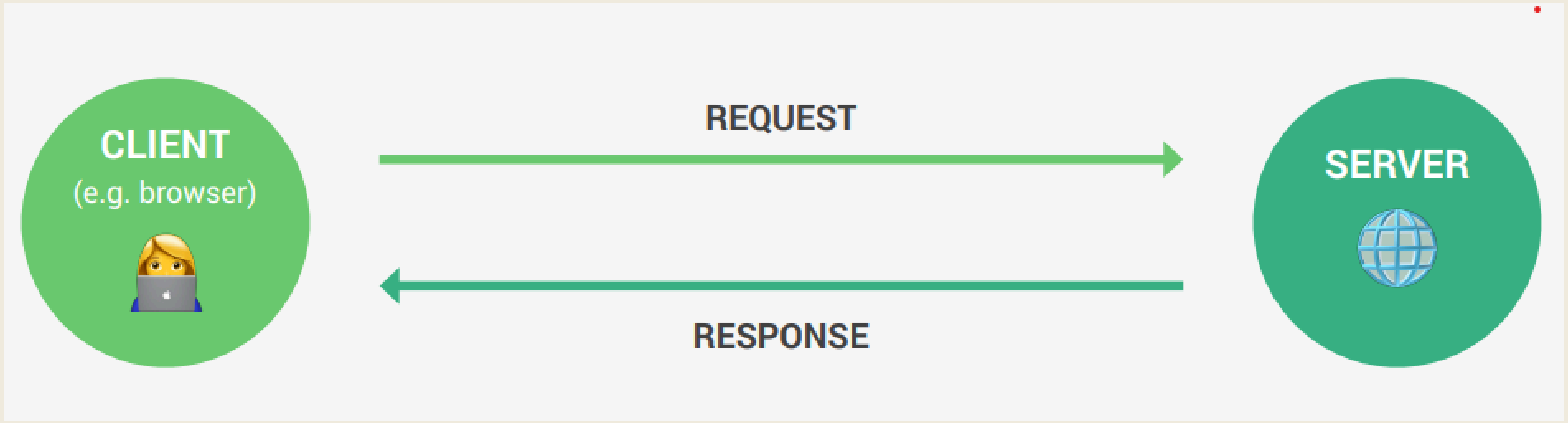
WHAT HAPPENS WHEN WE ACCESS A WEBPAGE

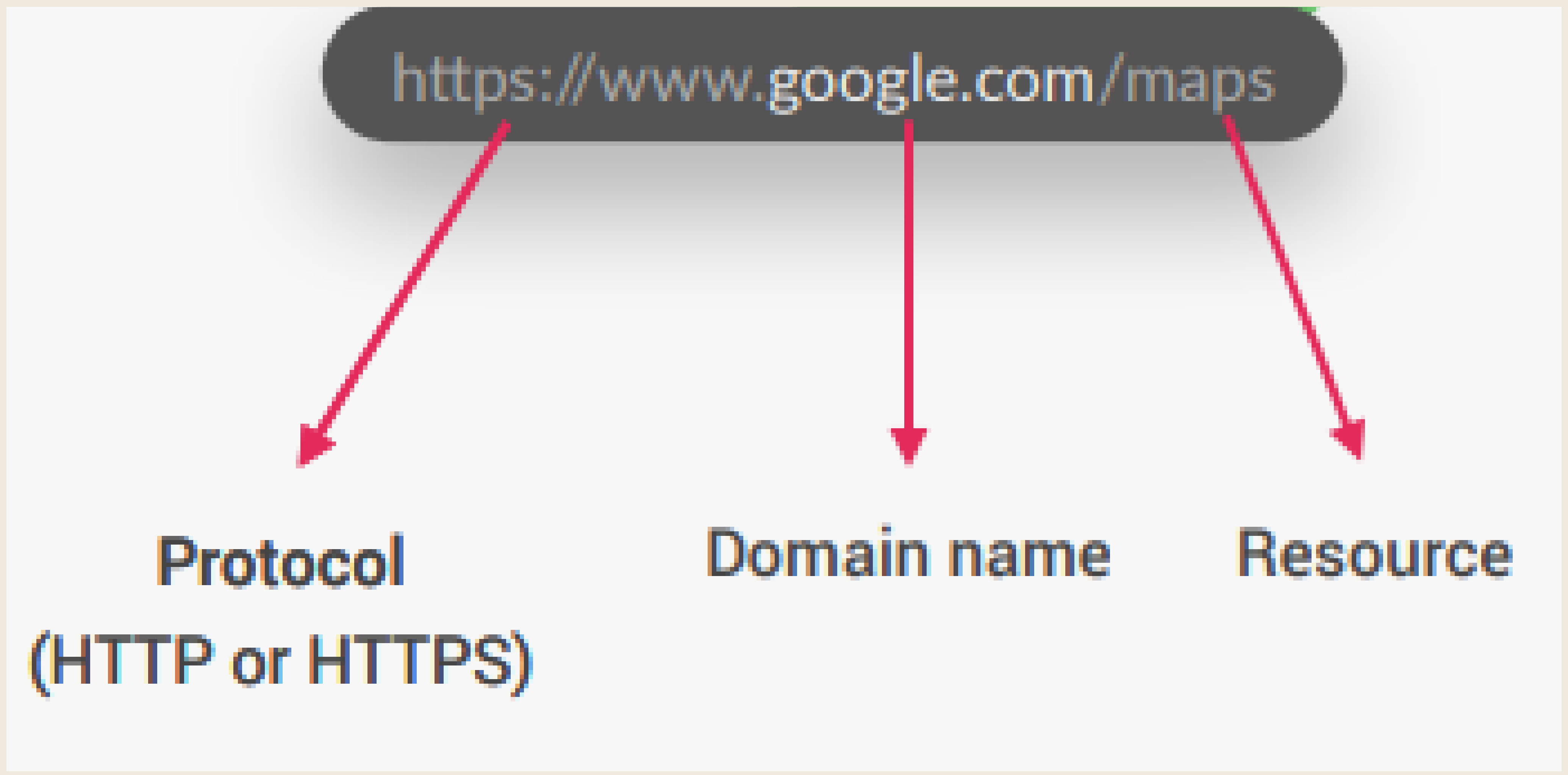
Request-response model or Client-server architecture

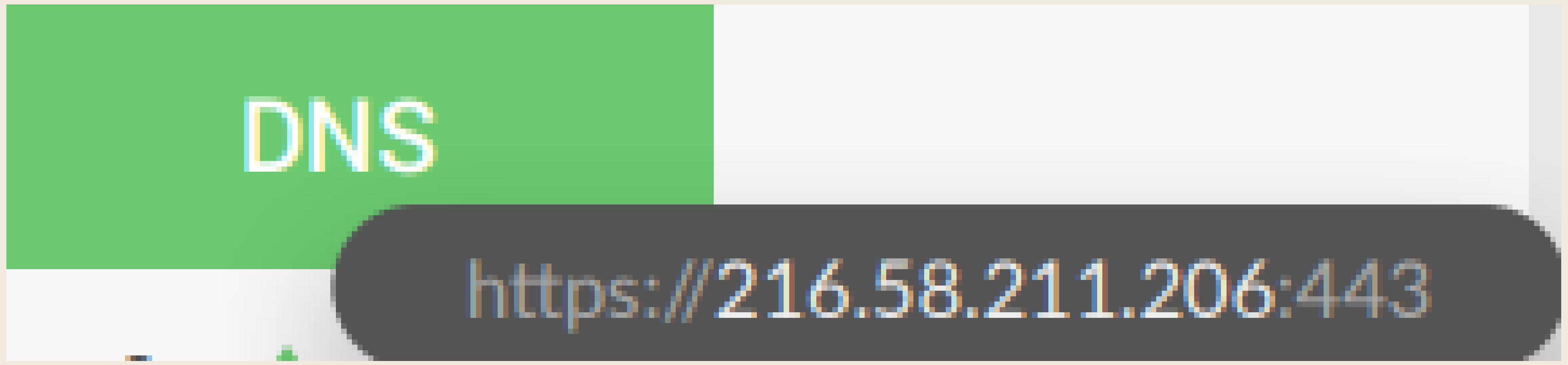


THERE ARE SEVERAL STEPS THAT TAKE PLACE BEHIND THE SCENES TO RETRIEVE AND DISPLAY THE REQUESTED WEBPAGE IN YOUR BROWSE

Enter URL → DNS Lookup → IP Address

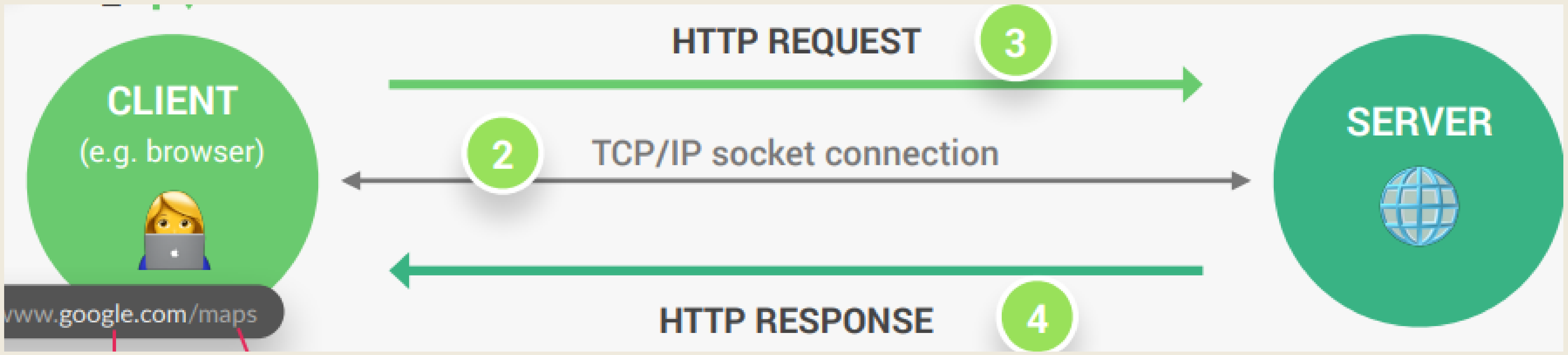
- You enter a **URL (e.g., https://www.example.com)** into your browser's address bar and press Enter.
- The browser sends a **DNS request to find the IP address** of the server that hosts the webpage.
- The DNS server **translates** the domain name (example.com) **into an IP address** (e.g., 192.168.1.1), which identifies the server.


https://www.google.com/maps
Protocol (HTTP or HTTPS)
Domain name
Resource


DNS
https://216.58.211.206:443

TCP/IP & SSL CONNECTION ESTABLISHED

- The browser then **establishes a connection** with the server **using TCP/IP** (Transmission Control Protocol/Internet Protocol).
- If the connection is **secure** (https), the **SSL/TLS** handshake occurs, and a secure connection is established.


HTTP REQUEST
3
CLIENT
(e.g. browser)
2
TCP/IP socket connection
SERVER
www.google.com/maps
HTTP RESPONSE
4

BROWSER SENDS HTTP REQUEST

The browser sends an HTTP request to the server. The request contains:

- **Method**: Typically GET or POST.
- **Headers**: Information about the client (browser type, supported formats, etc.).
- **Request Path**: The specific resource requested (e.g., /index.html).

Server Receives Request

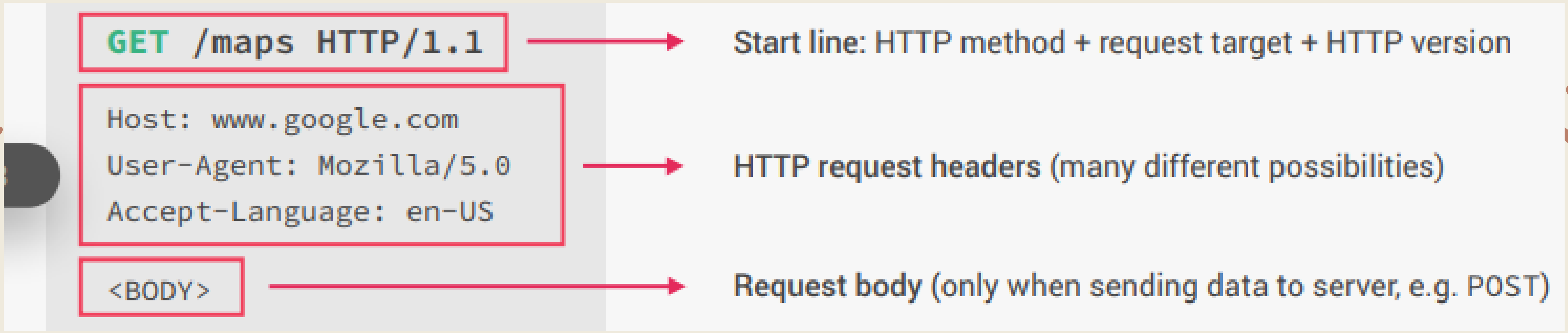
The web server (e.g., Apache, Nginx, IIS) receives the request and determines which resource to serve based on the request path.

Processing the PHP Script

- The server detects that a PHP script is needed. It forwards the request to the PHP engine.
- The PHP engine executes the script, processing server-side code. PHP interacts with the database (like MySQL) if needed, runs logic, and prepares the response.

Processing the Request (Server-Side)

- The server determines the type of resource requested. It may be:
  - **A static file** like an HTML, CSS, JavaScript file, image, or video.
  - **A dynamic script** like PHP, Python, Ruby, or Node.js.
- If it’s a **dynamic resource** (e.g., a PHP file), the server passes the request to the relevant **server-side language engine** (e.g., PHP interpreter).
- The server-side code executes, potentially interacting with a **database** (like MySQL) to retrieve data or perform operations.



```
GET /maps HTTP/1.1
Host: www.google.com
User-Agent: Mozilla/5.0
Accept-Language: en-US
<BODY>
```

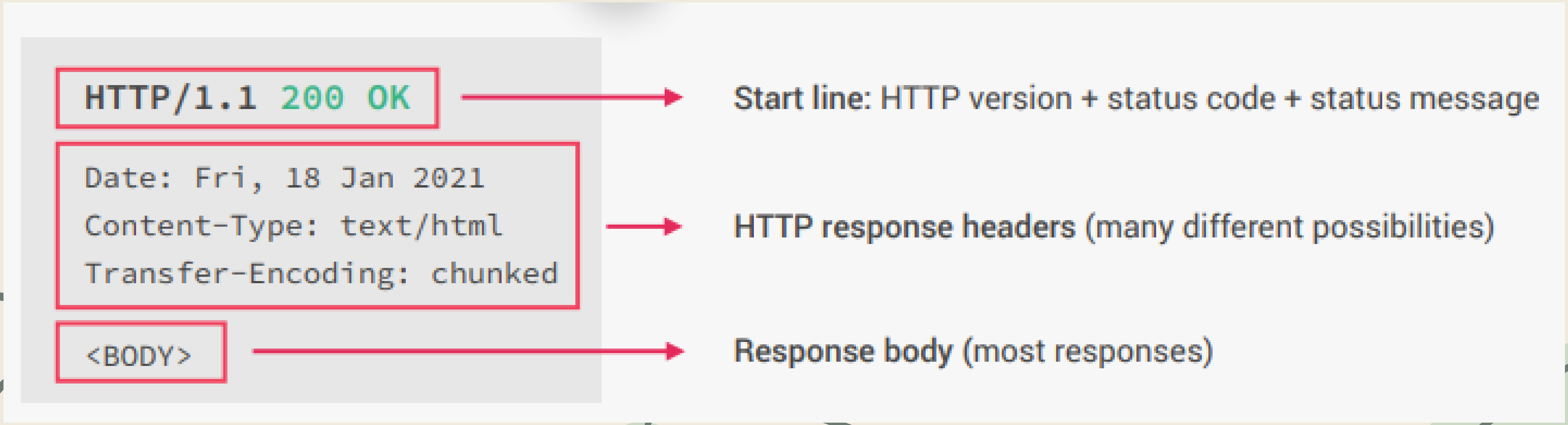
Start line: HTTP method + request target + HTTP version
HTTP request headers (many different possibilities)
Request body (only when sending data to server, e.g. POST)

Server Sends an HTTP Response

- The server prepares an HTTP response, which includes:
    - **Status Code**: (e.g., 200 OK for a successful request or 404 Not Found if the resource doesn't exist).
    - **Response Headers**: Information about the server, caching policies, etc.
    - **Response Body**: The actual content of the webpage (e.g., HTML, CSS, JavaScript).

Browser Receives the Response

The browser receives the HTTP response and starts processing it.



```
HTTP/1.1 200 OK

Date: Fri, 18 Jan 2021
Content-Type: text/html
Transfer-Encoding: chunked

<BODY>
```

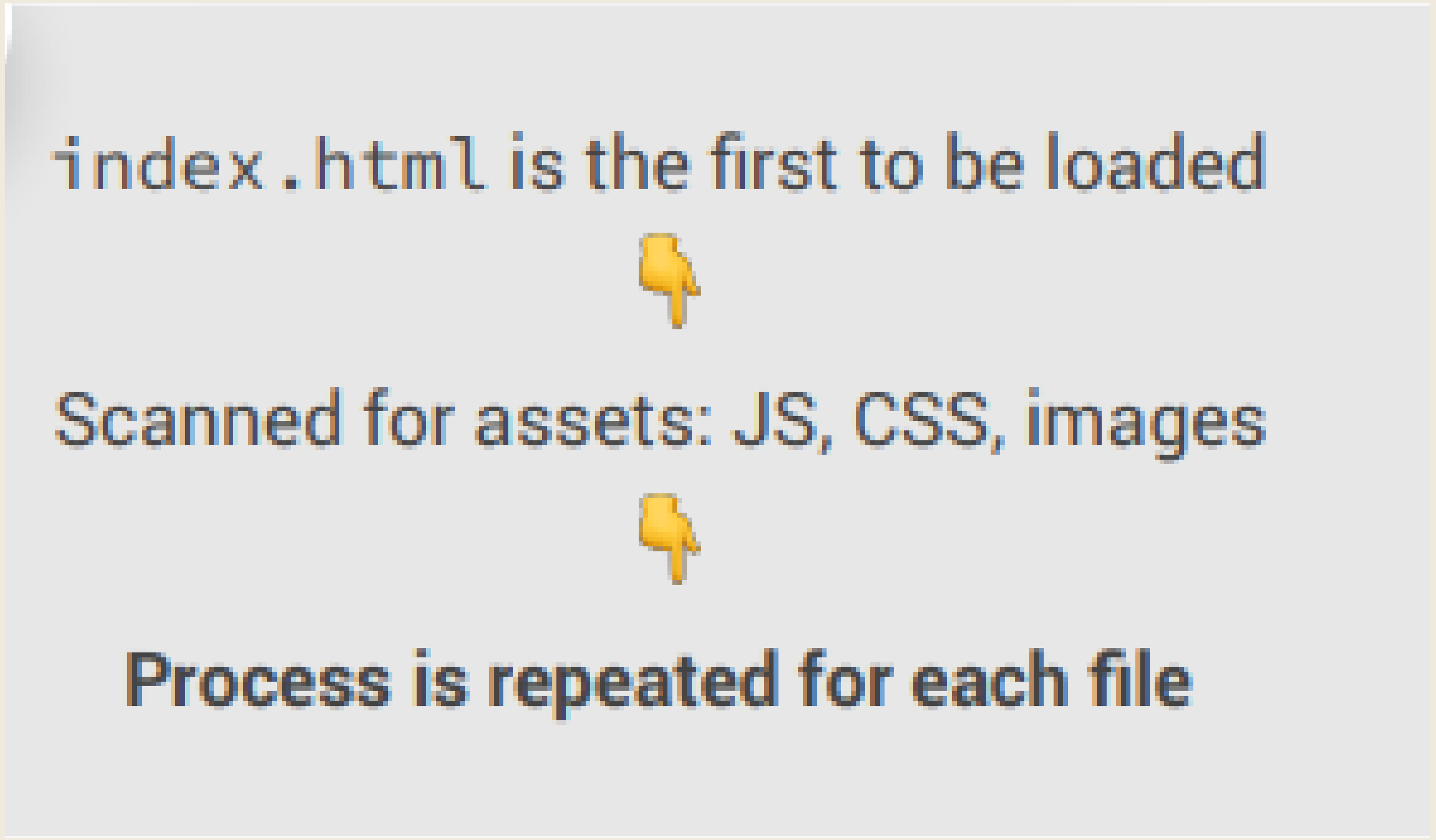
Start line: HTTP version + status code + status message
HTTP response headers (many different possibilities)
Response body (most responses)

Rendering the Webpage

- The browser parses the HTML content received. This process involves:
  - Building the **DOM** (Document Object Model), a structured representation of the HTML document.
  - Fetching and processing **CSS** to style the page (forming the CSSOM - CSS Object Model).
  - Fetching and executing **JavaScript** to add interactivity.
  - Building the render tree by combining the DOM and CSSOM.
  - The browser then **paints** the visual representation on the screen and handles the layout.

Fetching Additional Resources

- While parsing the HTML, the browser encounters references to other resources (like images, CSS files, JavaScript files, etc.) and makes additional HTTP requests to fetch these resources.



`index.html` is the first to be loaded

👇

Scanned for assets: JS, CSS, images

👇

Process is repeated for each file

JavaScript Execution

- Any JavaScript files or inline JavaScript code are executed. JavaScript can manipulate the DOM and modify the rendered page dynamically.

User Interaction

- The final rendered webpage is presented to the user, allowing them to interact with it (click links, submit forms, scroll, etc.). The browser handles these interactions, potentially sending new HTTP requests as needed.

Example

- If you visit https://www.example.com/login:
    - **Browser**: Sends a request to https://www.example.com's IP address.
    - **Server**: Receives the request and runs a PHP script to check if the login form is submitted.
    - **PHP**: Validates user credentials and accesses the database to verify them.
    - **Server**: Sends an HTML response with a success message or redirects to a dashboard.
    - **Browser**: Renders the response and allows further interactions.

THANK YOU

BY EMAN GHAZY