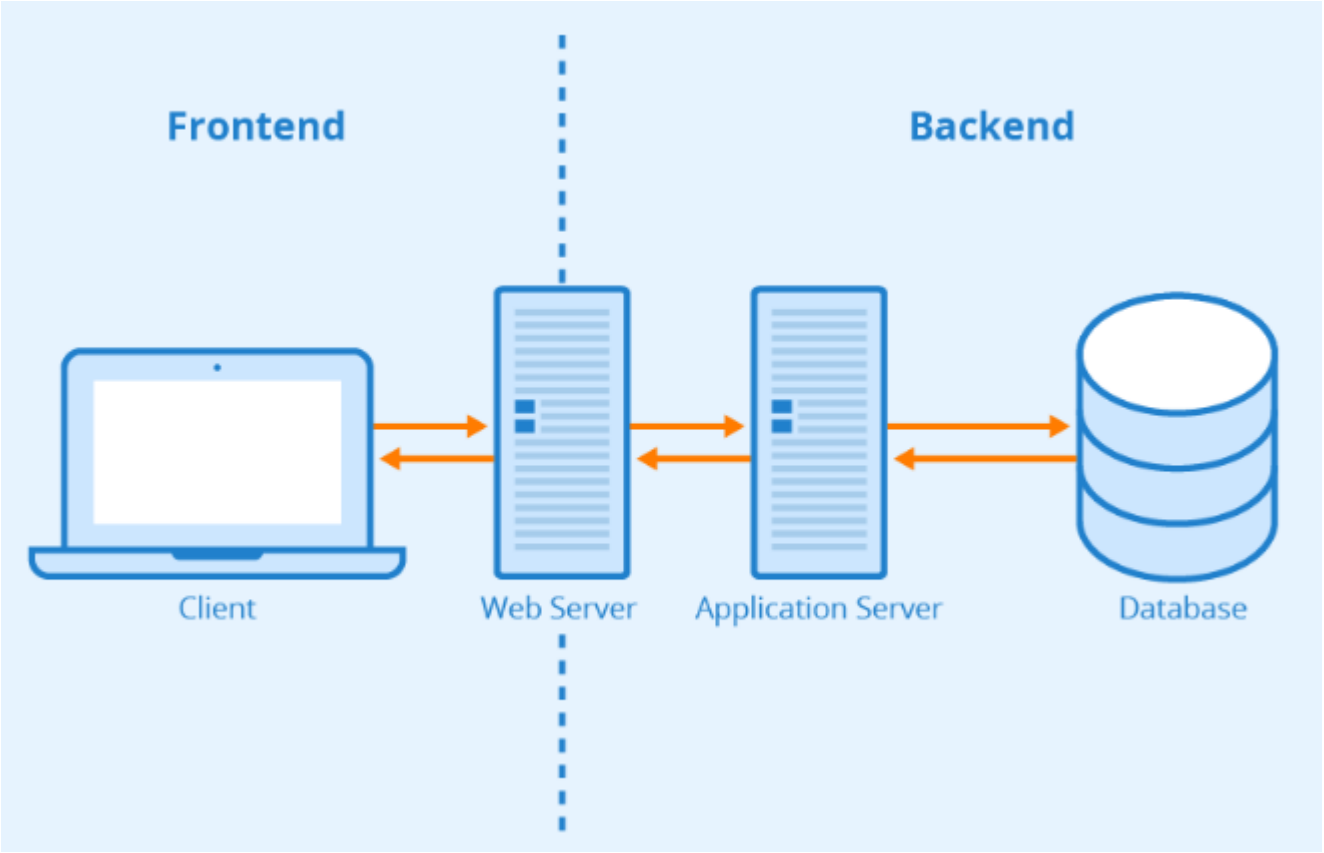
기초 웹

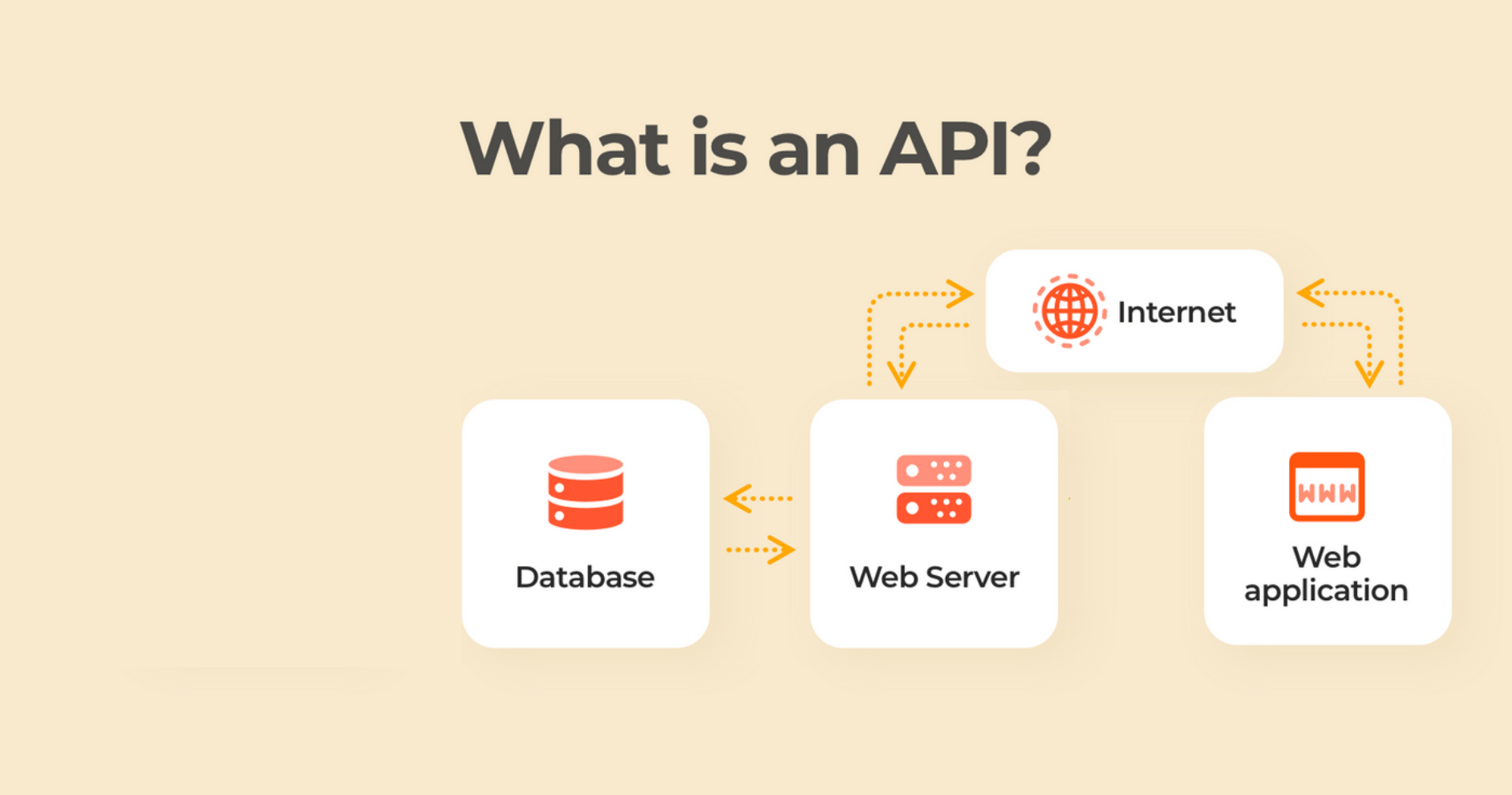
22 Winter CNU 기초 스터디

21 남정연
21 박준서

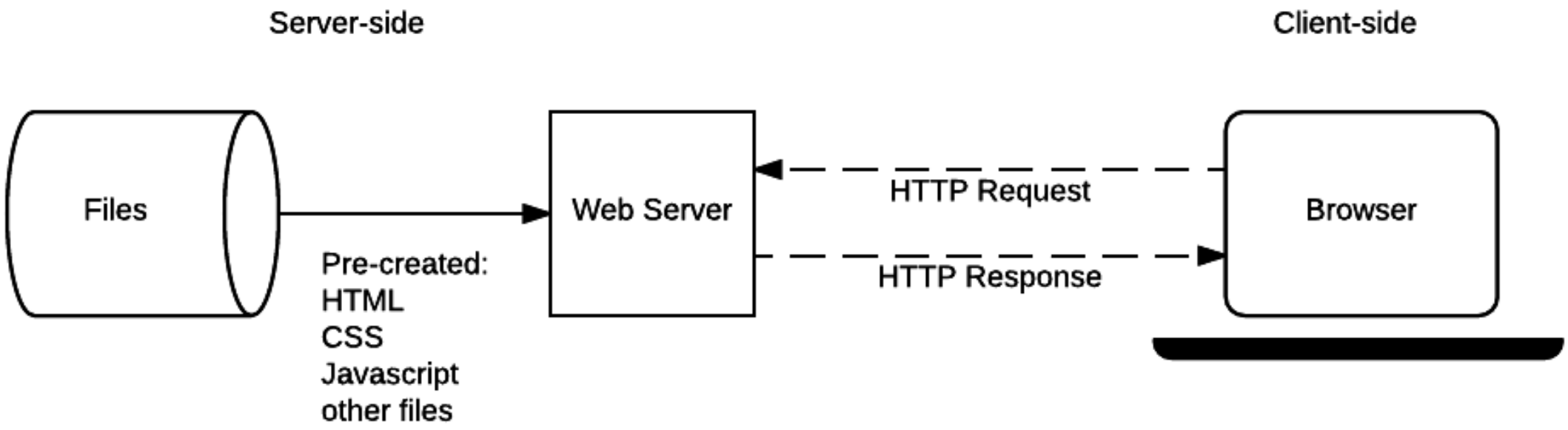
Backend?


Frontend
Backend
Client
Web Server
Application Server
Database

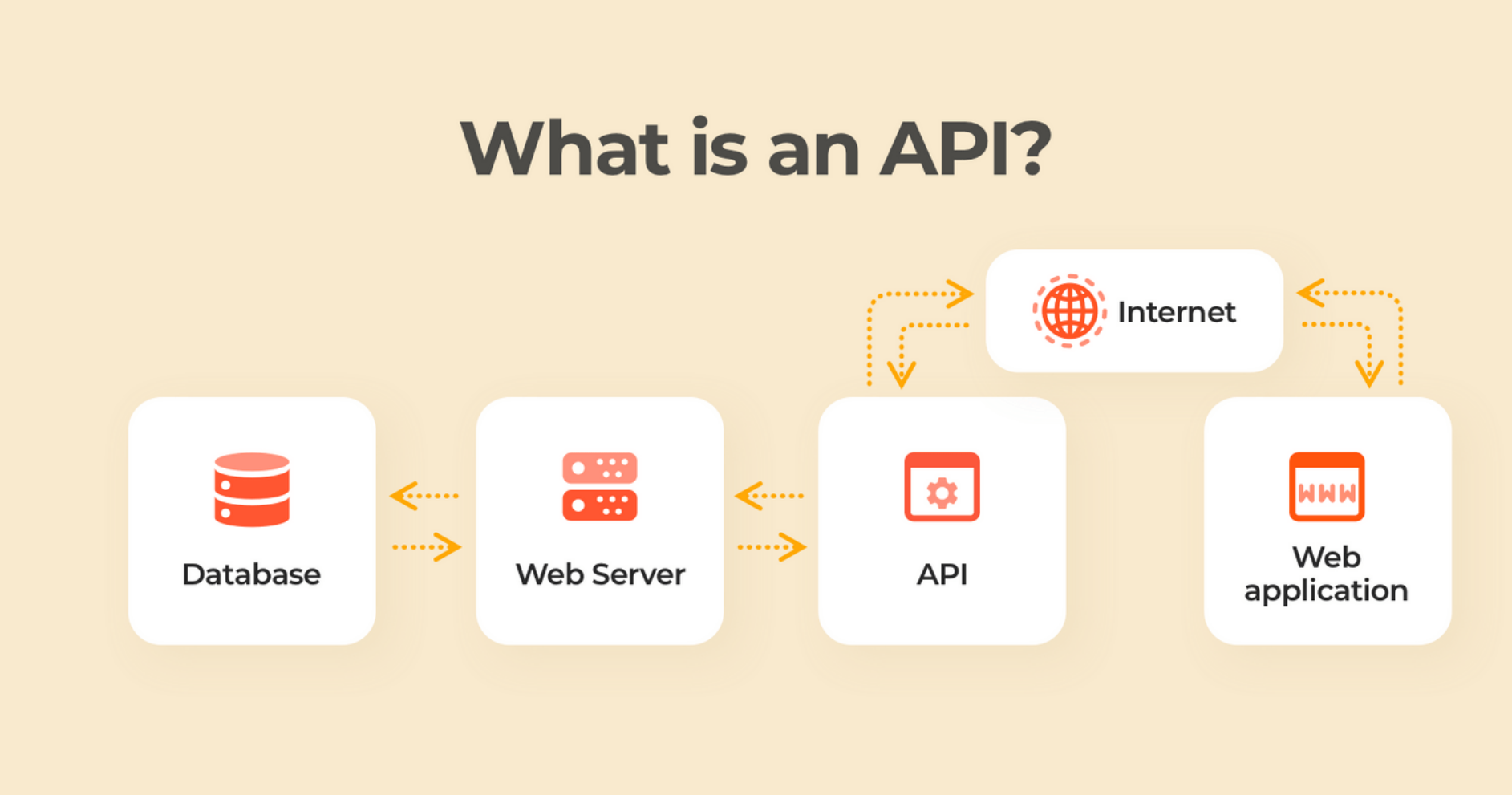
Interactions with Frontend



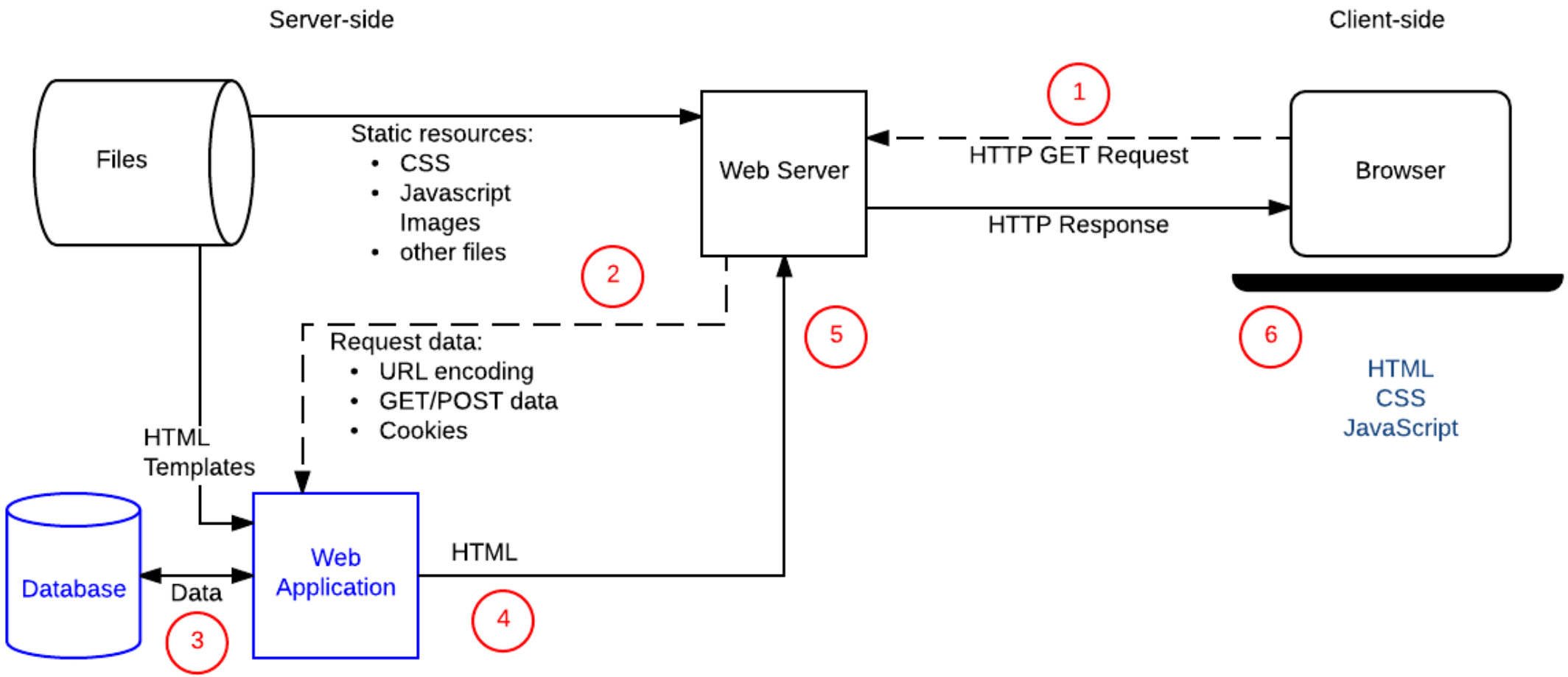
Interactions with Frontend



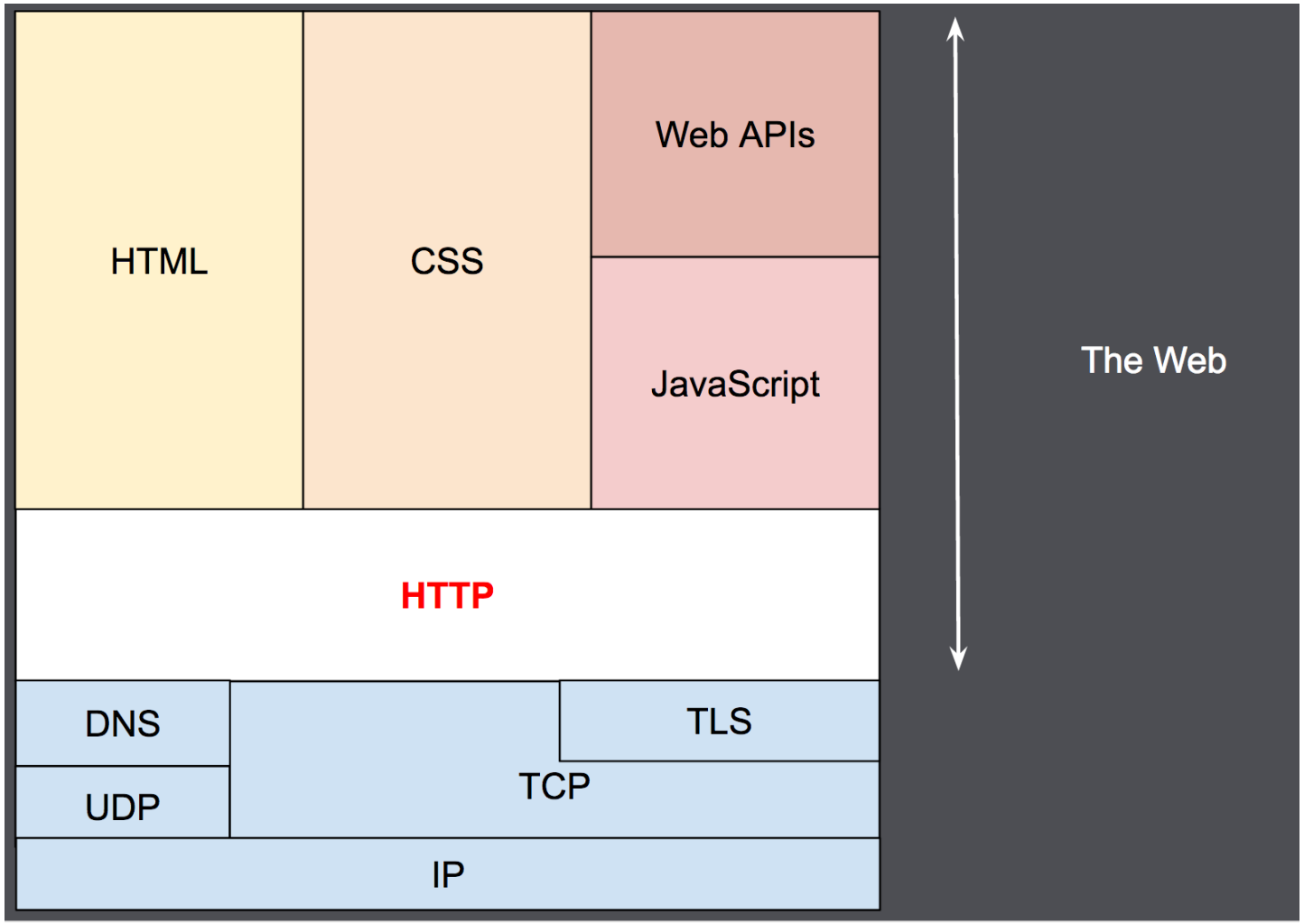
Interactions with Frontend



Interactions with Frontend



HTTP



HTTP 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 a `GET` request, but without the response body.

`POST`
The `POST` method submits 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.

`CONNECT`
The `CONNECT` method establishes a tunnel to the server identified by the target resource.

`OPTIONS`
The `OPTIONS` method describes 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 applies partial modifications to a resource.

HTTP Status Code

Successful responses

`200 OK`

The request succeeded. The result meaning of "success" depends on the HTTP method:

- `GET`: The resource has been fetched and transmitted in the message body.
- `HEAD`: The representation headers are included in the response without any message body.
- `PUT` or `POST`: The resource describing the result of the action is transmitted in the message body.
- `TRACE`: The message body contains the request message as received by the server.

`201 Created`

The request succeeded, and a new resource was created as a result. This is typically the response sent after `POST` requests, or some `PUT` requests.

`202 Accepted`

The request has been received but not yet acted upon. It is noncommittal, since there is no way in HTTP to later send an asynchronous response indicating the outcome of the request. It is intended for cases where another process or server handles the request, or for batch processing.

HTTP Status Code

Redirection messages

300 Multiple Choice

The request has more than one possible response. The user agent or user should choose one of them. (There is no standardized way of choosing one of the responses, but HTML links to the possibilities are recommended so the user can pick.)

301 Moved Permanently

The URL of the requested resource has been changed permanently. The new URL is given in the response.

302 Found

This response code means that the URI of requested resource has been changed *temporarily*. Further changes in the URI might be made in the future. Therefore, this same URI should be used by the client in future requests.

303 See Other

The server sent this response to direct the client to get the requested resource at another URI with a GET request.

HTTP Status Code

Client error responses

`400 Bad Request`

The server could not understand the request due to invalid syntax.

`401 Unauthorized`

Although the HTTP standard specifies "unauthorized", semantically this response means "unauthenticated". That is, 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, however this status code is used very rarely and no standard convention exists.

`403 Forbidden`

The client does not have access rights to the content; that is, it is unauthorized, so the server is refusing to give the requested resource. Unlike `401 Unauthorized`, the client's identity is known to the server.

`404 Not Found`

The server can not find the requested resource. In the browser, this means the URL is not recognized. In an API, this can also mean that the endpoint is valid but the resource itself does not exist. Servers may also send this response instead of `403 Forbidden` to hide the existence of a resource from an unauthorized client. This response code is probably the most well known due to its frequent occurrence on the web.

HTTP Status Code

Server error responses

500 Internal Server Error

The server has encountered a situation it does not know how to handle.

501 Not Implemented

The request method is not supported by the server and cannot be handled. The only methods that servers are required to support (and therefore that must not return this code) are `GET` and `HEAD`.

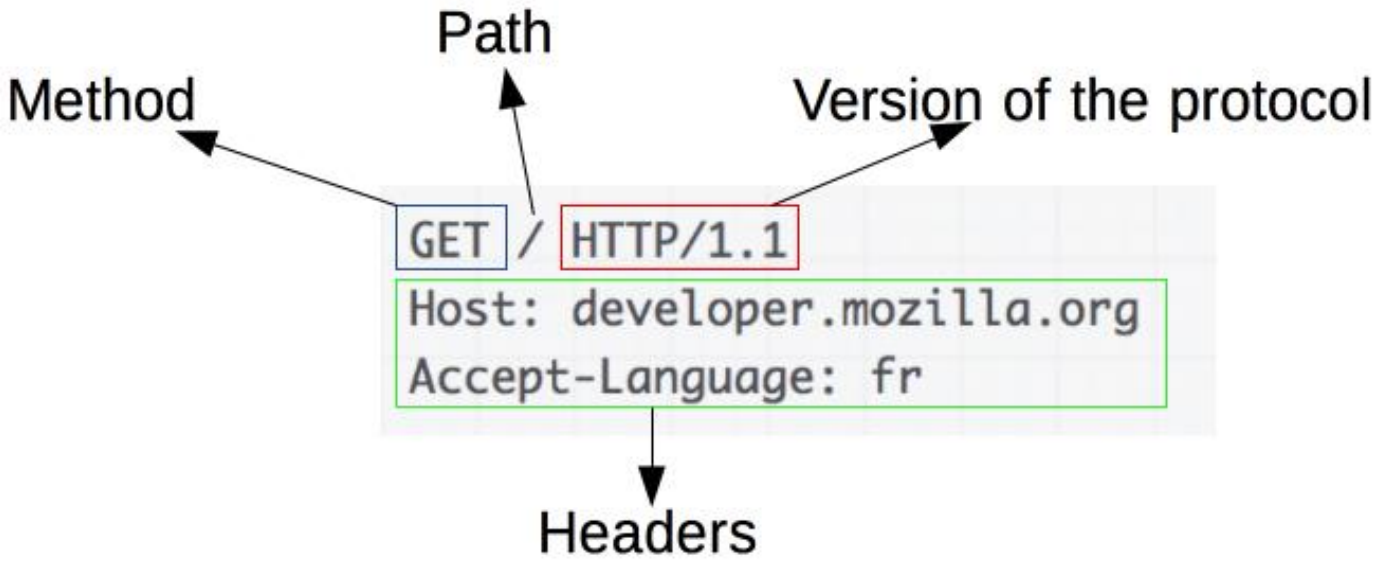
502 Bad Gateway

This error response means that the server, while working as a gateway to get a response needed to handle the request, got an invalid response.

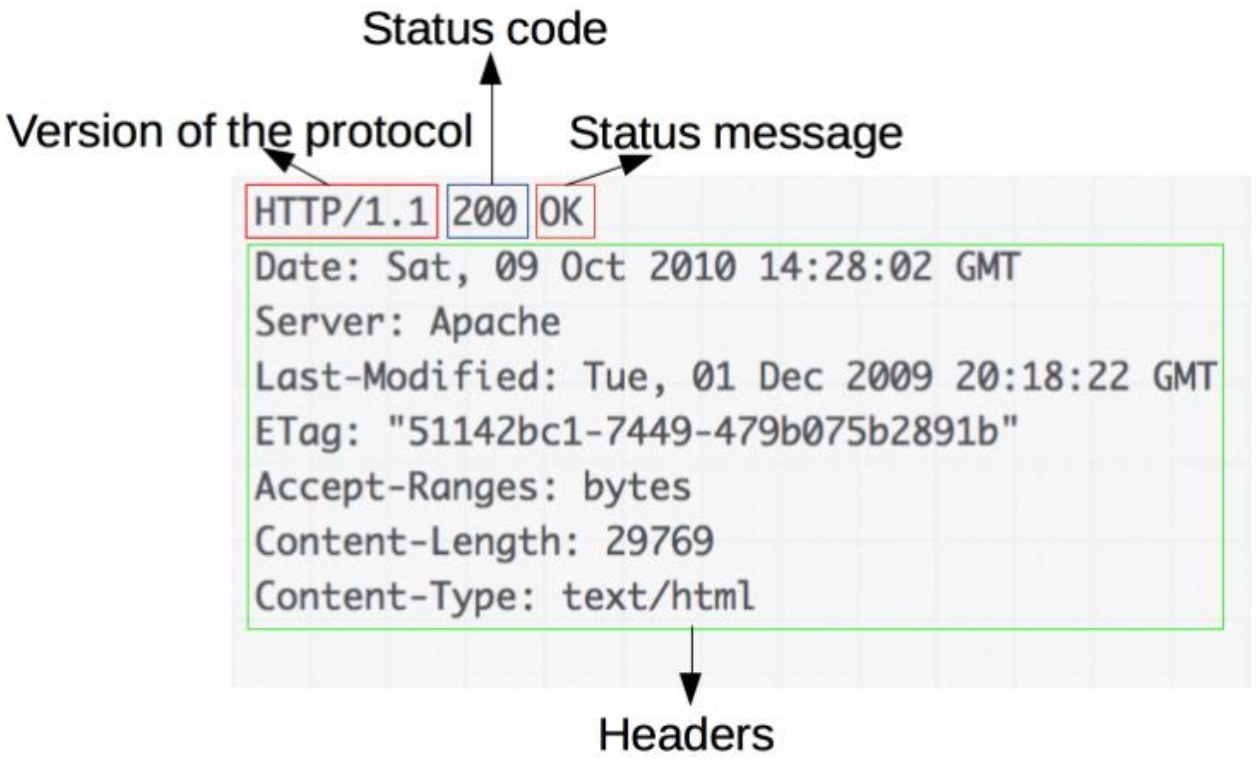
503 Service Unavailable

The server is not ready to handle the request. Common causes are a server that is down for maintenance or that is overloaded. Note that together with this response, a user-friendly page explaining the problem should be sent. This response should be used for temporary conditions and the `Retry-After` HTTP header should, if possible, contain the estimated time before the recovery of the service. The webmaster must also take care about the caching-related headers that are sent along with this response, as these temporary condition responses should usually not be cached.

HTTP Request



HTTP Response



HTTP Request

```
GET /en-US/search?q=client+server+overview&topic=apps&topic=html&topic=css&topic=js&topic=api&topic=webdev
Host: developer.mozilla.org
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://developer.mozilla.org/en-US/
Accept-Encoding: gzip, deflate, sdch, br
Accept-Charset: ISO-8859-1,UTF-8;q=0.7,*;q=0.7
Accept-Language: en-US,en;q=0.8,es;q=0.6
Cookie: sessionid=6ynxs23n521lu21b1t136rhbv7ezngie; csrftoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT; dwf_sectio
```

HTTP Request

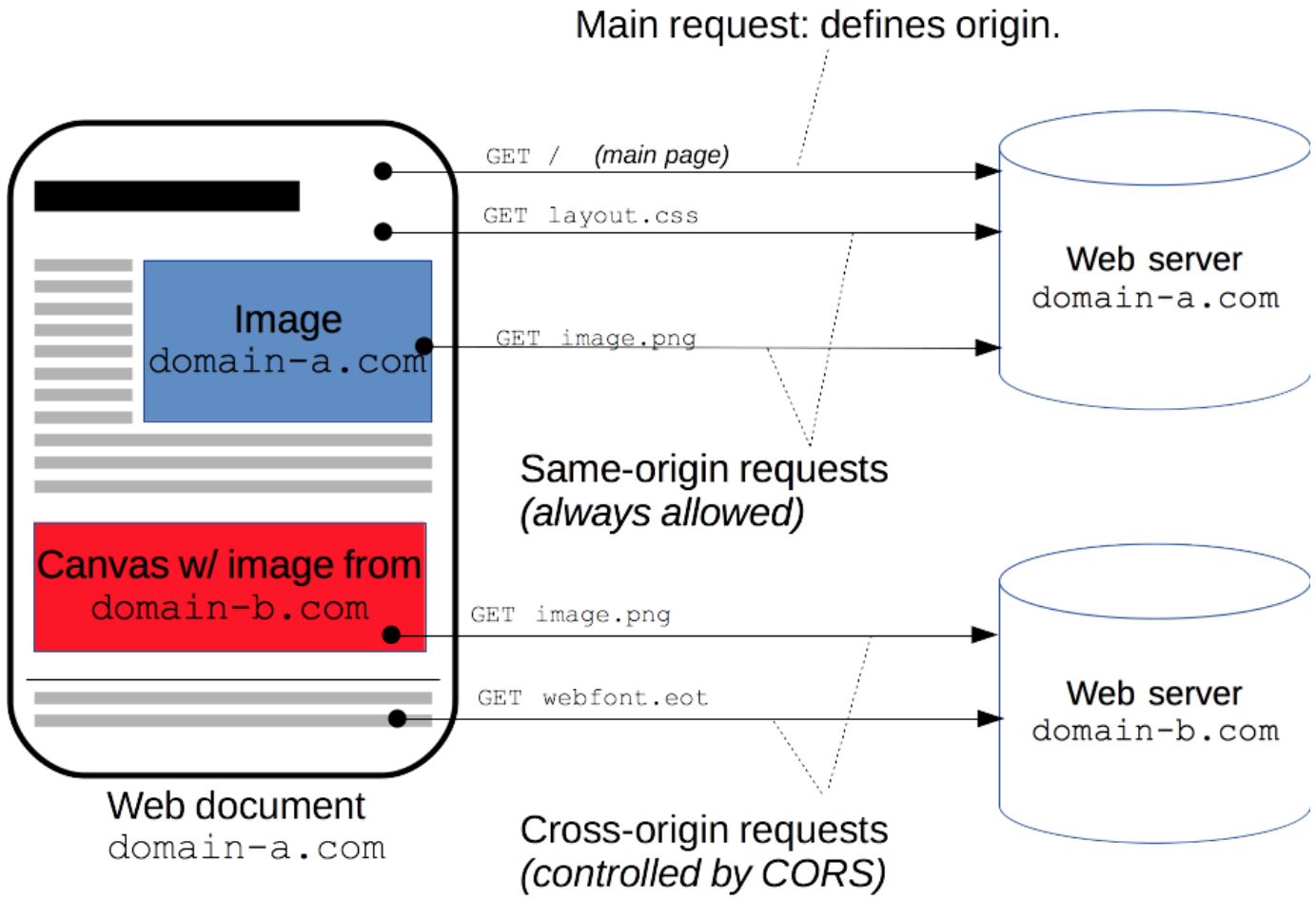
```
POST /en-US/profiles/hamishwillee/edit HTTP/1.1
Host: developer.mozilla.org
Connection: keep-alive
Content-Length: 432
Pragma: no-cache
Cache-Control: no-cache
Origin: https://developer.mozilla.org
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.11
Content-Type: application/x-www-form-urlencoded
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Referer: https://developer.mozilla.org/en-US/profiles/hamishwillee/edit
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.8,es;q=0.6
Cookie: sessionid=6ynxs23n521lu21b1t136rhbv7ezngie; _gat=1; csrftoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT; dwf

csrfmiddlewaretoken=zIPUJsAZv6pcgCBJSCj1zU6pQZbfMUAT&user-username=hamishwillee&user-fullname=Hamish+Willee
```

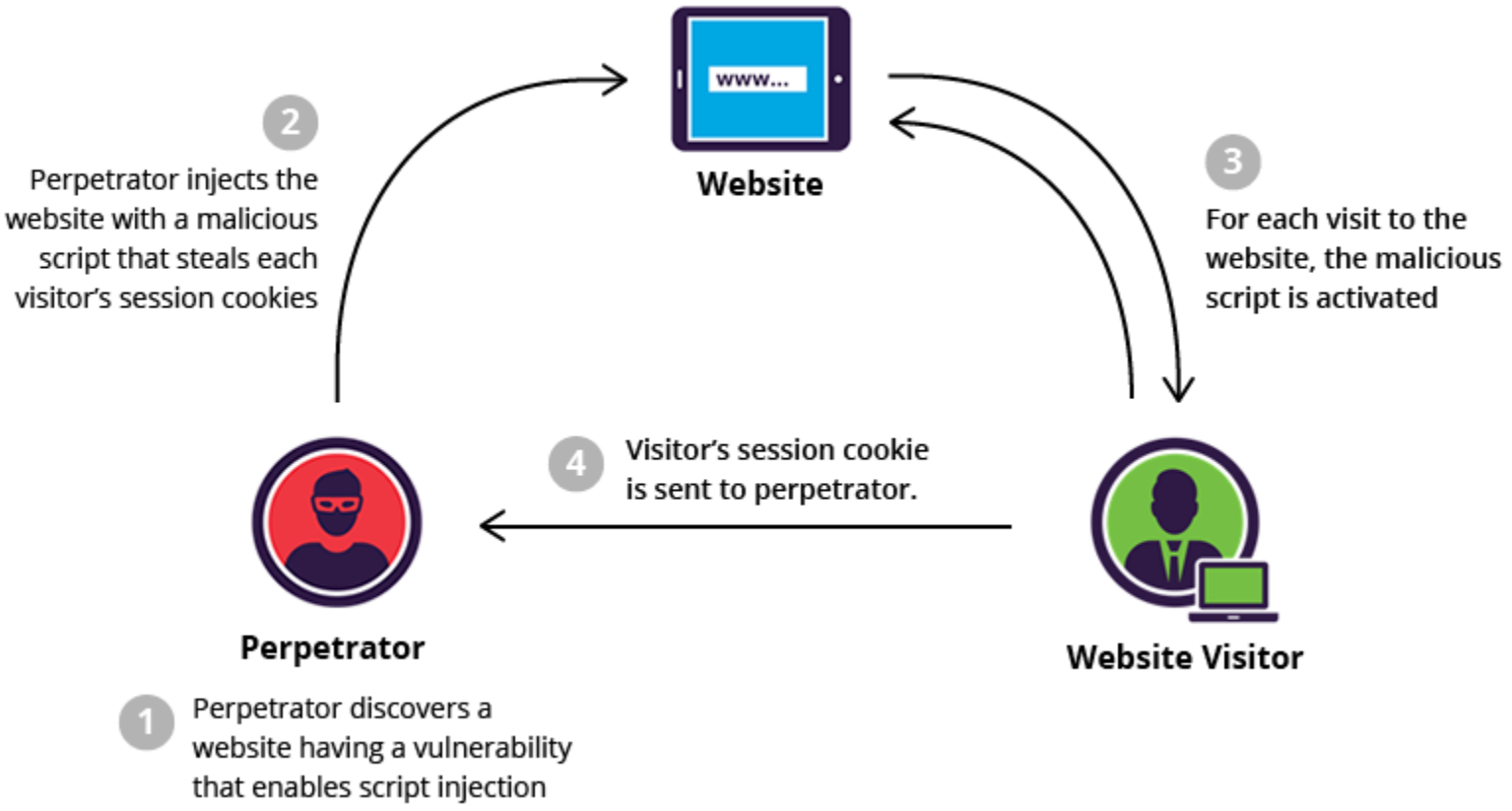
HTTP Response

```
HTTP/1.1 302 FOUND
Server: Apache
X-Backend-Server: developer3.webapp.scl3.mozilla.com
Vary: Cookie
Vary: Accept-Encoding
Content-Type: text/html; charset=utf-8
Date: Wed, 07 Sep 2016 00:38:13 GMT
Location: https://developer.mozilla.org/en-US/profiles/hamishwillee
Keep-Alive: timeout=5, max=1000
Connection: Keep-Alive
X-Frame-Options: DENY
X-Cache-Info: not cacheable; request wasn't a GET or HEAD
Content-Length: 0
```

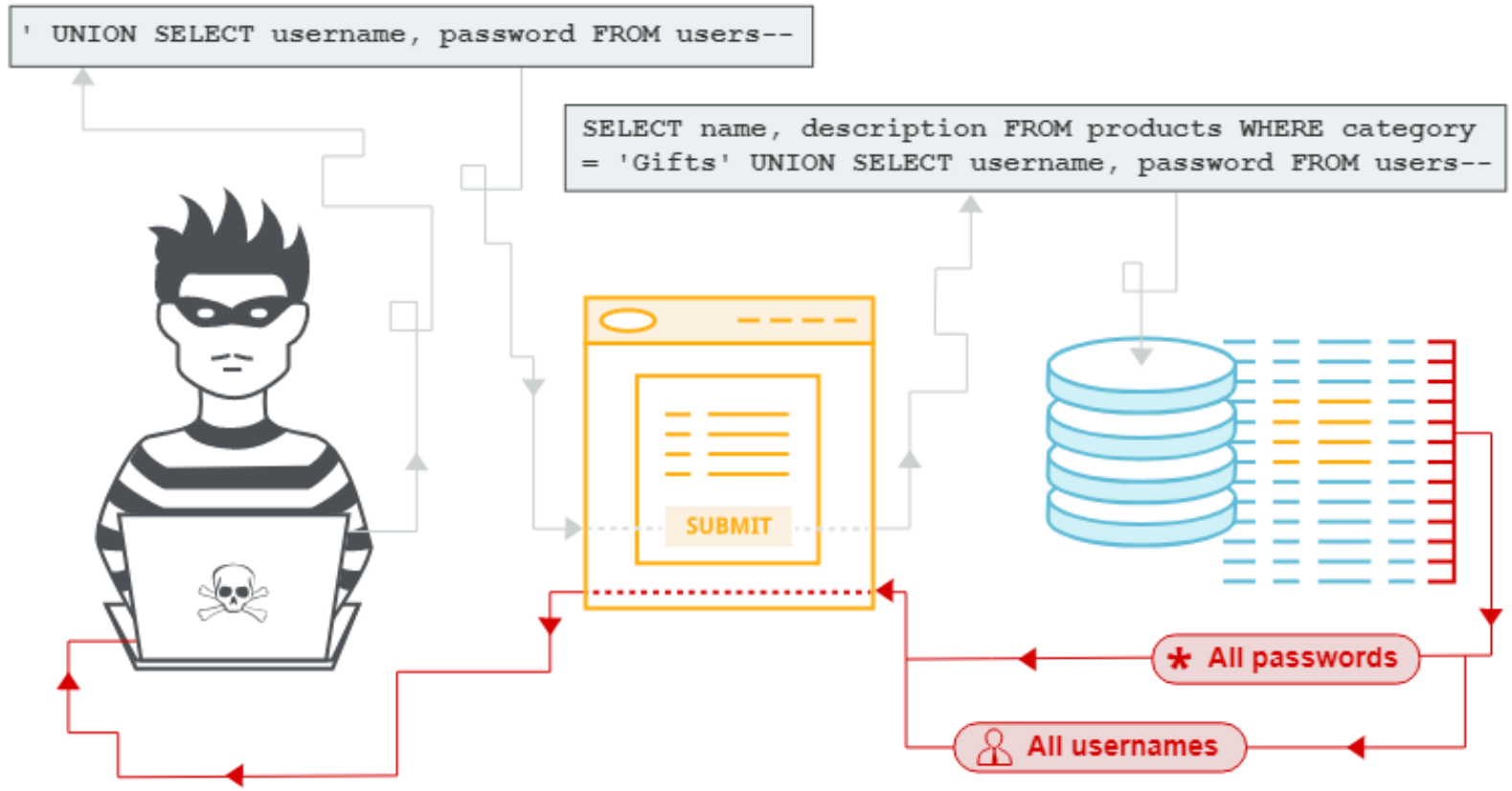
CORS(Cross-Origin Resource Sharing)



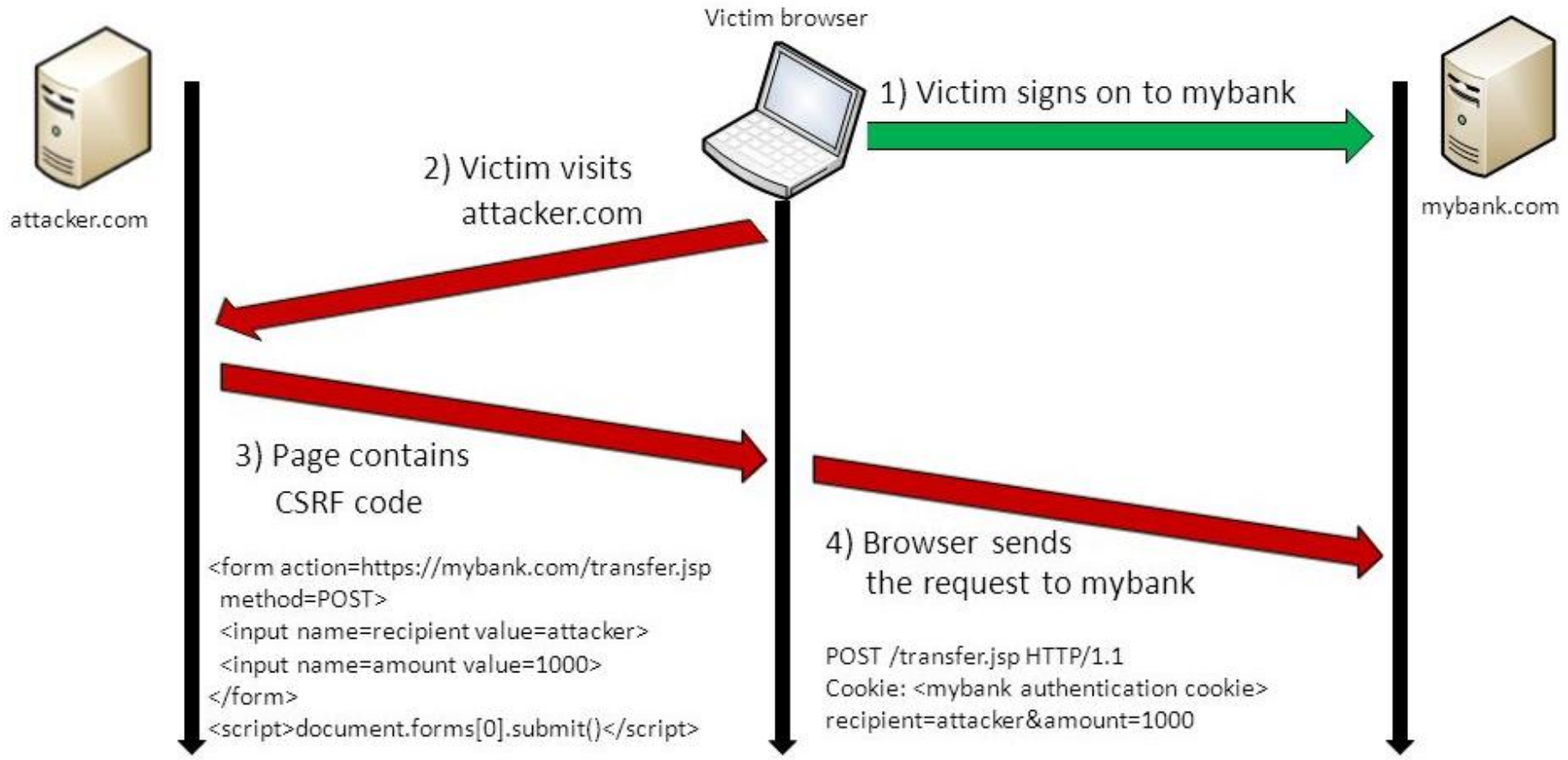
XSS (Cross-Site Scripting)



SQL Injection



CSRF(Cross-Site Request Forgery)



CSRF(Cross-Site Request Forgery)