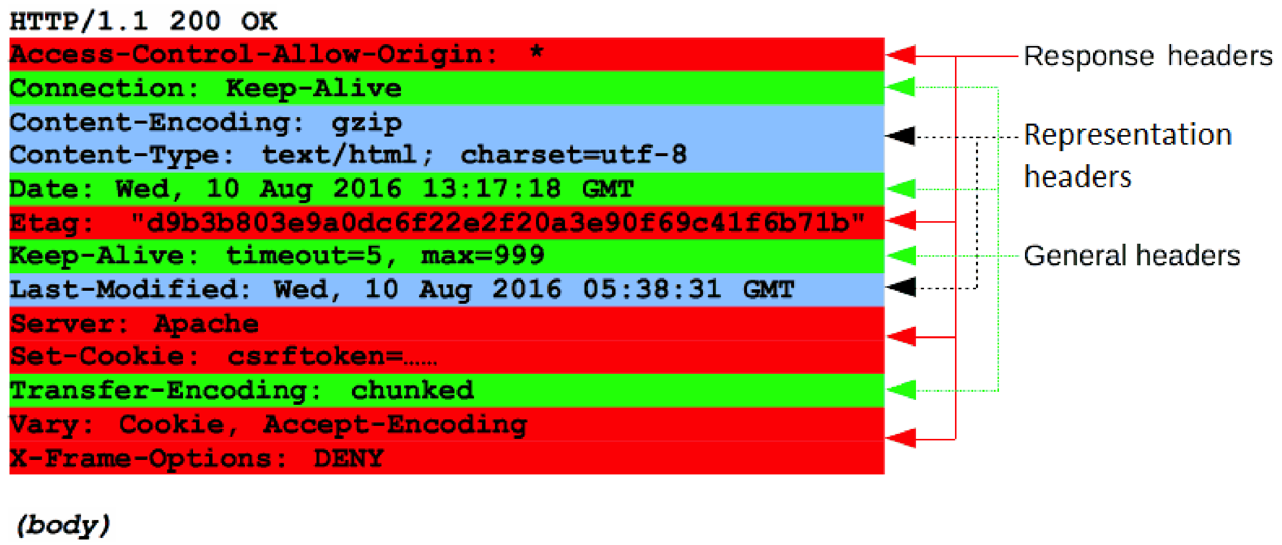
HTTP Request Methods

- GET
  The `GET` method requests a representation of the specified resource. Requests using `GET` should only retrieve data.
- HEAD
  The `HEAD` method asks for a response identical to that of a `GET` request, but without the response body.
- POST
  The `POST` method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- PUT
  The `PUT` method replaces all current representations of the target resource with the request payload.
- DELETE
  The `DELETE` method deletes the specified resource.

HTTP Request Methods

- `CONNECT`
  The `CONNECT` method establishes a tunnel to the server identified by the target resource.
- `OPTIONS`
  The `OPTIONS` method is used to describe the communication options for the target resource.
- `TRACE`
  The `TRACE` method performs a message loop-back test along the path to the target resource.
- `PATCH`
  The `PATCH` method is used to apply partial modifications to a resource.

HTTP Response



```
HTTP/1.1 200 OK
Access-Control-Allow-Origin:  *
Connection:  Keep-Alive
Content-Encoding:  gzip
Content-Type:  text/html;  charset=utf-8
Date:  Wed,  10  Aug  2016  13:17:18  GMT
Etag:  "d9b3b803e9a0dc6f22e2f20a3e90f69c41f6b71b"
Keep-Alive:  timeout=5,  max=999
Last-Modified:  Wed,  10  Aug  2016  05:38:31  GMT
Server:  Apache
Set-Cookie:  csrftoken=......
Transfer-Encoding:  chunked
Vary:  Cookie,  Accept-Encoding
X-Frame-Options:  DENY

(body)
```

HTTP Response Status Codes

- Informational responses (100–199)
- Successful responses (200–299)
- Redirects (300–399)
- Client errors (400–499)
- Server errors (500–599)

Successful Responses

- `200 OK`
  The request has succeeded.
- `201 Created`
  The request has succeeded and a new resource has been created as a result.
- `202 Accepted`
  The request has been received but not yet acted upon. It is intended for cases where another process or server handles the request, or for batch processing.
- `203 Non-Authoritative Information`
  This is mostly used for mirrors or backups of another resource. Except for that specific case, the "200 OK" response is preferred to this status.
- `204 No Content`
  There is no content to send for this request, but the headers may be useful. The user-agent may update its cached headers for this resource with the new ones.
- `205 Reset Content`
  Tells the user-agent to reset the document which sent this request.
- `206 Partial Content`
  This response code is used when the `Range` header is sent from the client to request only part of a resource.

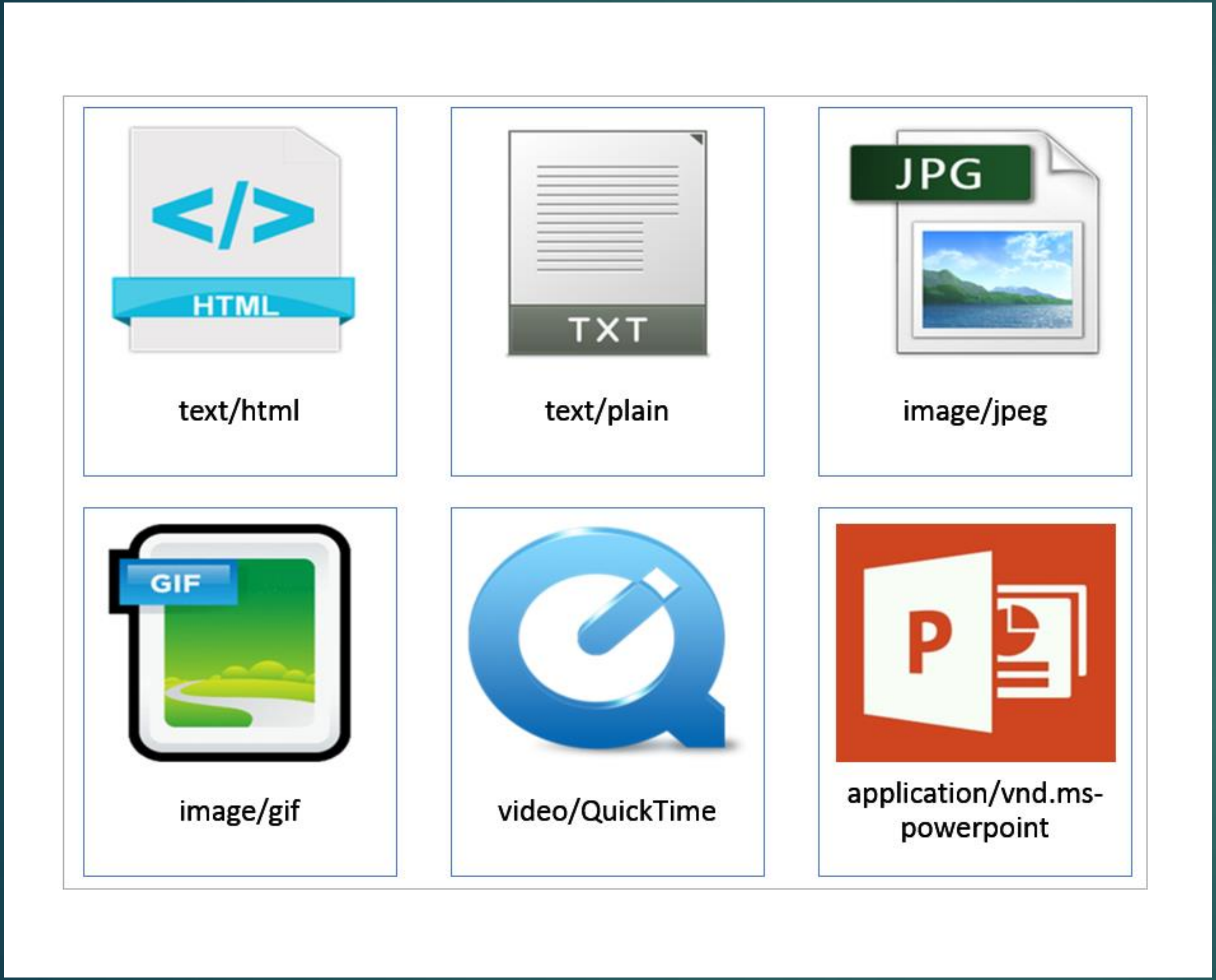
Client Error Responses

- 400 Bad Request
  The server could not understand the request due to invalid syntax.
- 401 Unauthorized
  The client must authenticate itself to get the requested response.
- 402 Payment Required
  This response code is reserved for future use. The initial aim for creating this code was using it for digital payment systems.
- 403 Forbidden
  The client does not have access rights to the content.
- 404 Not Found
  The server cannot find the requested resource.
- 405 Method Not Allowed
  The request method is known by the server but is not supported by the target resource. For example, an API may forbid DELETE-ing a resource.
- 406 Not Acceptable
  This response is sent when the web server, after performing server-driven content negotiation, doesn't find any content that conforms to the criteria given by the user agent.

MIME Types

- Also known as a **Multipurpose Internet Mail Extensions or MIME type.**
- A standard that indicates the nature and format of a document, file, or assortment of bytes.
- The simplest MIME type consists of a *type* and a *subtype*
- Strings which, when concatenated with a slash (/) between them, comprise a MIME type.
  - `type/subtype`

MIME Types


HTML
TXT
JPG
GIF
P
text/html
text/plain
image/jpeg
image/gif
video/QuickTime
application/vnd.ms-powerpoint

HTTP Cookies

- An **HTTP cookie** (web cookie, browser cookie) is a small piece of data that a server sends to the user's web browser. The browser may store it and send it back with later requests to the same server.
- Cookies are mainly used for three purposes:
  - **Session management:** Logins, shopping carts, game scores, or anything else the server should remember
  - **Personalization:** User preferences, themes, and other settings
  - **Tracking:** Recording and analyzing user behavior

HTTP Cookies

- The lifetime of a cookie can be defined in two ways:
  - *Session* cookies are deleted when the current session ends. The browser defines when the "current session" ends, and some browsers use *session restoring* when restarting, which can cause session cookies to last indefinitely long.
  - *Permanent* cookies are deleted at a date specified by the `Expires` attribute, or after a period of time specified by the `Max-Age` attribute
- A cookie with the `Secure` attribute is sent to the server only with an encrypted request over the HTTPS.
- A cookie with the `HttpOnly` attribute is inaccessible to the JavaScript `Document.cookie` API.

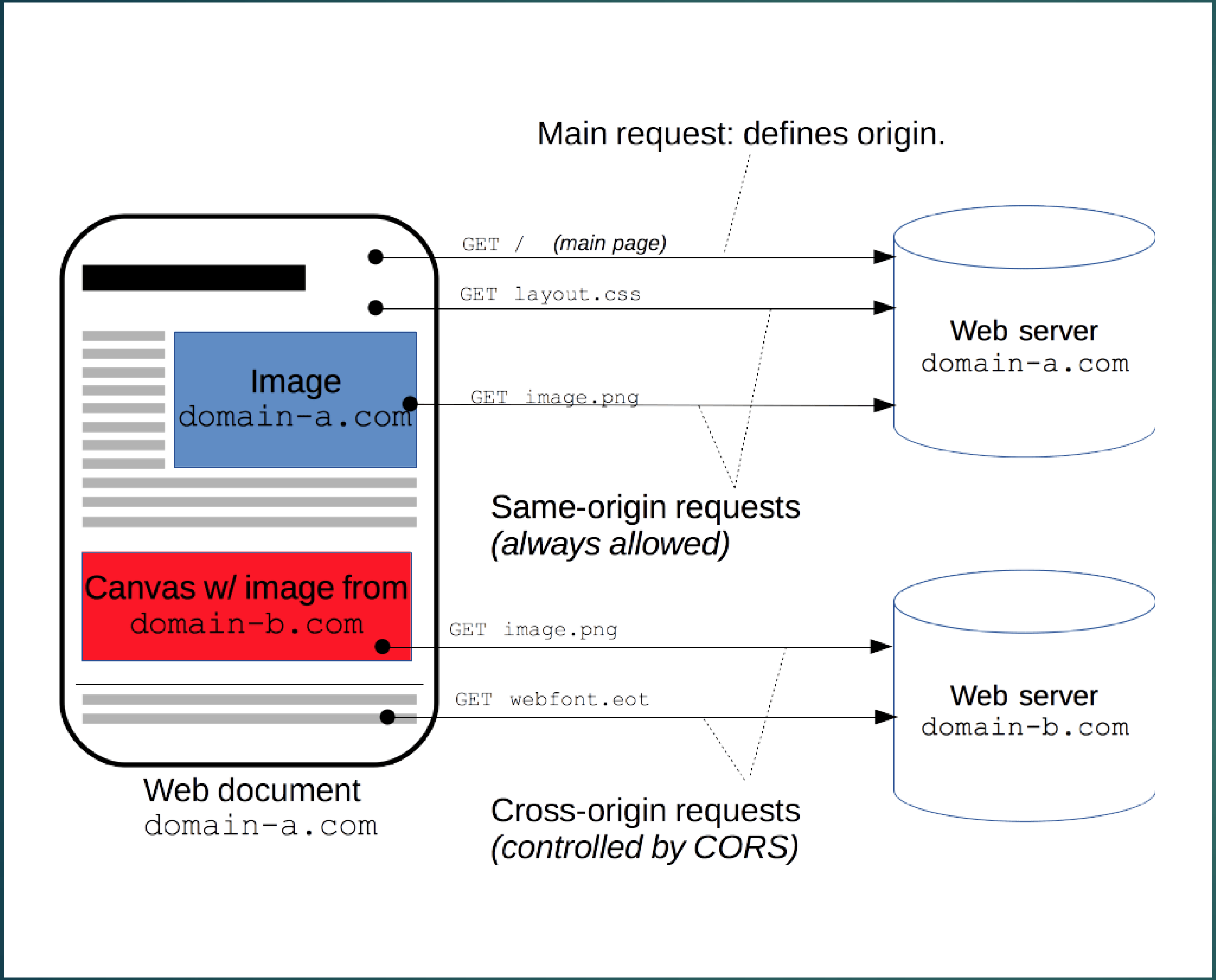
HTTP Cookies

```
Set-Cookie: <cookie-name>=<cookie-value>
Set-Cookie: <cookie-name>=<cookie-value>; Expires=<date>
Set-Cookie: <cookie-name>=<cookie-value>; Max-Age=<number>
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-value>
Set-Cookie: <cookie-name>=<cookie-value>; Path=<path-value>
Set-Cookie: <cookie-name>=<cookie-value>; Secure
Set-Cookie: <cookie-name>=<cookie-value>; HttpOnly

Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Strict
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=Lax
Set-Cookie: <cookie-name>=<cookie-value>; SameSite=None; Secure

// Multiple attributes are also possible, for example:
Set-Cookie: <cookie-name>=<cookie-value>; Domain=<domain-
value>; Secure; HttpOnly
```

HTTP Cookies

- The **`Cookie`** HTTP request header contains stored HTTP cookies associated with the server (i.e., previously sent by the server with the `Set-Cookie` header or set in JavaScript using `Document.cookie`)

- Syntax:
  - `Cookie: <cookie-list>`
  - `Cookie: name=value`
  - `Cookie: name=value; name2=value2; name3=value3`

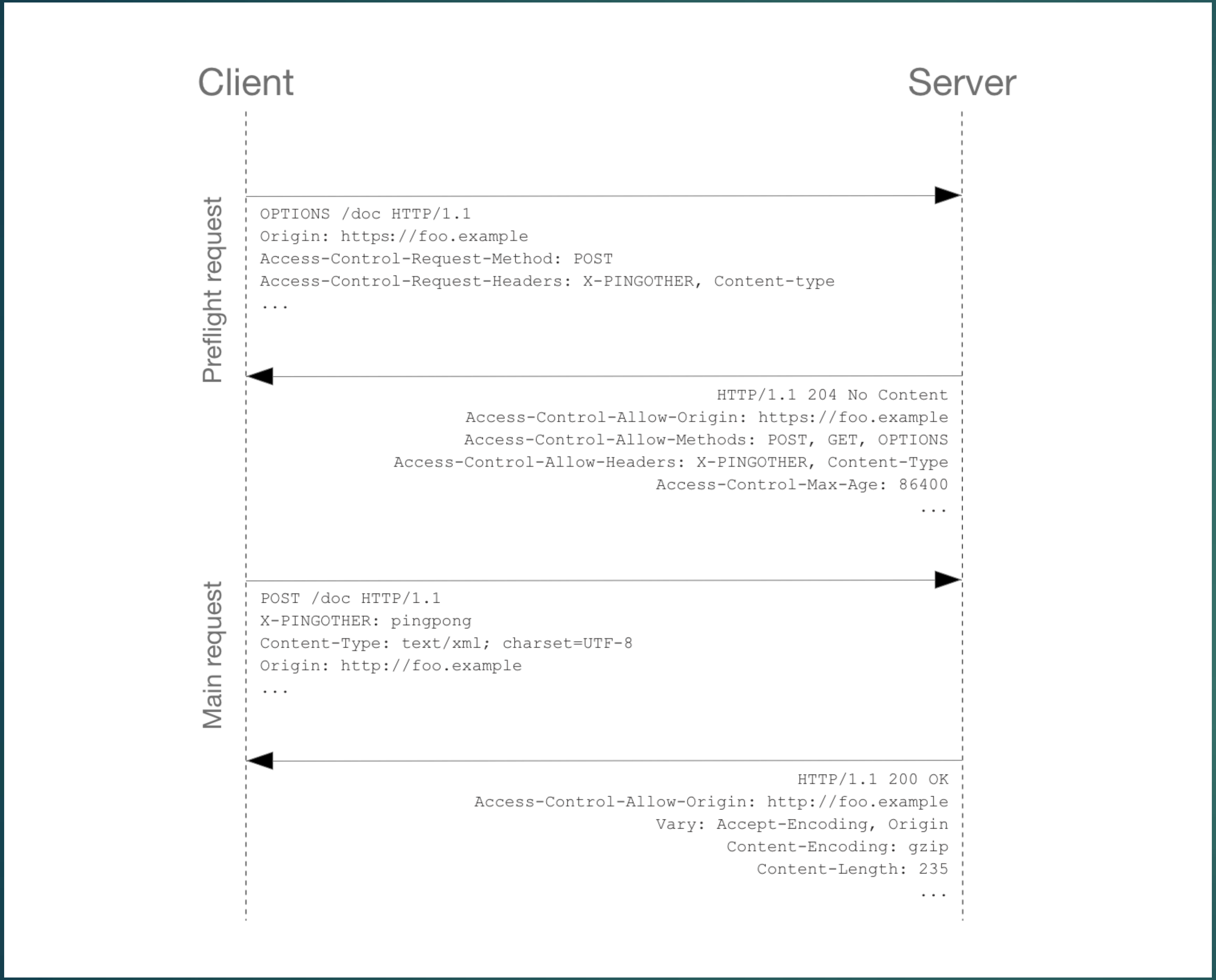


Cross-Origin Resource Sharing (CORS)

Cross-Origin Resource Sharing (CORS)

- Cross-origin resource sharing is a security mechanism that prevents accessing website resources from a different domains or subdomains.
- To resolve the cors error we need to add the `Access-Control-Allow-Origin` header to the response.
- cross-origin sharing standard can enable cross-site HTTP requests for:
  - Invocations of the `XMLHttpRequest` or Fetch APIs.
  - Web Fonts (for cross-domain font usage in `@font-face` within CSS).
  - WebGL textures.
  - Images/video frames drawn to a canvas using `drawImage()`.
  - CSS Shapes from images.

Cross-Origin Resource Sharing (CORS)



```
OPTIONS /doc HTTP/1.1
Origin: https://foo.example
Access-Control-Request-Method: POST
Access-Control-Request-Headers: X-PINGOTHER, Content-type
...
```

```
HTTP/1.1 204 No Content
Access-Control-Allow-Origin: https://foo.example
Access-Control-Allow-Methods: POST, GET, OPTIONS
Access-Control-Allow-Headers: X-PINGOTHER, Content-Type
Access-Control-Max-Age: 86400
...
```

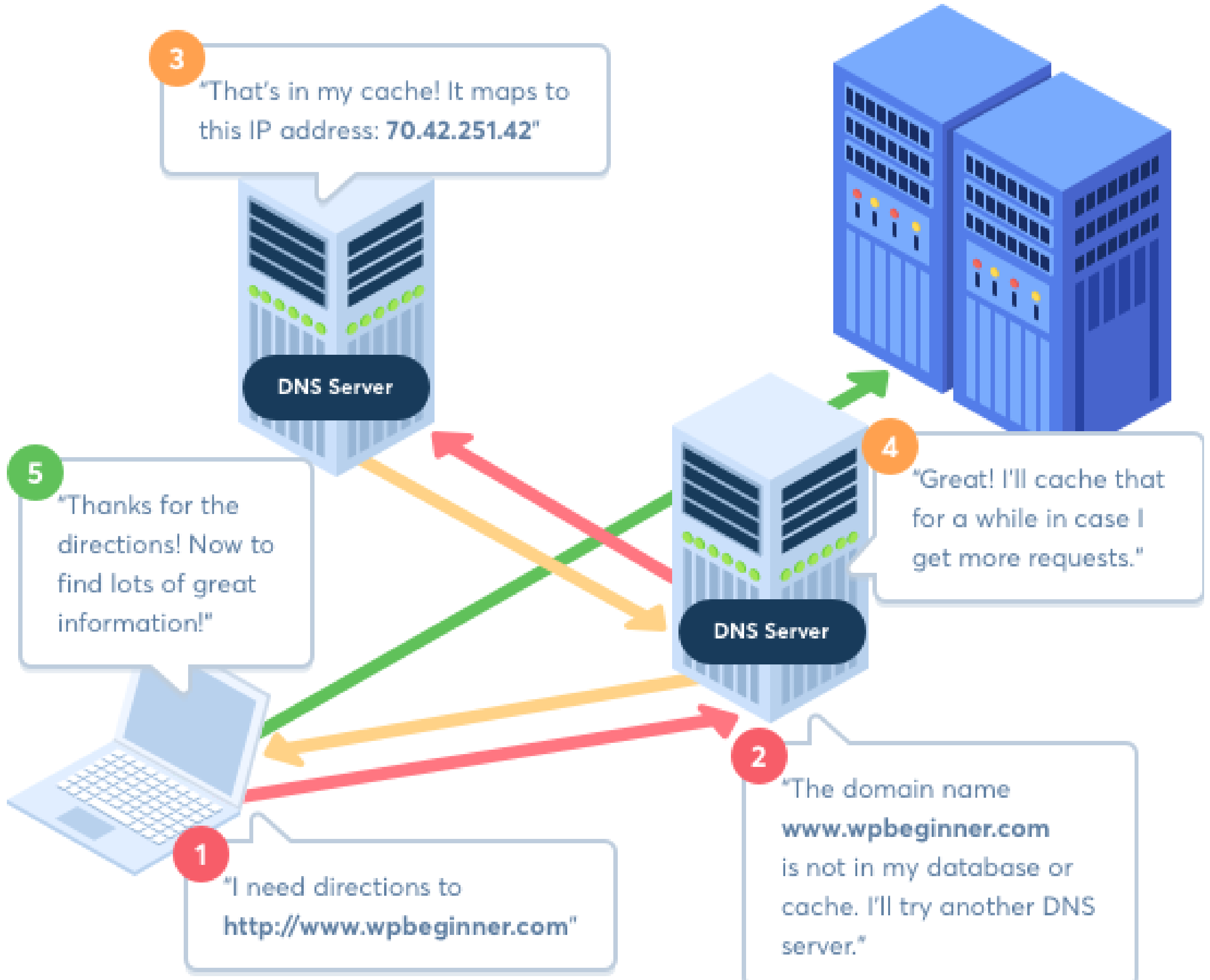
```
POST /doc HTTP/1.1
X-PINGOTHER: pingpong
Content-Type: text/xml; charset=UTF-8
Origin: http://foo.example
...
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: http://foo.example
Vary: Accept-Encoding, Origin
Content-Encoding: gzip
Content-Length: 235
...
```

DNS

- The Domain Name System (DNS) is the phonebook of the Internet. Humans access information online through domain names, like nytimes.com or espn.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.
- Each device connected to the Internet has a unique IP address which other machines use to find the device. DNS servers eliminate the need for humans to memorize IP addresses such as 192.168.1.1 (in IPv4), or more complex newer alphanumeric IP addresses such as 2400:cb00:2048:1::c629:d7a2 (in IPv6).

How Domain Name Works



3 "That's in my cache! It maps to this IP address: **70.42.251.42**"
DNS Server
4 "Great! I'll cache that for a while in case I get more requests."
DNS Server
5 "Thanks for the directions! Now to find lots of great information!"
1 "I need directions to **http://www.wpbeginner.com**"
2 "The domain name **www.wpbeginner.com** is not in my database or cache. I'll try another DNS server."

What is a Name Server?

- A special type of server that keeps all the DNS records of your domain name. Its job is to provide your DNS information to anyone requesting it.
- Nameservers are typically managed by your domain name registrar or hosting provider.
- Each nameserver has its own address and can store the records of many websites. For example, if your website is hosted on Bluehost then the name server used to manage your DNS records will be on nameservers with addresses that look like this: NS1.bluehost.com, NS2.bluehost.com, NS3.bluehost.com
- Each domain name must have at least two nameservers. The first nameserver is the **primary** server. If the primary server does not respond, then the **secondary** nameserver is used to resolve the domain name.

Domain Names

- A domain name is **simply a human readable form of an IP address**. In other words, it's the destination you type into a web browser — such as www.google.com.
- A domain name consists of one or more parts, that are conventionally concatenated, and delimited by dots, such as *example.com*.
- The right-most label conveys the top-level domain; for example, the domain name *www.example.com* belongs to the top-level domain ***com***.
- **Top-level domains:** .com, .net, .org, .ps, ...etc.
- **Second-level and lower level domains:** .co.uk, .edu.ps, ...etc.
- **IDN domains**: .فلسطين

Web Hosting

- A type of Internet hosting service that allows individuals and organizations to make their website accessible via the World Wide Web.
- Web hosts are companies that provide space on a server owned or leased for use by clients, as well as providing Internet connectivity, typically in a data center.

Type of Web Hosting

- **Shared**
  One's website is placed on the same server as many other sites, ranging from a few sites to hundreds of websites
- **VPS (Virtual Dedicated Server)**
  The users may have root access to their own virtual space. Customers are sometimes responsible for patching and maintaining the server (unmanaged server) or the VPS provider may provide server admin tasks for the customer (managed server).
- **Dedicated Hosting Server**
  The user gets his or her own Web server and gains full control over it.
- **Cloud Hosting**
  Cloud hosting also allows providers to charge users only for resources consumed by the user. users can request additional resources on-demand such as only during periods of peak traffic

SSL Certificates

- SSL (Secure Socket Layer), more commonly called TLS (Transport Layer Security), is a protocol for encrypting Internet traffic and verifying server identity.
- Any website with an HTTPS web address uses SSL/TLS.
- SSL certificates are what enable websites to move from HTTP to HTTPS, which is more secure.
- SSL certificates make SSL/TLS encryption possible, and they contain the website's public key and the website's identity, along with related information.

Why do websites need an SSL certificate?

- **Encryption:** SSL/TLS encryption is possible because of the public-private key pairing that SSL certificates facilitate. Clients (such as web browsers) get the public key necessary to open a TLS connection from a server's SSL certificate.
- **Authentication:** SSL certificates verify that a client is talking to the correct server that actually owns the domain. This helps prevent domain spoofing and other kinds of attacks.
- **HTTPS:** Most crucially for businesses, an SSL certificate is necessary for an HTTPS web address. HTTPS is the secure form of HTTP, and HTTPS websites are websites that have their traffic encrypted by SSL/TLS.

Are all SSL certificates the same?

- Based on the number of domain names or subdomains:
  - **Single** – secures one fully-qualified domain name or subdomain name
  - **Wildcard** - covers one domain name and an unlimited number of its subdomains
  - **Multi-Domain** – secures multiple domain names
- Level of validation needed:
  - **Domain Validation** – covers basic encryption and verification of the ownership of the domain name registration.
  - **Organization Validation** – in addition to basic encryption and verification of ownership of the domain name registration, certain details of the owner are authenticated.
  - **Extended Validation (EV)** – this provides the highest degree of security. In addition to ownership of the domain name registration and entity authentication, the legal, physical and operational existence of the entity is verified.

SSL Certificates

- SSL certificates include:
  - The domain name that the certificate was issued for
  - Which person, organization, or device it was issued to
  - Which certificate authority issued it
  - The certificate authority's digital signature
  - Associated subdomains
  - Issue date of the certificate
  - Expiration date of the certificate
  - The public key (the private key is kept secret)

How does a website obtain an SSL certificate?

- Domains need to obtain it from a certificate authority (CA).
- A CA is an outside organization, a trusted third party, that generates and gives out SSL certificates.
- The CA will also digitally sign the certificate with their own private key, allowing client devices to verify it.
- Most, but not all, CAs will charge a fee for issuing an SSL certificate.
- Certificate needs to be installed and activated on the website's origin server. Web hosting services can usually handle this for website operators.

Self-signed SSL Certificate

- Technically, anyone can create their own SSL certificate by generating a public-private key pairing and including all the information mentioned above.
- Such certificates are called self-signed certificates because the digital signature used, instead of being from a CA, would be the website's own private key.
- With self-signed certificates, there's no outside authority to verify that the origin server is who it claims to be.
- Browsers don't consider self-signed certificates trustworthy and may still mark sites with one as "not secure," despite the https:// URL. They may also terminate the connection altogether, blocking the website from loading.

Free SSL Certificates!

- Cloudflare Free SSL/TLS (https://www.cloudflare.com/ssl)
- FreeSSL (https://freessl.org)
- ZeroSSL (https://zerossl.com)
- SSL For Free (https://www.sslforfree.com)
- Let's Encrypt (https://letsencrypt.org)

When choosing the right SSL provider, consider the fact that users' web browsers normally keep a cached list of trusted CAs on file – so if a digital certificate is signed by an entity that's not on the "approved" list, the browser will send a warning message to the user that the website may not be trustworthy.

References

- An overview of HTTP - HTTP | MDN (mozilla.org)
- Identifying resources on the Web - HTTP | MDN (mozilla.org)
- HTTP Messages - HTTP | MDN (mozilla.org)
- HTTP request methods - HTTP | MDN (mozilla.org)
- HTTP response status codes - HTTP | MDN (mozilla.org)
- HTTP headers - HTTP | MDN (mozilla.org)
- MIME types (IANA media types) - HTTP | MDN (mozilla.org)
- Redirections in HTTP - HTTP | MDN (mozilla.org)
- Using HTTP cookies - HTTP | MDN (mozilla.org)
- Cross-Origin Resource Sharing (CORS) - HTTP | MDN (mozilla.org)
- What is DNS? | How DNS works | Cloudflare
- What is an SSL certificate? | How to get a free SSL certificate | Cloudflare