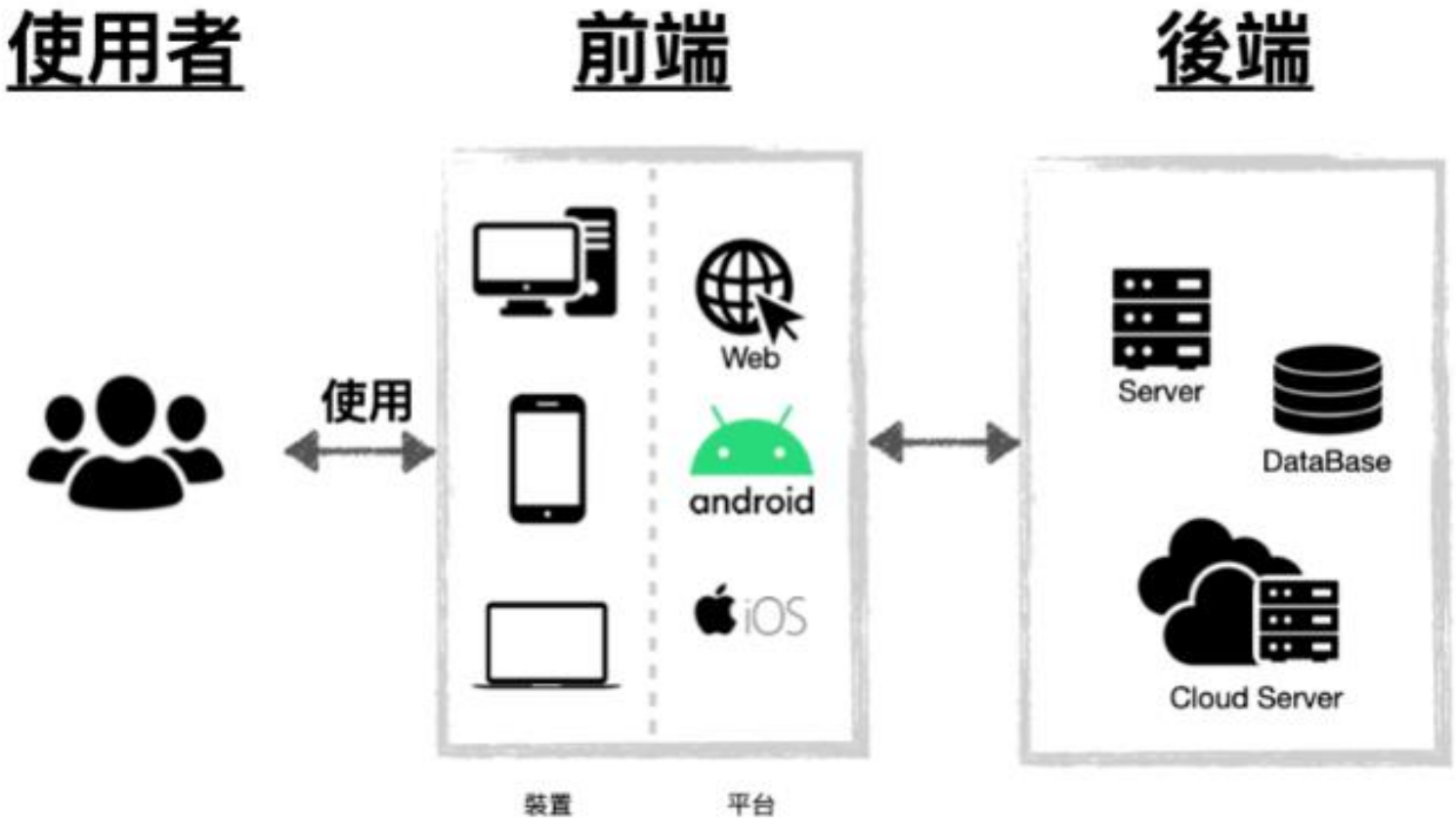
web

王姝云

網頁是什麼

簡單來說就是用你的瀏覽器找到的一個頁面，每個頁面有不同功能



前端 前端框架/套件 Bootstrap, jQuery, React...

前端 Web 前端語言 HTML, CSS, JavaScript

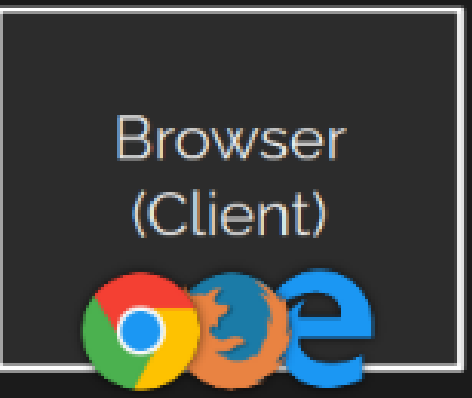
後端 Web 開發框架 Laravel, Express, Spring, Flask...

後端 Web 後端語言 PHP, Node.js, Java, Python...

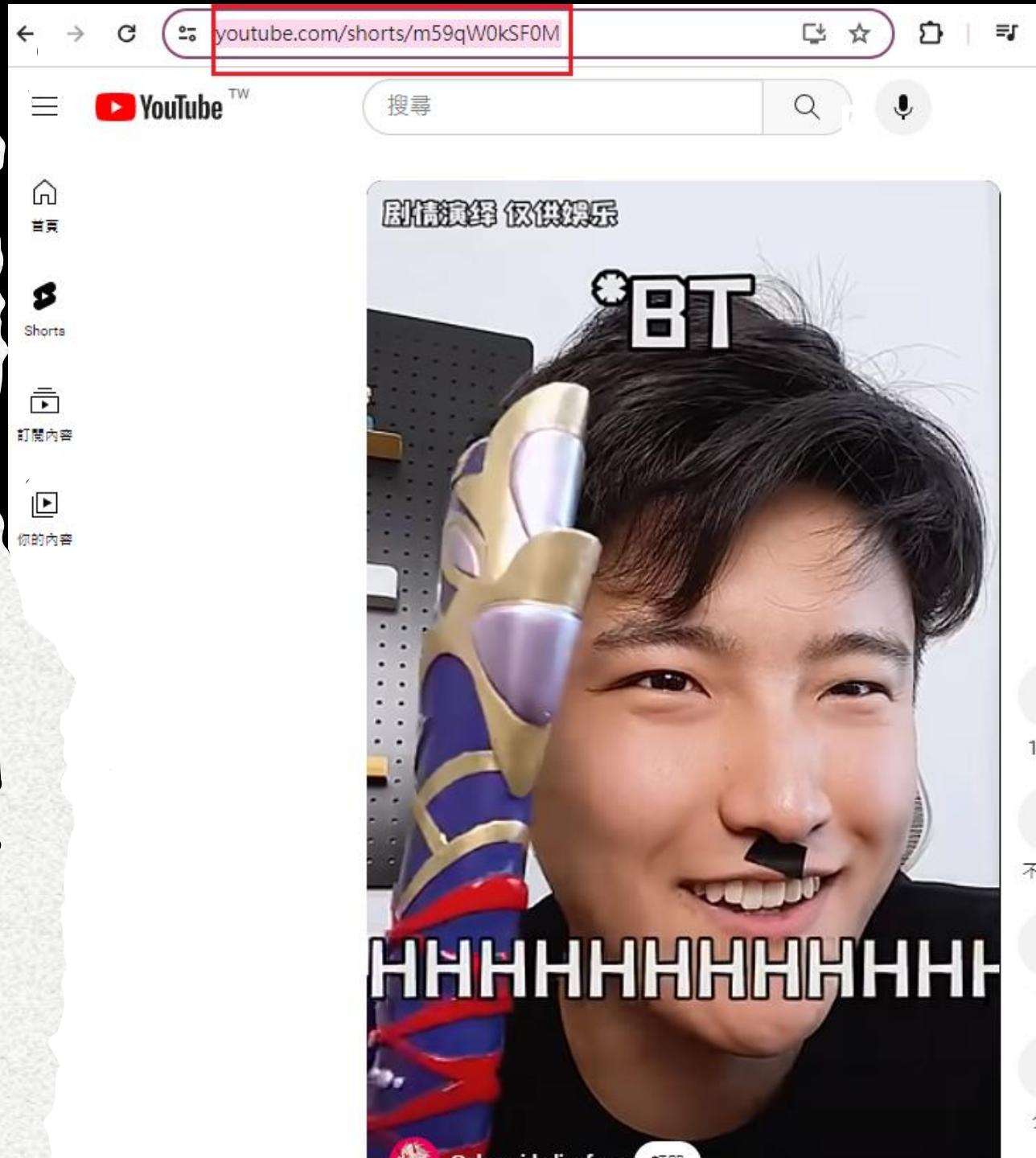
伺服器 Apache, Nginx, IIS ...

資料儲存 Database, Cache, File Storage

運作環境 OS(Linux/Windows), Cloud, Container



大家上網的時候有注意過這個嗎



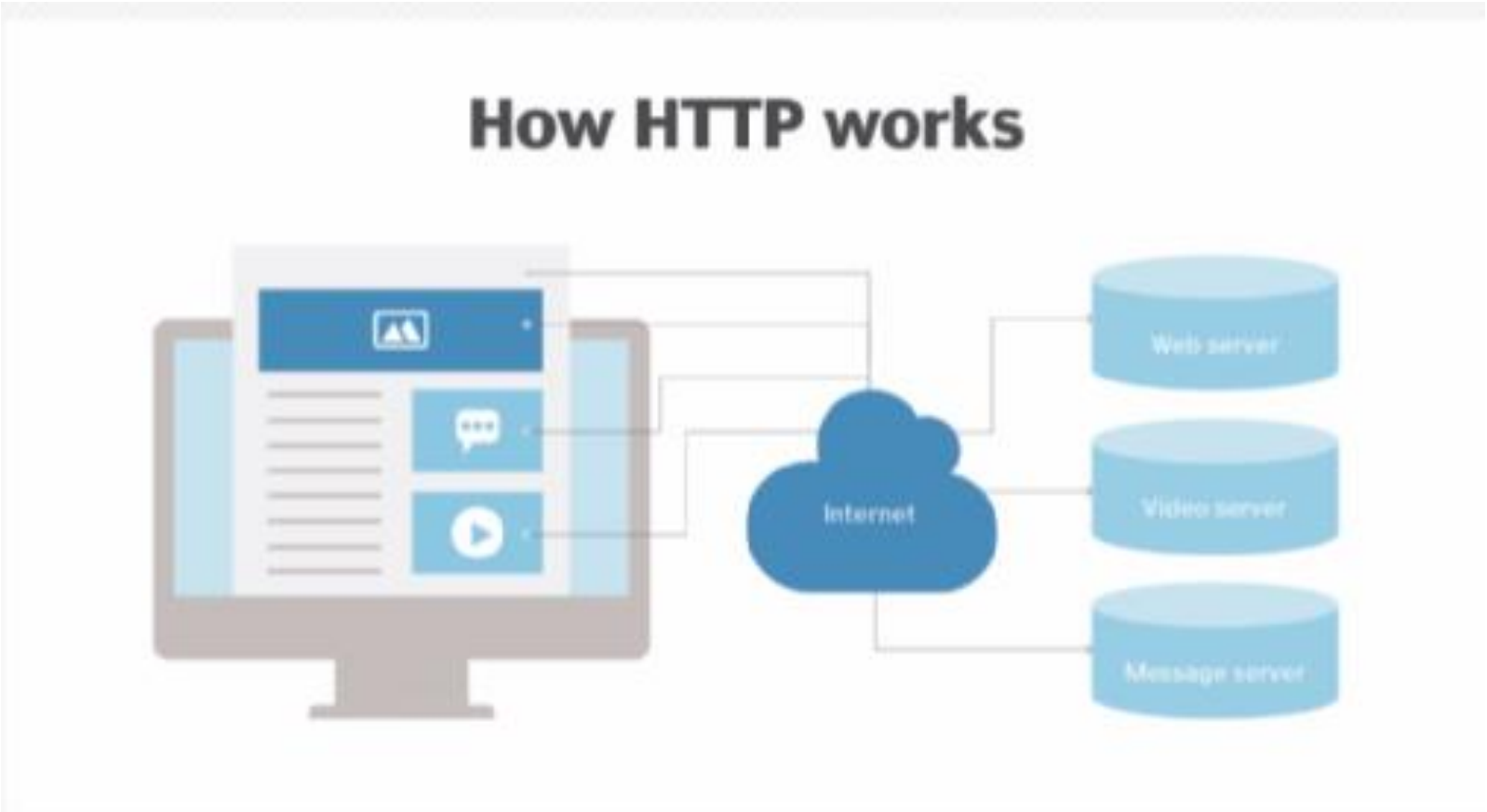
URL(全球資源定位器)


域名
https://www.cadiis.com.tw/
安全協定
子網域
網域分類類型
網址申請
所在國家
域名
https://www.cadiis.com.tw/portfolio/wowprime
安全協定
子網域
網域
分類
類型
網址申請
所在國家
路徑名稱
?password=123&user=naup
參數

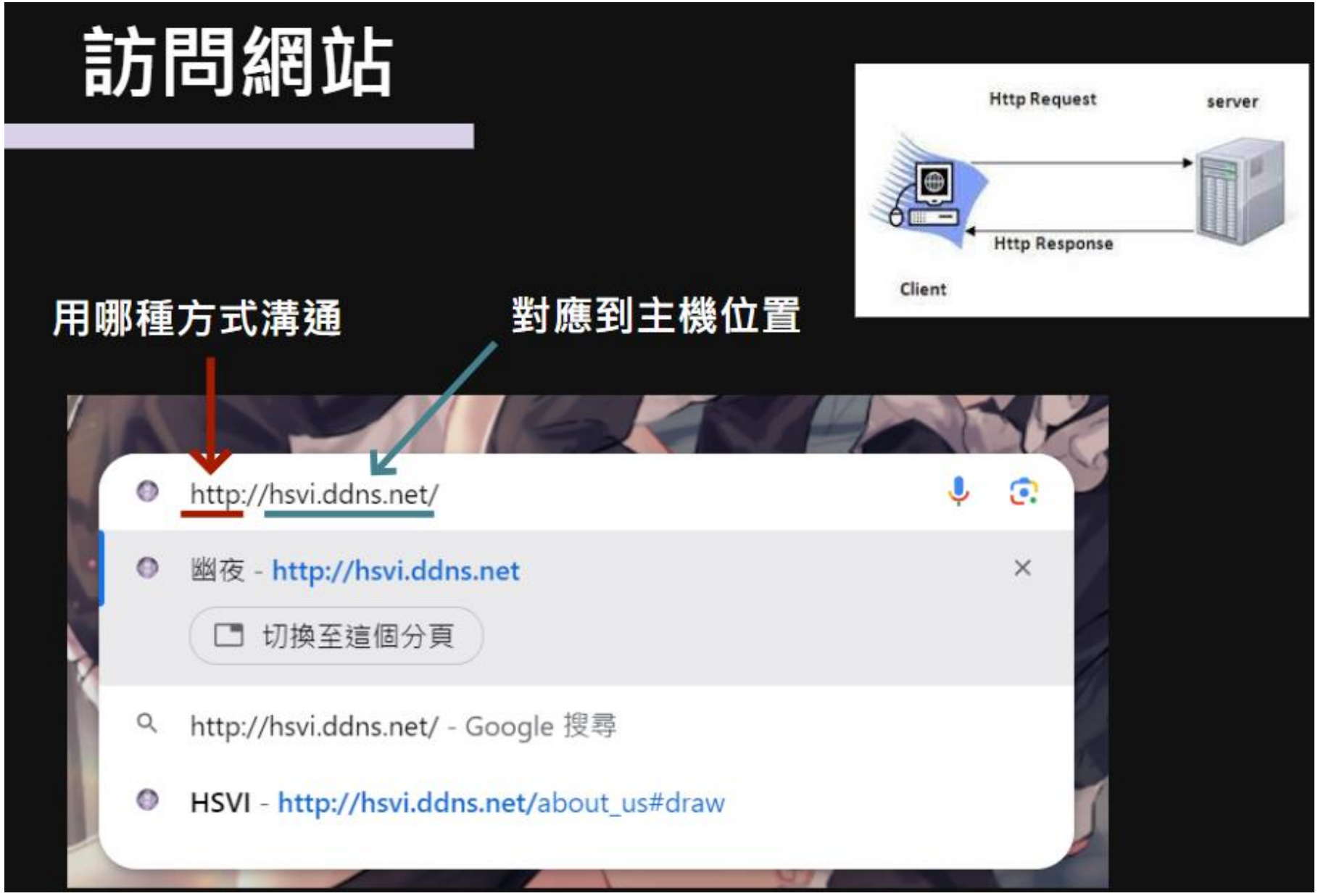
URL參數說明

- 協定:用甚麼協定溝通
- 域名:資源所在的主機名稱
- 路徑:主機上資源所在的位置
- 參數:GET請求時傳遞給server的資料
- 子網域:由你註冊網域延伸出來，在同個網域創建不同的子網域可以用來區分更多的服務或是不同的網站

HTTP(HyperText Transfer Protocol 超文本傳輸協定)



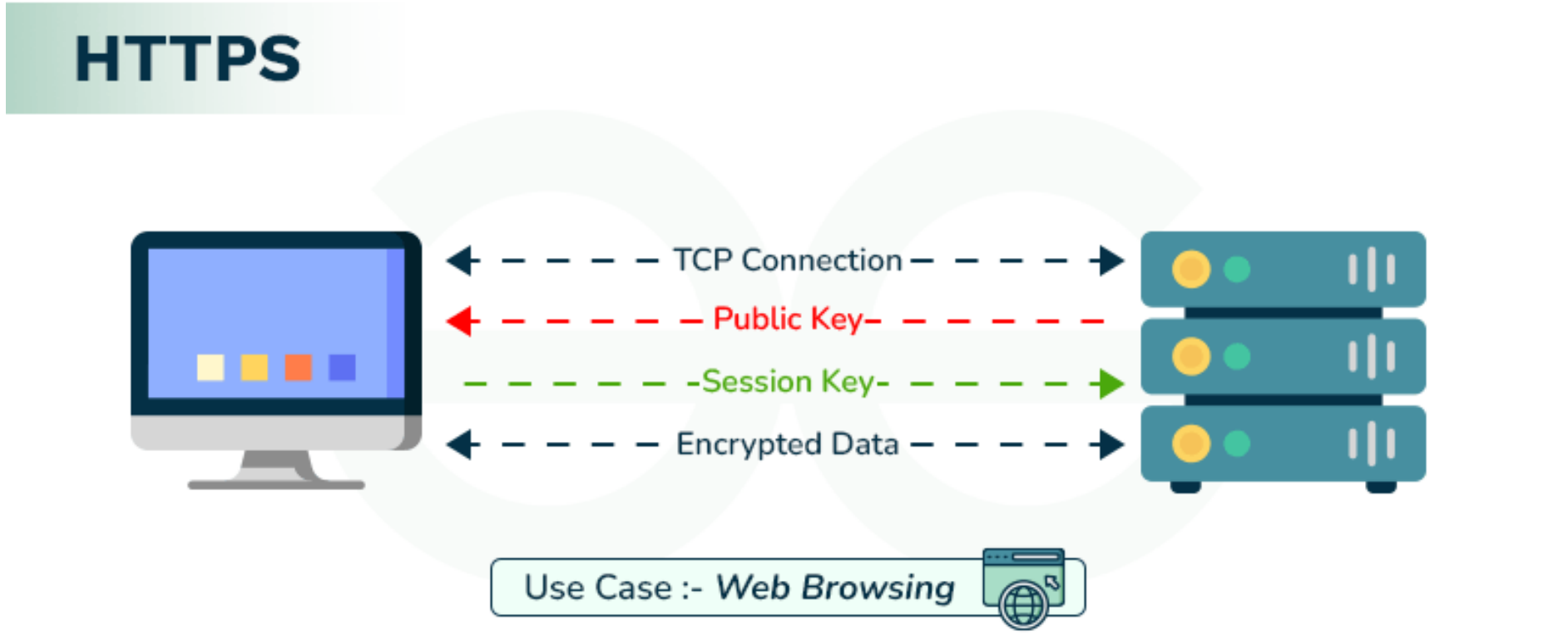
為甚麼電腦間需要溝通?


訪問網站
Http Request
server
Http Response
Client
用哪種方式溝通
對應到主機位置
http://hsvi.ddns.net/
幽夜 - http://hsvi.ddns.net
切換至這個分頁
http://hsvi.ddns.net/ - Google 搜尋
HSVI - http://hsvi.ddns.net/about_us#draw

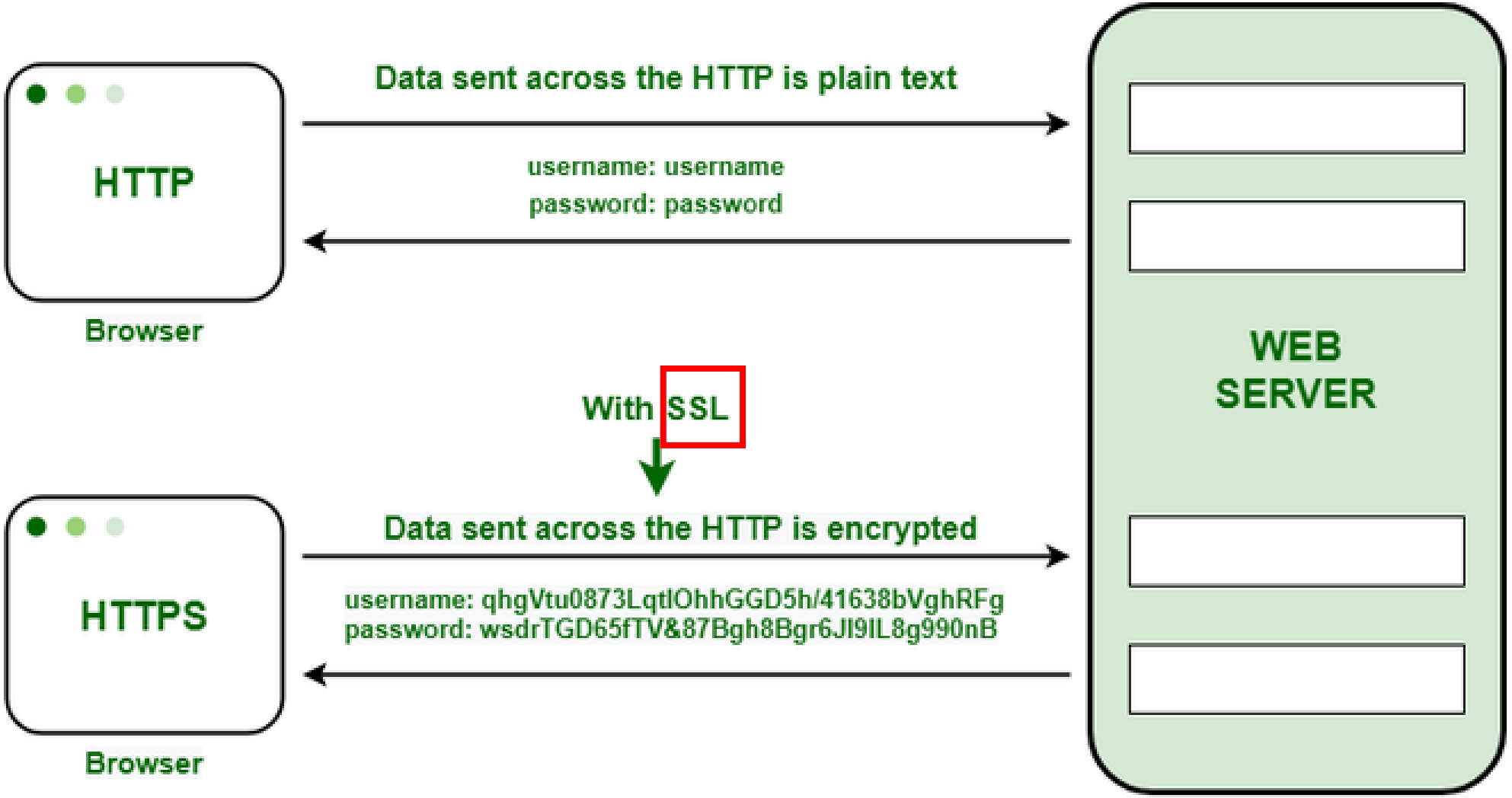
HTTP 不安全的理由

- HTTP 的數據是明文傳遞
- HTTP 沒有身份驗證的功能
- HTTP 沒有可靠的驗證內容的方法

什麼是HTTPS


HTTPS
TCP Connection
Public Key
Session Key
Encrypted Data
Use Case :- Web Browsing

HTTPS是如何工作的



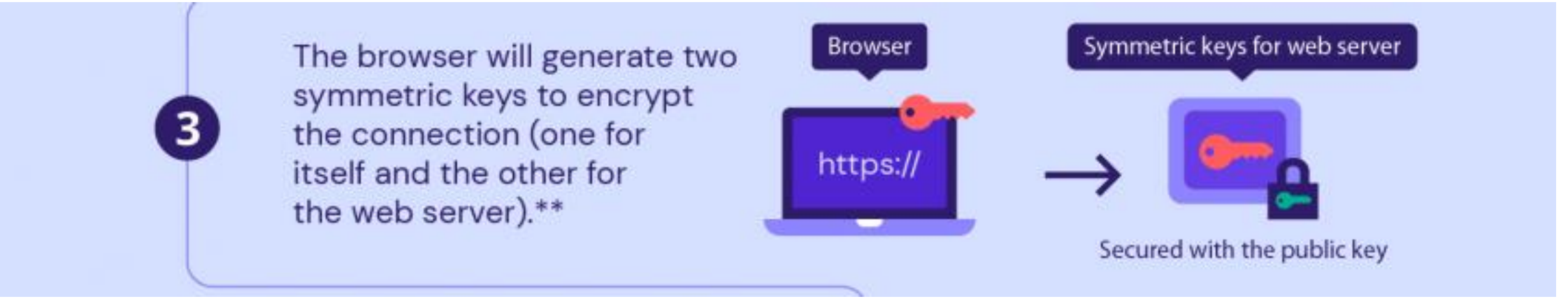
SSL(Secure Socket Layer)

- 確保瀏覽器直接與所需的伺服器通信。
- 確保只有通信系統才能訪問它們交換的消息。

網站擁有者從證書頒發機構（**CA**）購買 **SSL** 證書並將其安裝在其網站上

當訪問者瀏覽網站時，瀏覽器和 **Web** 伺服器使用稱為 **SSL** 握手的方法建立 **SSL** 連接。

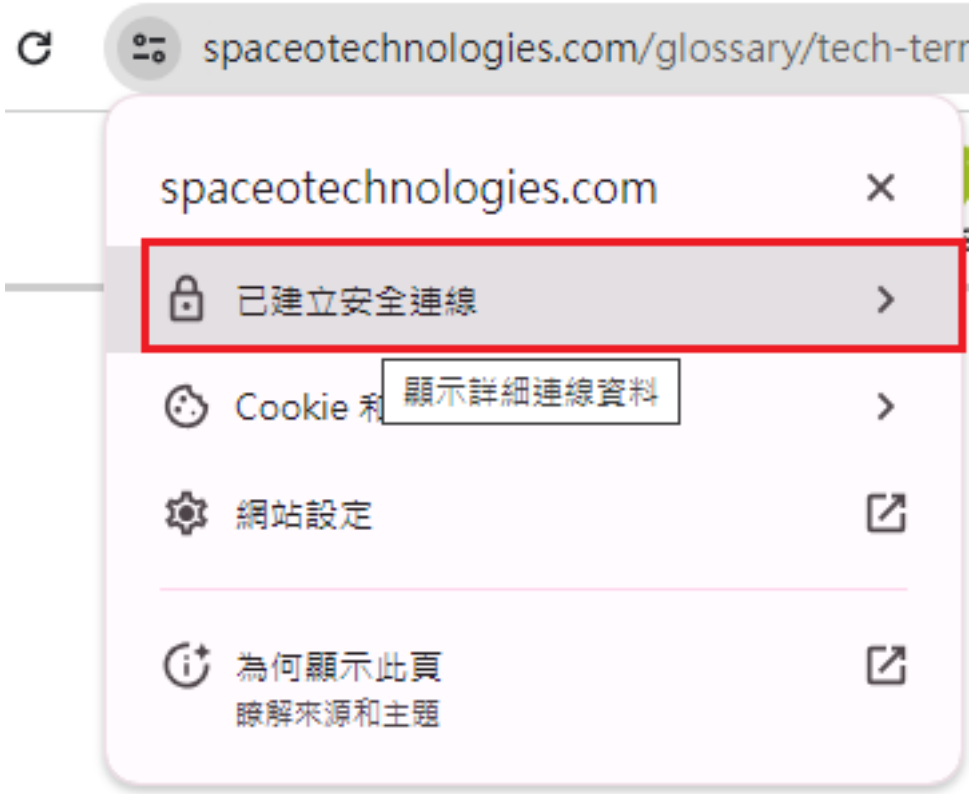
在**SSL**握手期間，瀏覽器要求伺服器提供其**SSL**證書和公鑰以證明其有效性。



驗證證書后，瀏覽器和 **Web** 伺服器將交換私鑰和公鑰以建立對稱會話金鑰。

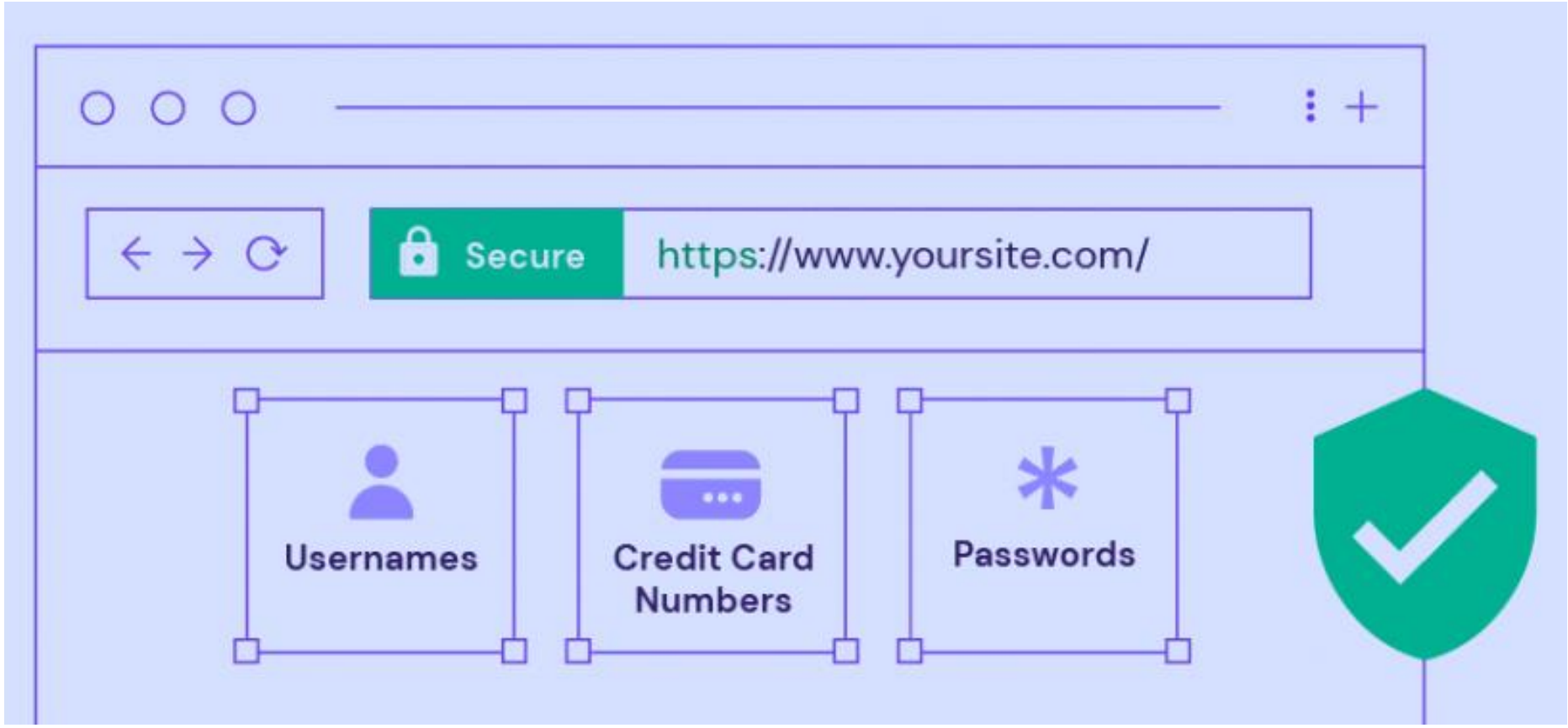
雙方使用此對稱密鑰對所有通信進行加密。此密鑰將在有限的時間內保持有效，並且僅對該特定會話有效

瀏覽器SSL


spaceotechnologies.com/glossary/tech-terr
spaceotechnologies.com
已建立安全連線
Cookie 和
顯示詳細連線資料
網站設定
為何顯示此頁
瞭解來源和主題


spaceotechnologies.com/glossary/tech-terms/what-is-ssl/
安全性
spaceotechnologies.com
已建立安全連線
你傳送給這個網站的資訊 (例如密碼或信用卡號碼) 不會外洩。 瞭解詳情
憑證有效
顯示憑證 (核發者：GTS CA 1P5)
憑證檢視者：spaceotechnologies.com
一般(G)
詳細資訊(D)
核發對象
一般名稱 (CN) spaceotechnologies.com
組織 (O) <不是憑證的一部分>
組織單位 (OU) <不是憑證的一部分>
發行者
一般名稱 (CN) GTS CA 1P5
組織 (O) Google Trust Services LLC
組織單位 (OU) <不是憑證的一部分>
有效期間
發行日期 2024年3月21日 星期四 上午8:59:20
到期日 2024年6月19日 星期三 上午8:59:19
SHA-256 指紋
憑證 0158e1fc96966d52a702a9e648a4152bf9a85cff4ba8a422b954d8b91a9cf6af
公開金鑰 511b17924393a6a914c0c1f57288280c5102cad1b779b059135503efbacbefe7
Home

何時以及為什麼必須使用 **SSL**？

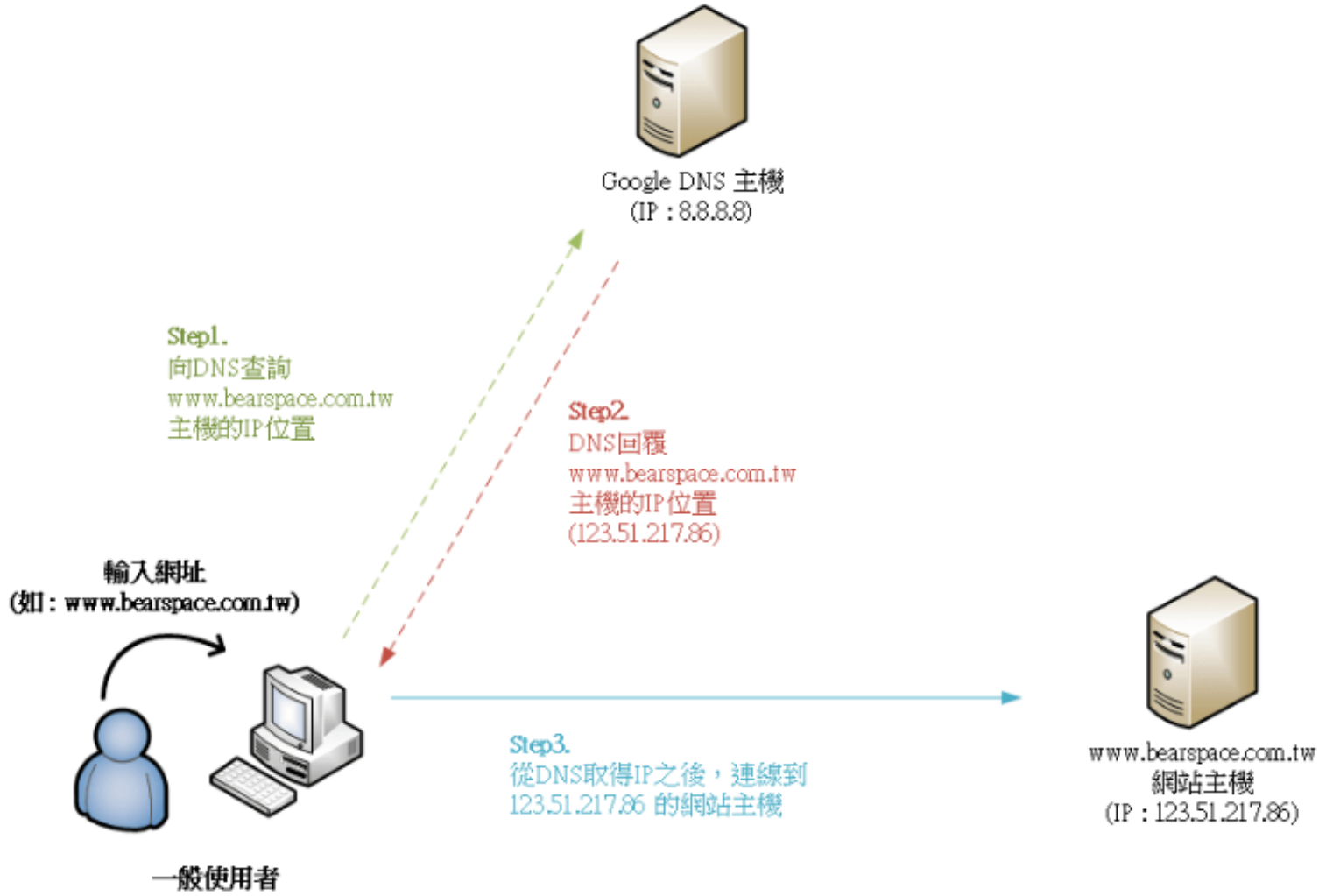


如何將**SSL**添加到您的網站

1. 選擇受信任的證書頒發機構。選擇可靠且值得信賴的SSL供應商，例如Let's Encrypt，DigiCert或Comodo。
2. 生成證書簽名請求（CSR）。使用 Microsoft Internet Information Services（IIS）、Apache 或 cPanel 生成 CSR。此檔案包含您的公鑰、功能變數名稱和組織資料。
3. 上傳 CSR。之後，將您的 CSR 檔上傳到選定的證書頒發機構，該證書頒發機構將進行背景調查並頒發簽名證書。
4. 上傳 CSR。之後，將您的 CSR 檔上傳到選定的證書頒發機構，該證書頒發機構將進行背景調查並頒發簽名證書。

DNS(domain name server)

提供域名和 IP address 互相轉換

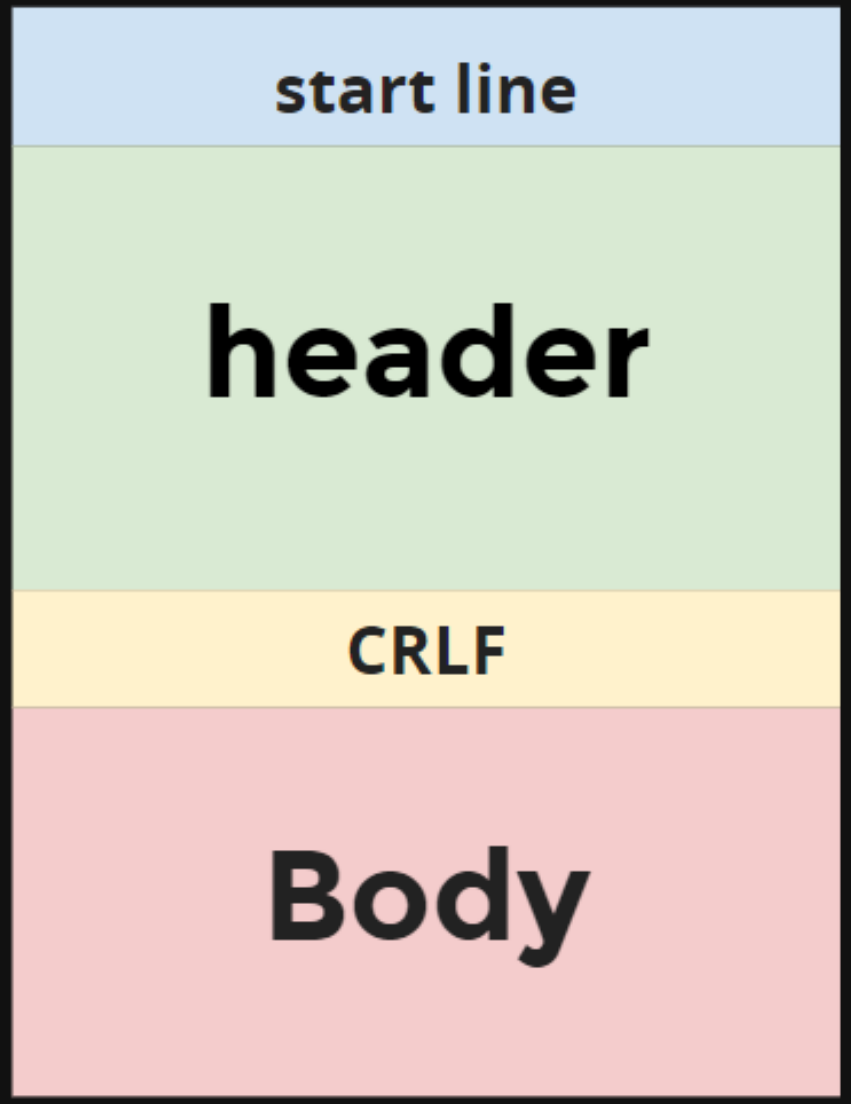


request & response

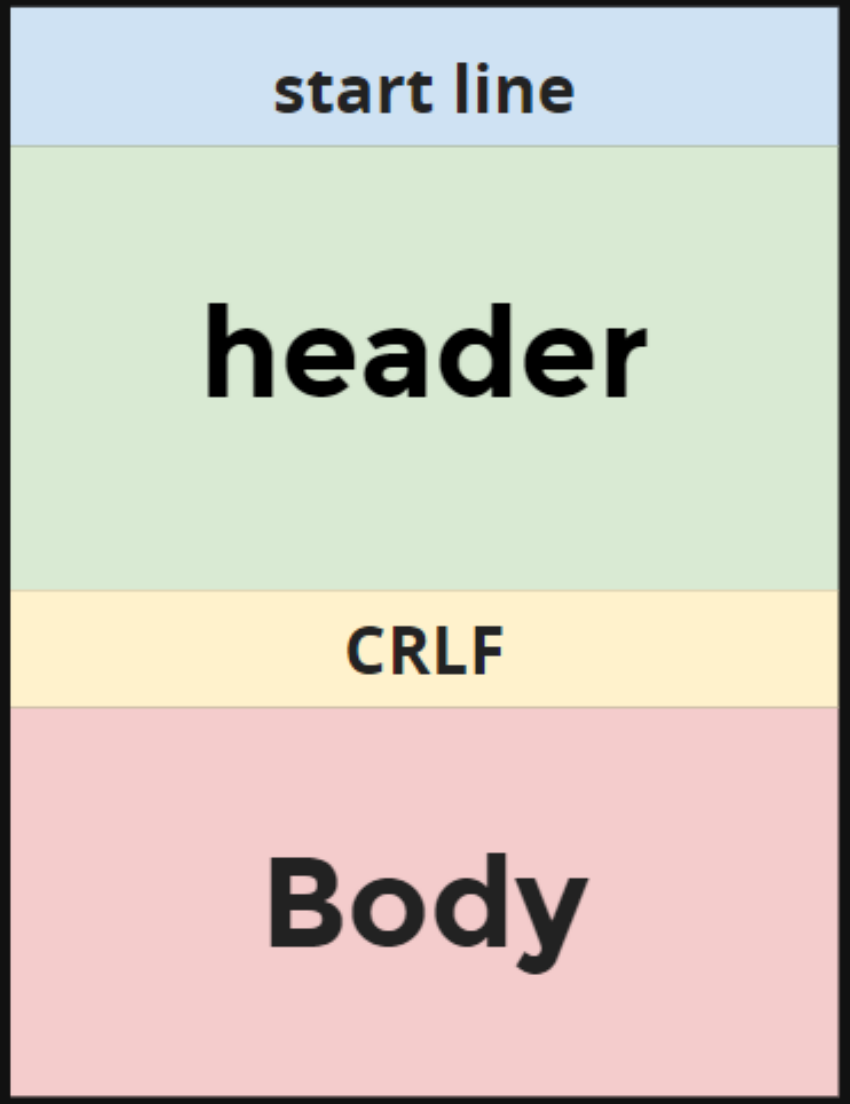
	明信片	查看方式	資料量	安全性
GET		直接看	少	相對差
POST	 包裹	打開本體後看	多	相對好

request & response

Request格式



Response格式



request

```
POST /?id=1 HTTP/1.1
Host: echo.paw.cloud
Content-Type: application/json; charset=utf-8
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0) Gecko/20100101 Firefox/53.0
Connection: close
Content-Length: 136

{
  "status": "ok",
  "extended": true,
  "results": [
    {"value": 0, "type": "int64"},
    {"value": 1.0e+3, "type": "decimal"}
  ]
}
```

response

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Date: Sat, 18 Feb 2017 00:01:57 GMT
Server: nginx/1.11.8
transfer-encoding: chunked
Connection: Close

<!doctype html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>echo</title>
....略
```

start-line

- 因應請求、回應，分別稱為
  - **請求行 request-line**
  - **狀態行 status-line**

request-line
方法 (method)

- GET：向指定的資源發出「顯示」請求。
- POST：向指定資源提交資料，並且Body中可帶傳輸的資料。
- PUT：上傳或取代指定的資源。
- PATCH : 覆蓋資料
- DELETE：刪除指定的資源。
- HEAD：與GET相似，但只會取得標頭與HTTP狀態。
- OPTIONS：回傳這個伺服器支援的所有HTTP Method。
- TRACE：回傳收到的請求內容。

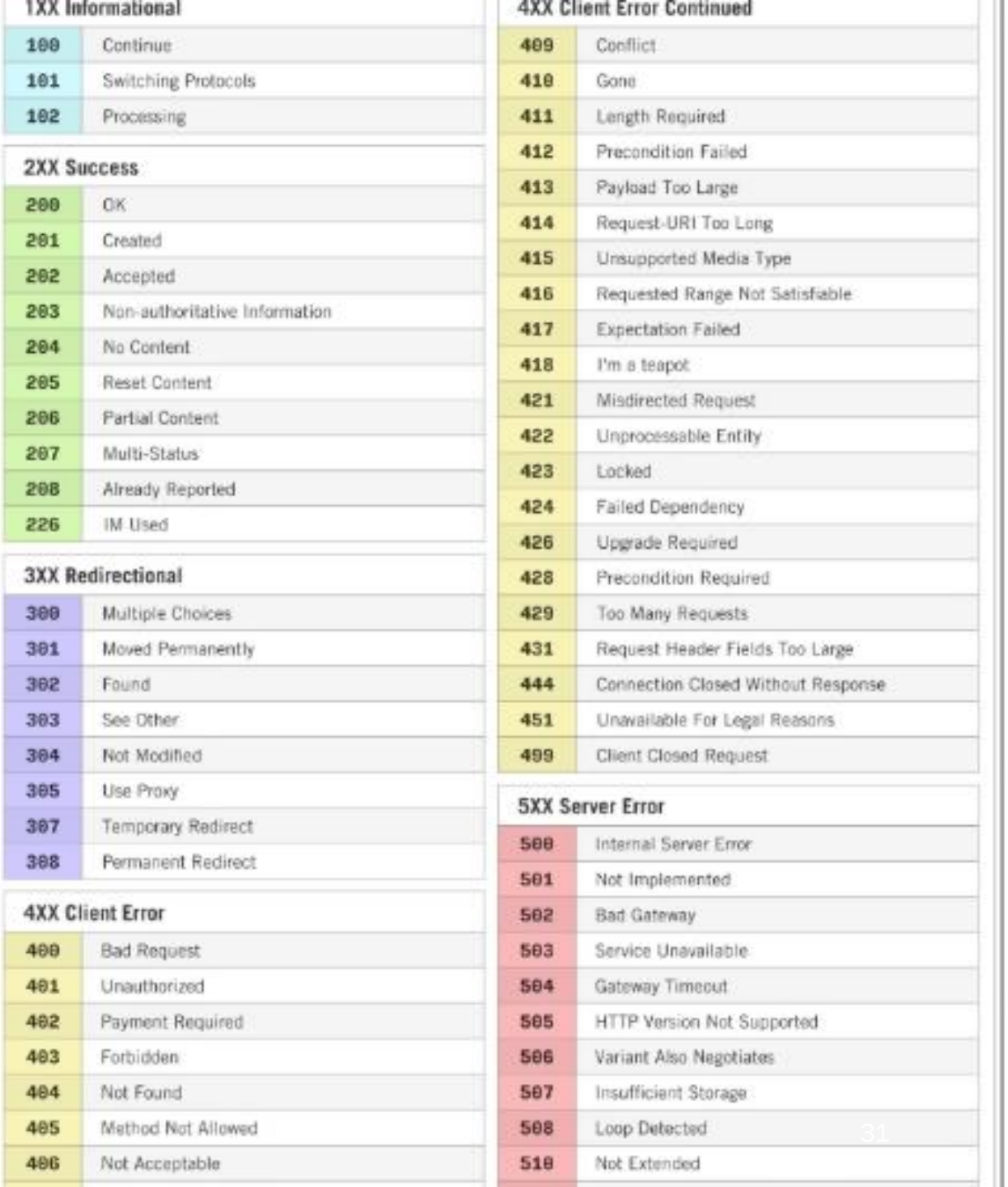
request-line
HTTP-version

- HTTP 版本
  - Ex : HTTP/1.1、HTTP/1.0

status-line
HTTP 版本 (**HTTP-version**)、
狀態碼 (**status-code**)、
原因短語 (**reason-phrase**)

status-code

- 1xx – 連接正在進行中
- 2xx – 請求成功完成，伺服器給了瀏覽器預期的響應
- 3xx –這個請求被收到了，但是需要重新定向
- 4xx –客戶端看起來可能發生了錯誤，妨礙了伺服器的處理
- 5xx – 客戶端的請求是有效的，但伺服器



1XX Informational	
100	Continue
101	Switching Protocols
102	Processing

2XX Success	
200	OK
201	Created
202	Accepted
203	Non-authoritative Information
204	No Content
205	Reset Content
206	Partial Content
207	Multi-Status
208	Already Reported
226	IM Used

3XX Redirectional	
300	Multiple Choices
301	Moved Permanently
302	Found
303	See Other
304	Not Modified
305	Use Proxy
307	Temporary Redirect
308	Permanent Redirect

4XX Client Error	
400	Bad Request
401	Unauthorized
402	Payment Required
403	Forbidden
404	Not Found
405	Method Not Allowed
406	Not Acceptable

4XX Client Error Continued	
409	Conflict
410	Gone
411	Length Required
412	Precondition Failed
413	Payload Too Large
414	Request-URI Too Long
415	Unsupported Media Type
416	Requested Range Not Satisfiable
417	Expectation Failed
418	I'm a teapot
421	Misdirected Request
422	Unprocessable Entity
423	Locked
424	Failed Dependency
426	Upgrade Required
428	Precondition Required
429	Too Many Requests
431	Request Header Fields Too Large
444	Connection Closed Without Response
451	Unavailable For Legal Reasons
499	Client Closed Request

5XX Server Error	
500	Internal Server Error
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout
505	HTTP Version Not Supported
506	Variant Also Negotiates
507	Insufficient Storage
508	Loop Detected
510	Not Extended

整理一下

- 200 -> 一切都正常
- 301 -> 永久性轉址
- 302 -> 暫時性轉址
- 404 -> 該網頁找不到
- 500 -> 內部服務器錯誤

header

- 紀錄各種資料/設定，等基本信息
- 可以想像是包裹外貼的那張紙

```
1 Host: echo.paw.cloud
2 Content-Type: application/json; charset=utf-8
3 User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.12; rv:53.0) Gecko/20100101 Firefox/53.0
4 Connection: close
5 Content-Length: 136
```

Host

要訪問的目標主機的域名或IP

Content-Type

指示請求中攜帶的數據類型(上方的是JSON格式、UTF-8編碼)

User-Agent

發送請求的客戶端應用或瀏覽器信息("Mozilla Firefox"瀏覽器發送，版本是53.0，在Macintosh操作系統上)

Content-Length

後續要讀取消息的長度

Connection

暗示連結類型(close表示請求完後連結會關閉)

CRLF
(carriage return followed by line feed)

- \r\n
- 網際網路嚴謹的換行標準

CRLF
(carriage return followed by line feed)

CR = 回車 (Carriage Return)

- 許多語言表示為: \r
- Unicode 中為 0x0D (十進制的 13)

LF = 換行 (Line Feed)

- 許多語言表示為: \n
- Unicode 中為 0x0A (十進制的 10)

雖然多數語言 \n 就能做到換行，規範上，仍以 \r\n 為主。

CR跟LF差異

- 在不同作業系統中，CR 和 LF 的組合方式不同
  - Windows -> CRLF
  - Unix/Linux -> LF

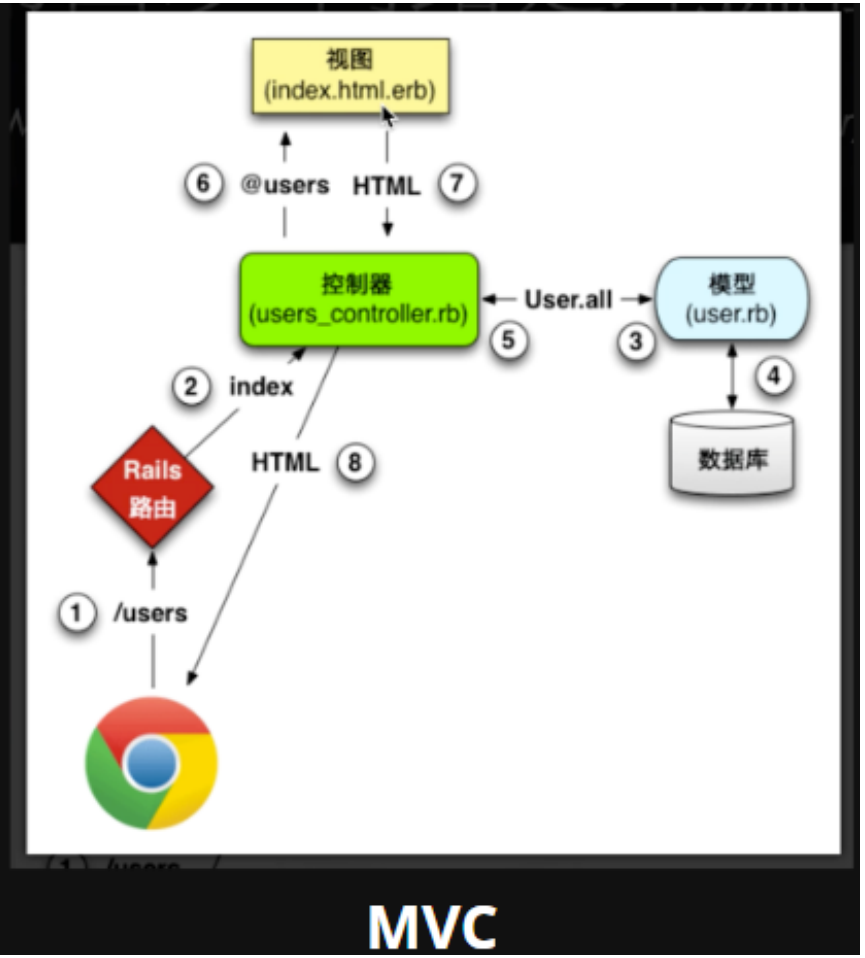
CR通常表示將游標移動到當前行的起始位置，但不會移動到下一行
LF符號通常表示移動游標到下一行的起始位置，實現文本的換行

body

訊息主體，像包裹的箱子，是訊息中乘載資料的地方。

從發出請求到連上網站

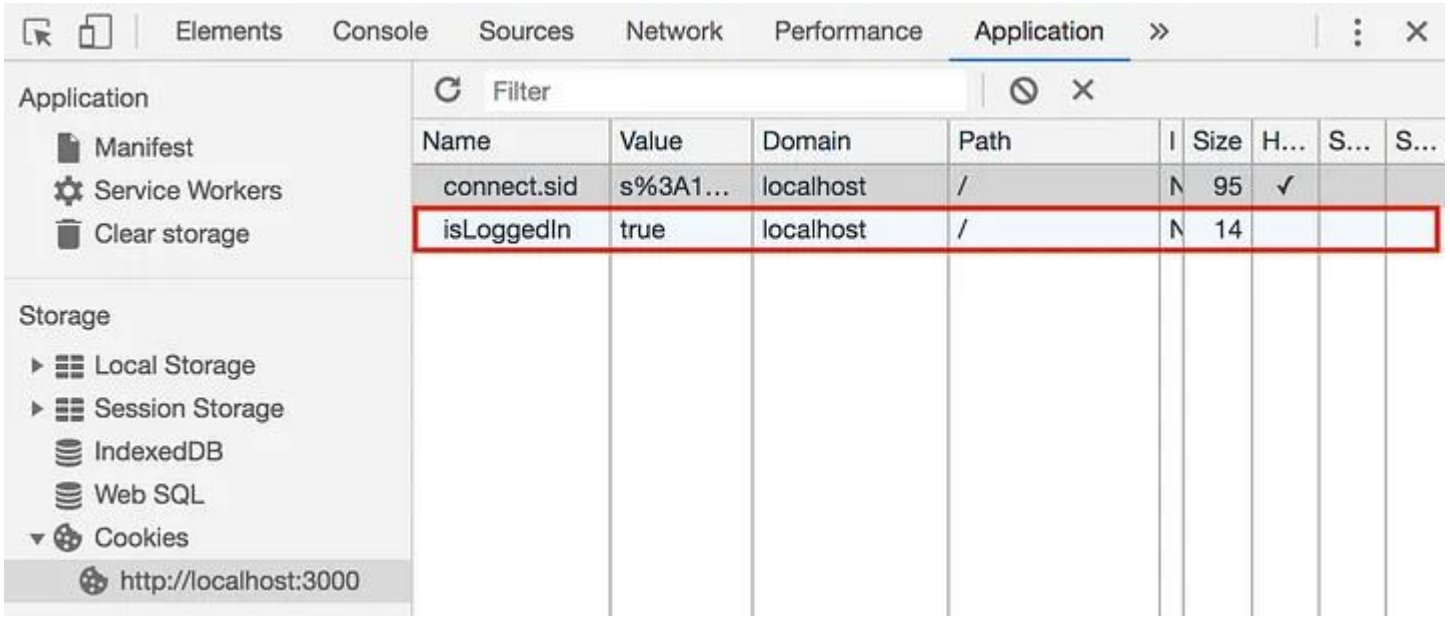
1.在瀏覽器上輸入網址
2.DNS域名解析
3.用解析出來的IP連上對應的主機
4.server處理請求(MVC...)
5.回傳頁面

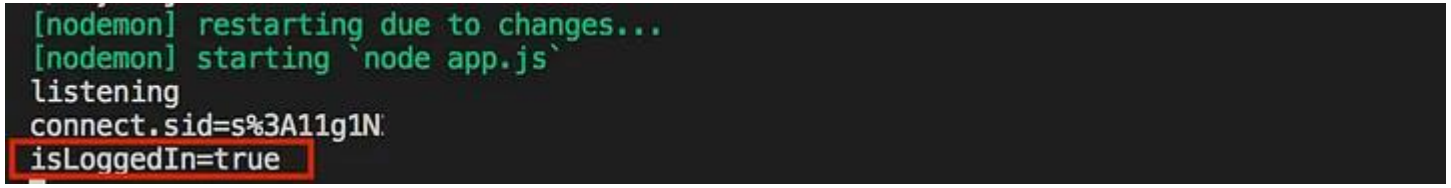


cookie

- Cookie是指某些網站為了辨別使用者的身分而在用戶端瀏覽器上存儲的一些小型文本檔案

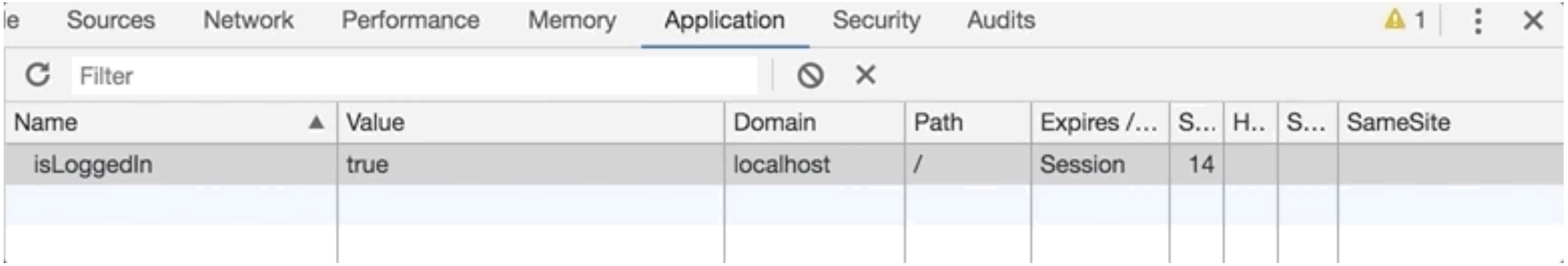
Cookie 不安全


Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
http://localhost:3000
Filter
Name
Value
Domain
Path
Size
connect.sid
s%3A1...
localhost
/
95
isLoggedIn
true
localhost
/
14


[nodemon] restarting due to changes...
[nodemon] starting `node app.js`
listening
connect.sid=s%3A11g1N
isLoggedIn=true

儘量避免將敏感資訊透過 **Cookie** 存在客戶端

用戶是有機會可以在瀏覽器中修改，讓伺服器收到不正確的訊息，讓伺服器誤以為使用者已經通過認證

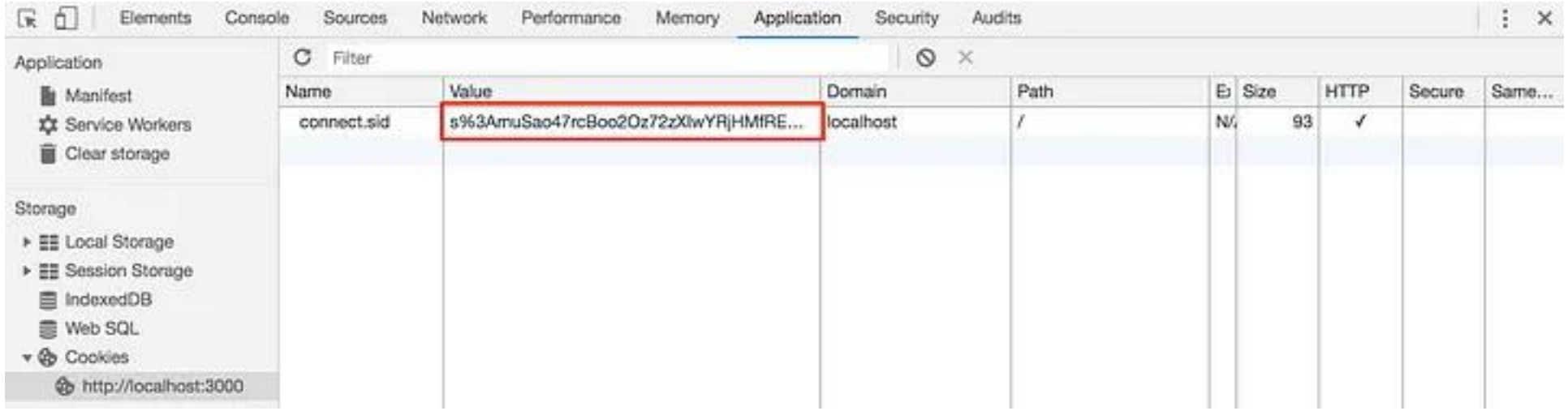


session


3. 客戶端在往後對伺服器發送的請求都附上含有 Session ID 的 Cookie
以利伺服器找到含有使用者「已經登入」狀態資訊的 Session
1. 透過登入頁面登入
客戶端
伺服器
2. 建立 Session，將使用者「已經登入」的狀態資訊存在 Session 中
並把相對應的 Session ID 存放在 Cookie 中回傳
Session

hashed Session ID — 用來讓伺服器辨識和找尋相對應的 **Session**

- hashed session ID 可以想像是把你提供的 secret 字串透過特殊的雜湊演算法（hashing algorithm）產生的一組獨特 ID



Session 總結

- 重點一：使用者敏感的資訊將被存放在伺服器端，而不是客戶端
- 重點二：伺服器會將一個獨特的 Session ID 附在回傳的 Cookie 給瀏覽器
- 重點三：有了 Session ID 同個用戶透過客戶端發出的「不同請求」給相同的伺服器時，伺服器都能辨識對方為相同的瀏覽器

編碼 & 加密 & 雜湊

感覺都是亂碼，沒什麼差

- TmF1cEpqaW4= (編碼->base64)
- jg7LzS3fpcH15kKScqUMdg==(加密->ECB)
- 9bb2508637df52b17523e4a4a9f727fca1923134a8ace76f0922 0a3b908e03d2(雜湊->sha256)

編碼(encode)

- 編碼並不會修改資料、也沒有任何加密的效果，單純就是換個方式來呈現

base64

- 基於64個可列印字元來表示

https://www.base64decode.org/

加密(encrypt)

密鑰:YourSecretKey123

加密後: jg7LzS3fpcH15kKScqUMdg==
解密後: NaupJjin

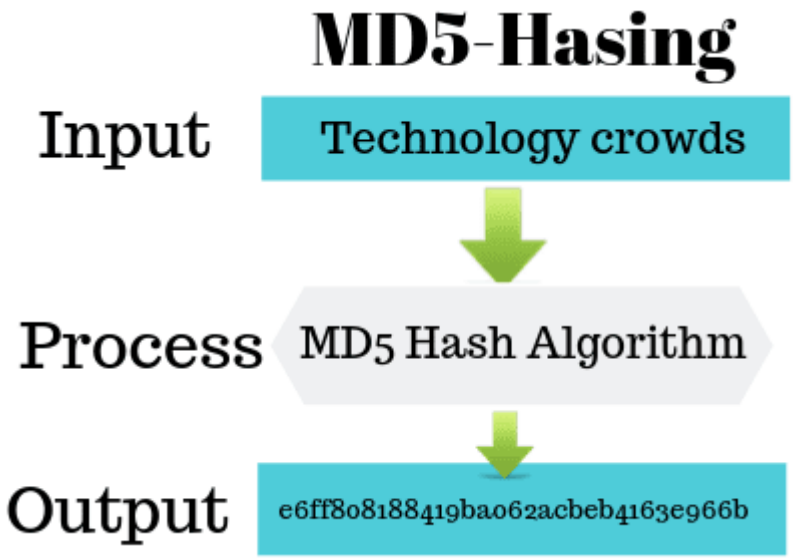
```python
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad
import base64

def encrypt_ecb(key, plaintext):
    cipher = AES.new(key, AES.MODE_ECB)
    padded_plaintext = pad(plaintext.encode('utf-8'), AES.block_size)
    encrypted = cipher.encrypt(padded_plaintext)
    return base64.b64encode(encrypted).decode('utf-8')

def decrypt_ecb(key, ciphertext):
    cipher = AES.new(key, AES.MODE_ECB)
    encrypted_bytes = base64.b64decode(ciphertext.encode('utf-8'))
```

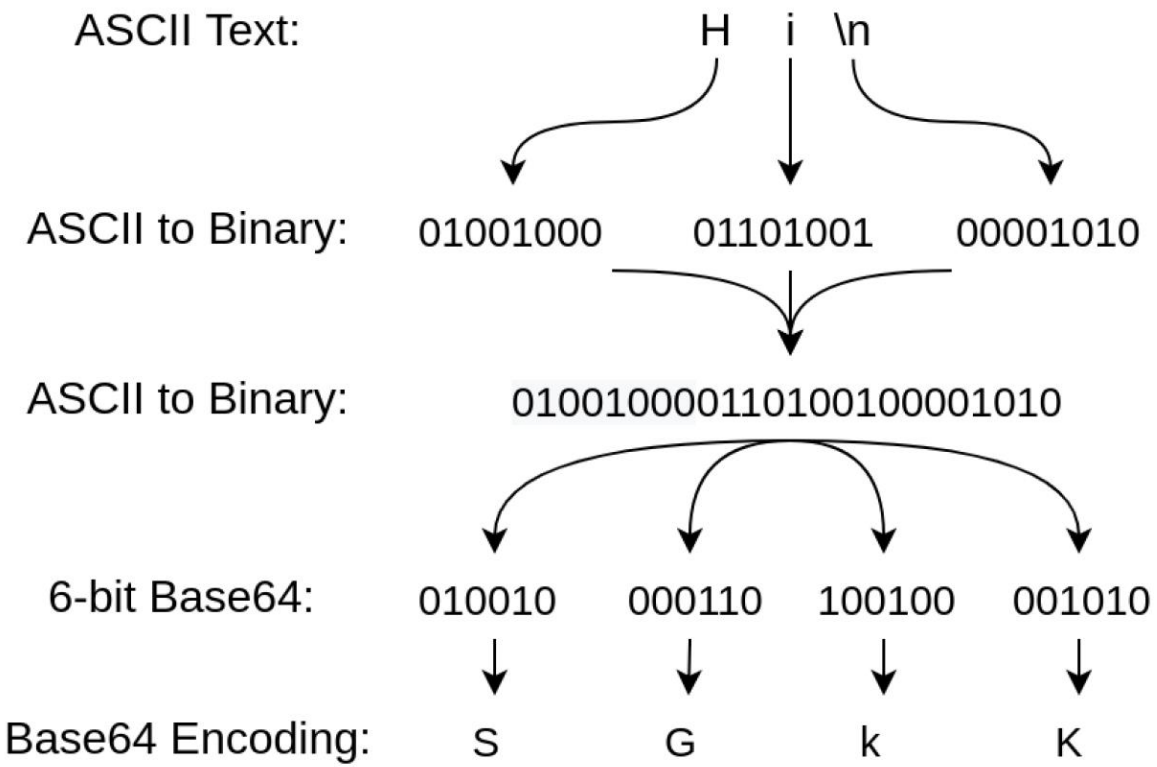
雜湊(hash)

- 由雜湊值是無法反推出原來的訊息
- 雜湊值必須隨明文改變而改變
- 明文相同出來的雜湊值相同

雜湊

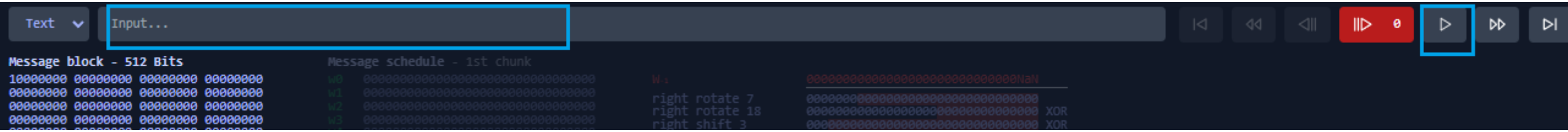
- MD5
- BASE64

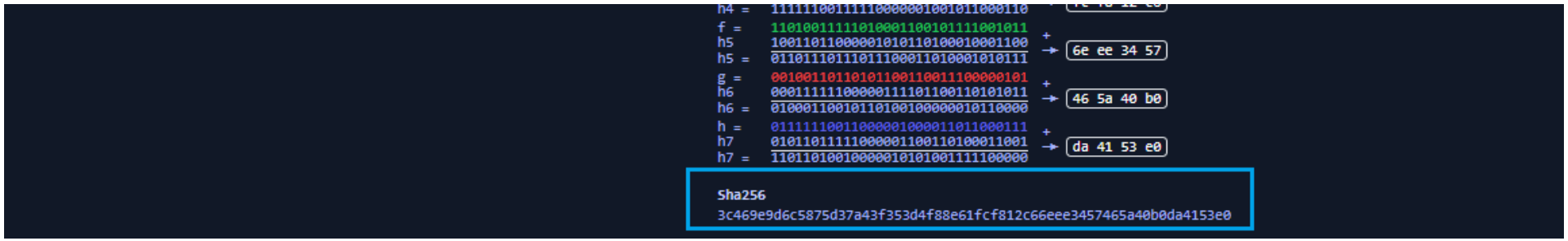




sha256 流程

- https://sha256algorithm.com/





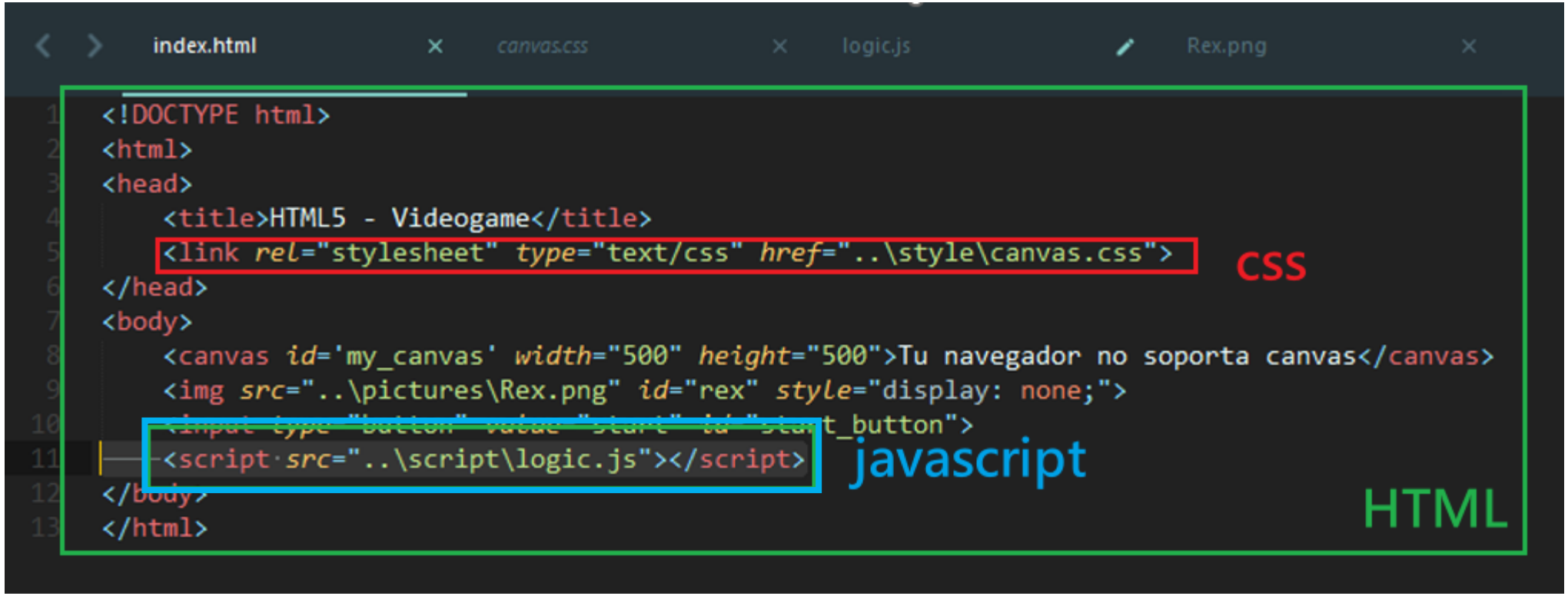
用sha256雜湊函式雜湊信息

```python
import hashlib

def calculate_sha256_hash(input_string):
    # 建立一個SHA-256雜湊物件
    sha256_hash = hashlib.sha256()

    # 將輸入的字串轉換為位元組，並更新雜湊物件
    sha256_hash.update(input_string.encode('utf-8'))

    # 取得計算後的雜湊值（以16進位表示）
    hash_value = sha256_hash.hexdigest()

    return hash_value

# 要計算雜湊的輸入字串
```

網頁基本語法

Html,css, javascript

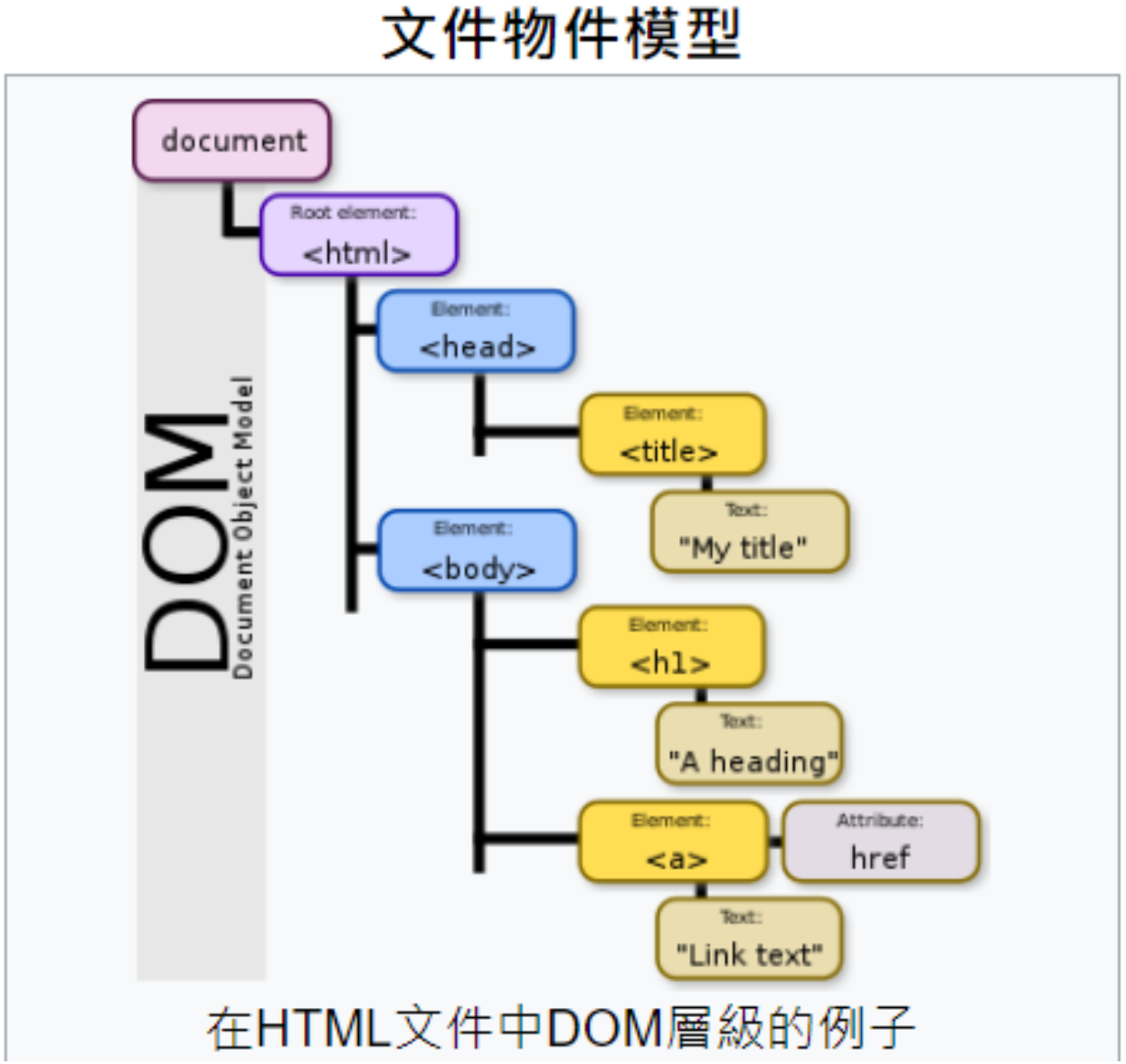
最簡單網頁架構


index.html
canvas.css
logic.js
Rex.png
<!DOCTYPE html>
<html>
<head>
<title>HTML5 - Videogame</title>
<link rel="stylesheet" type="text/css" href="..\style\canvas.css">
</head>
<body>
<canvas id='my_canvas' width="500" height="500">Tu navegador no soporta canvas</canvas>
<img src="..\pictures\Rex.png" id="rex" style="display: none;">
<script src="..\script\logic.js"></script>
</body>
</html>
CSS
javascript
HTML

Javascript最簡單的例子-alert

DOM架構

- Document Object Model的縮寫，也可以說是瀏覽器開放給JavaScript操作的API

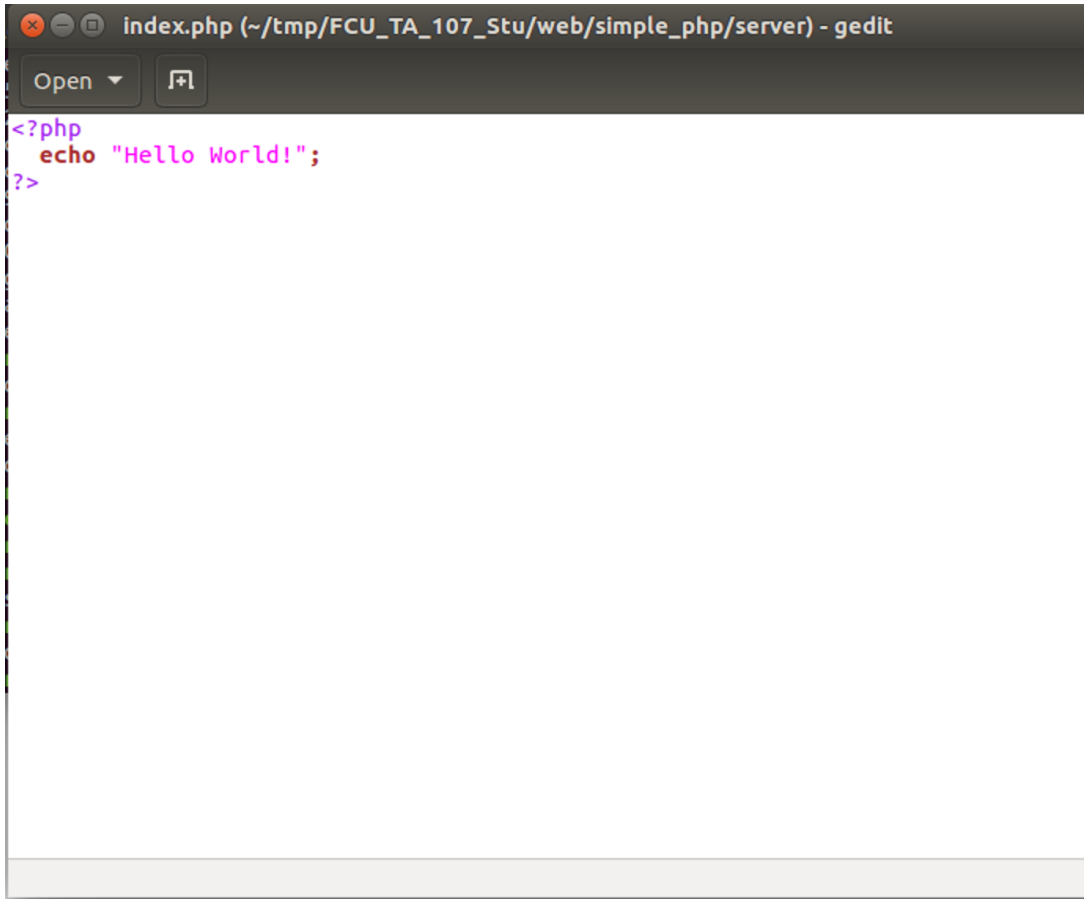


php

https://onlinephp.io/

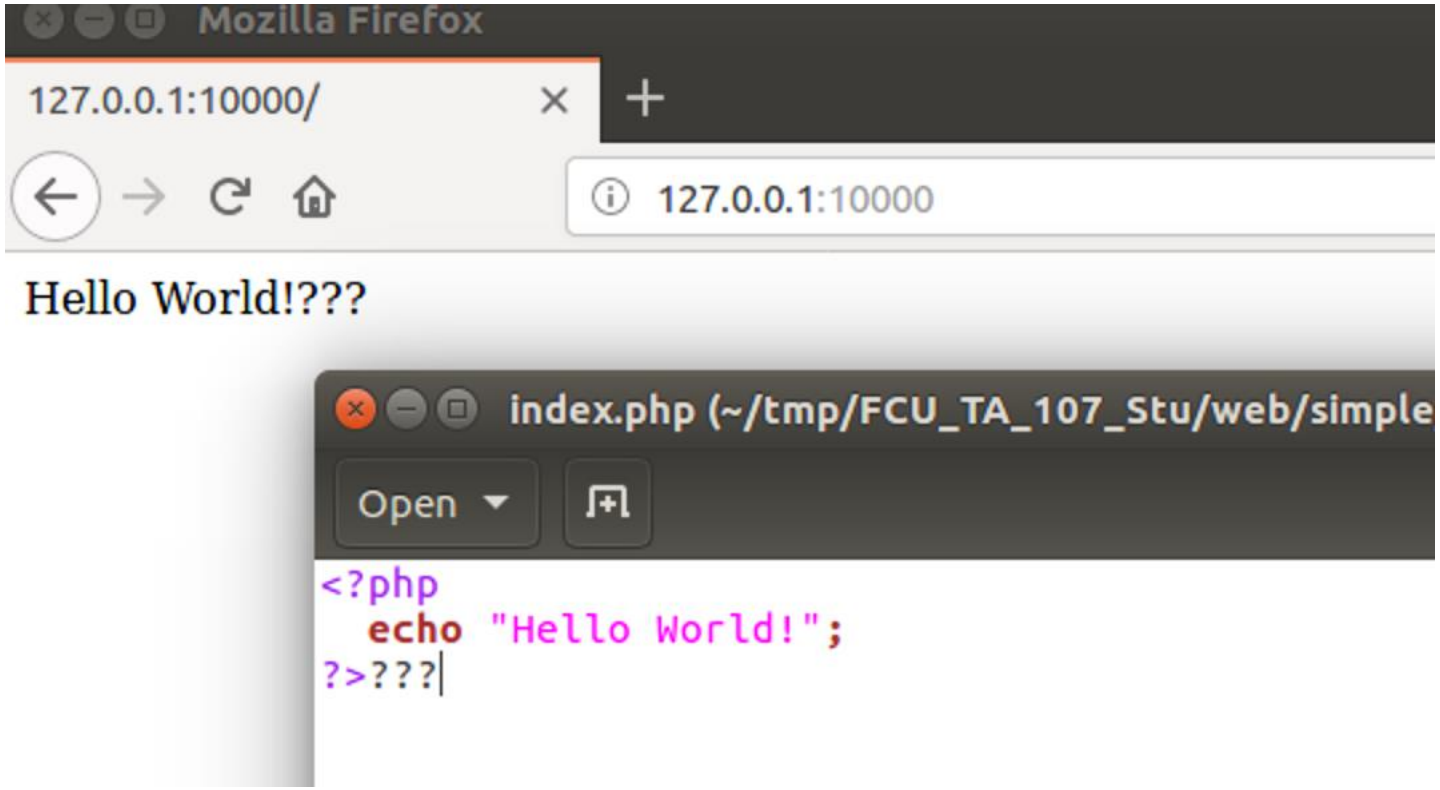
Php語法

- Php 被 <?php ?> 包起來

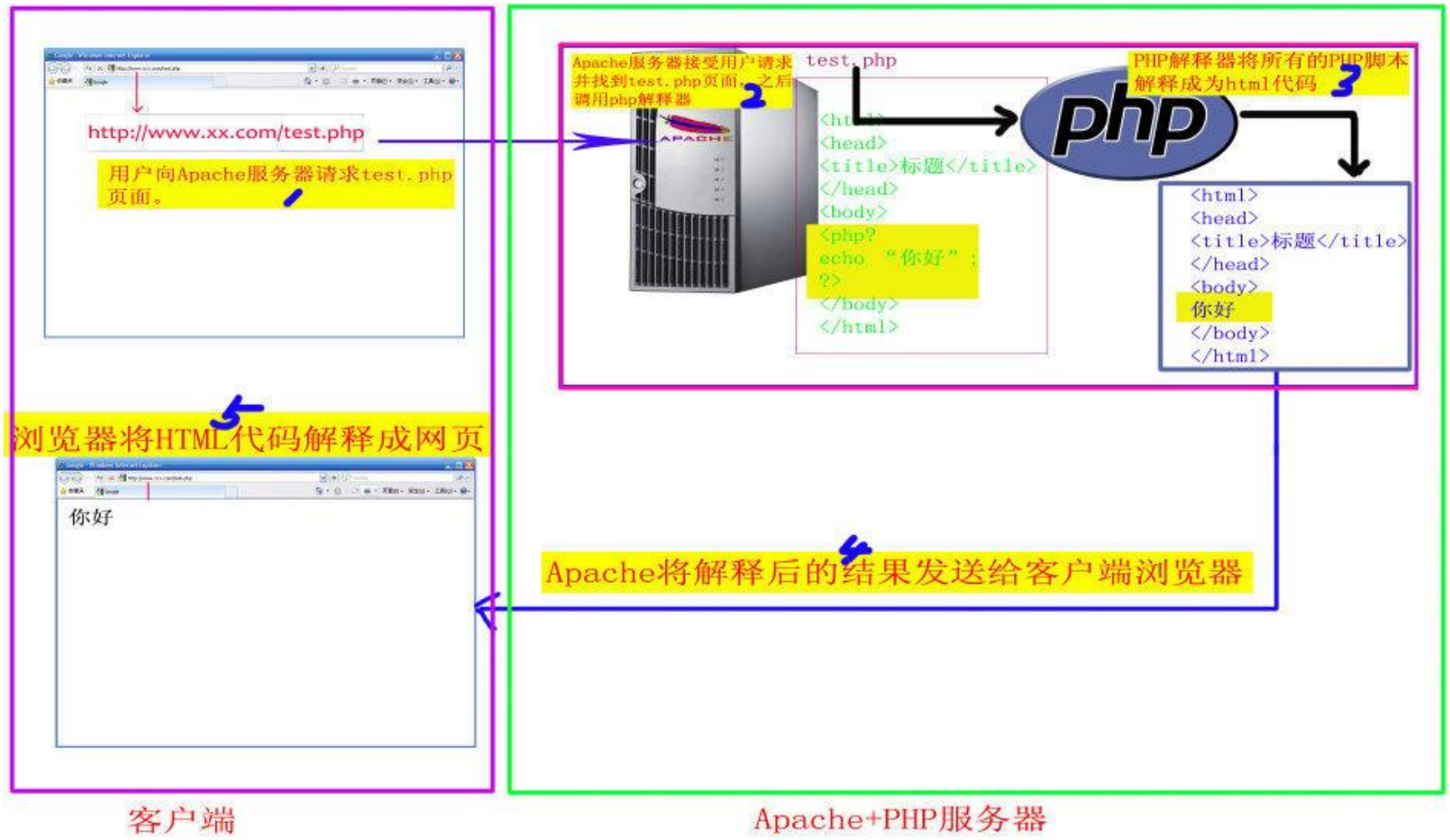


Php語法

- 沒有被標籤包起來，將直接列出不解析



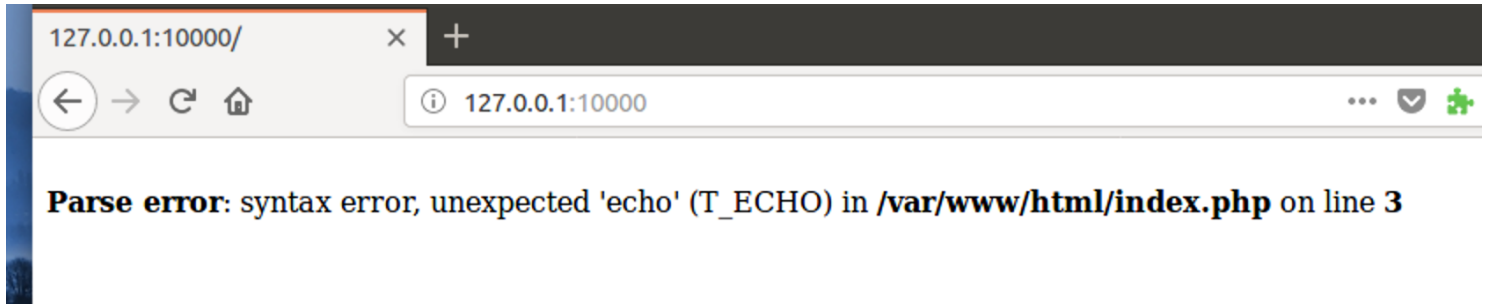
Php運行方式



圖源：https://www.jianshu.com/p/01e95d2418ad?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&ut

分號

- 每行要「;」
- 看到**Parse error**: syntax error, 通常就是沒「;」



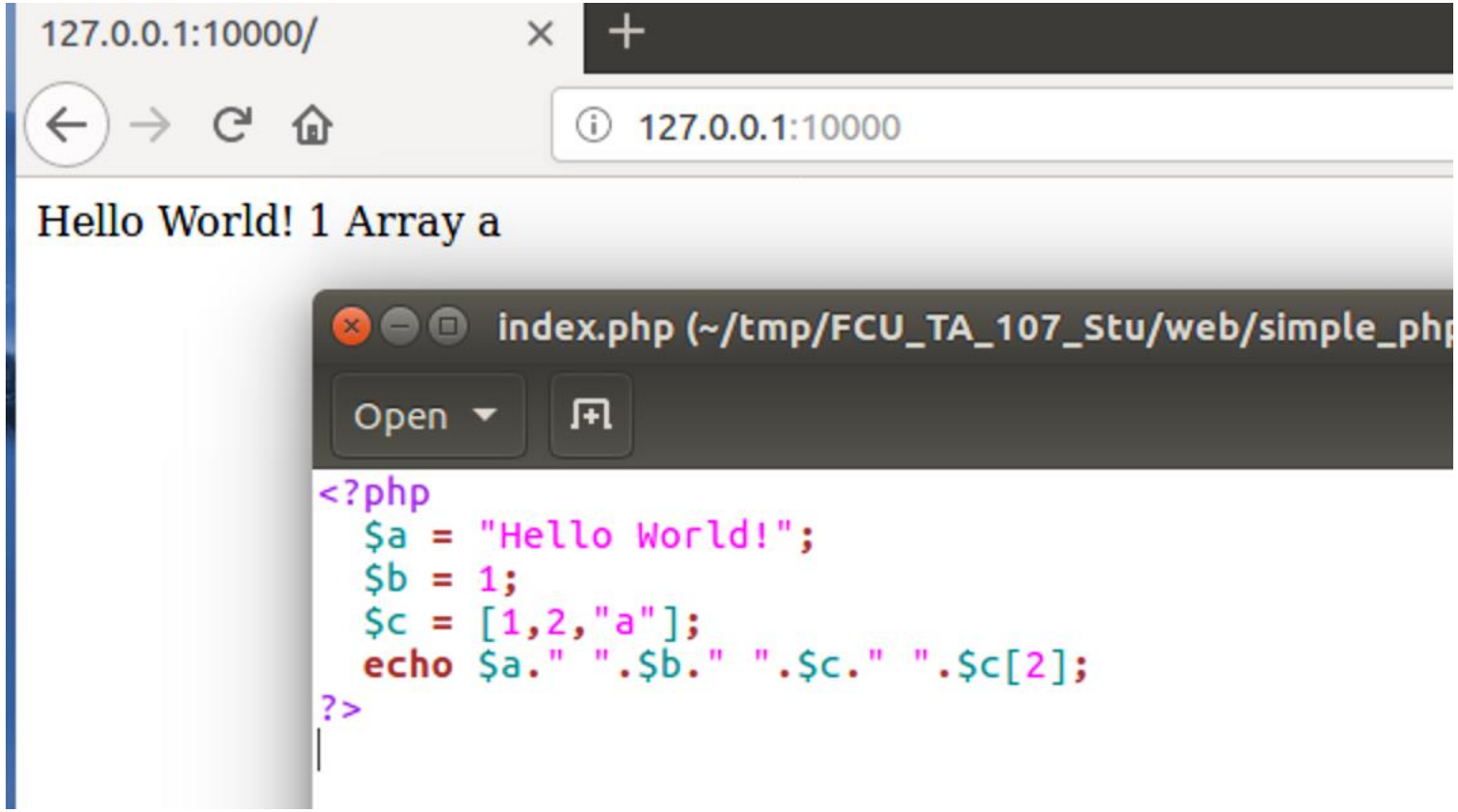
變數

- 用 $來宣告

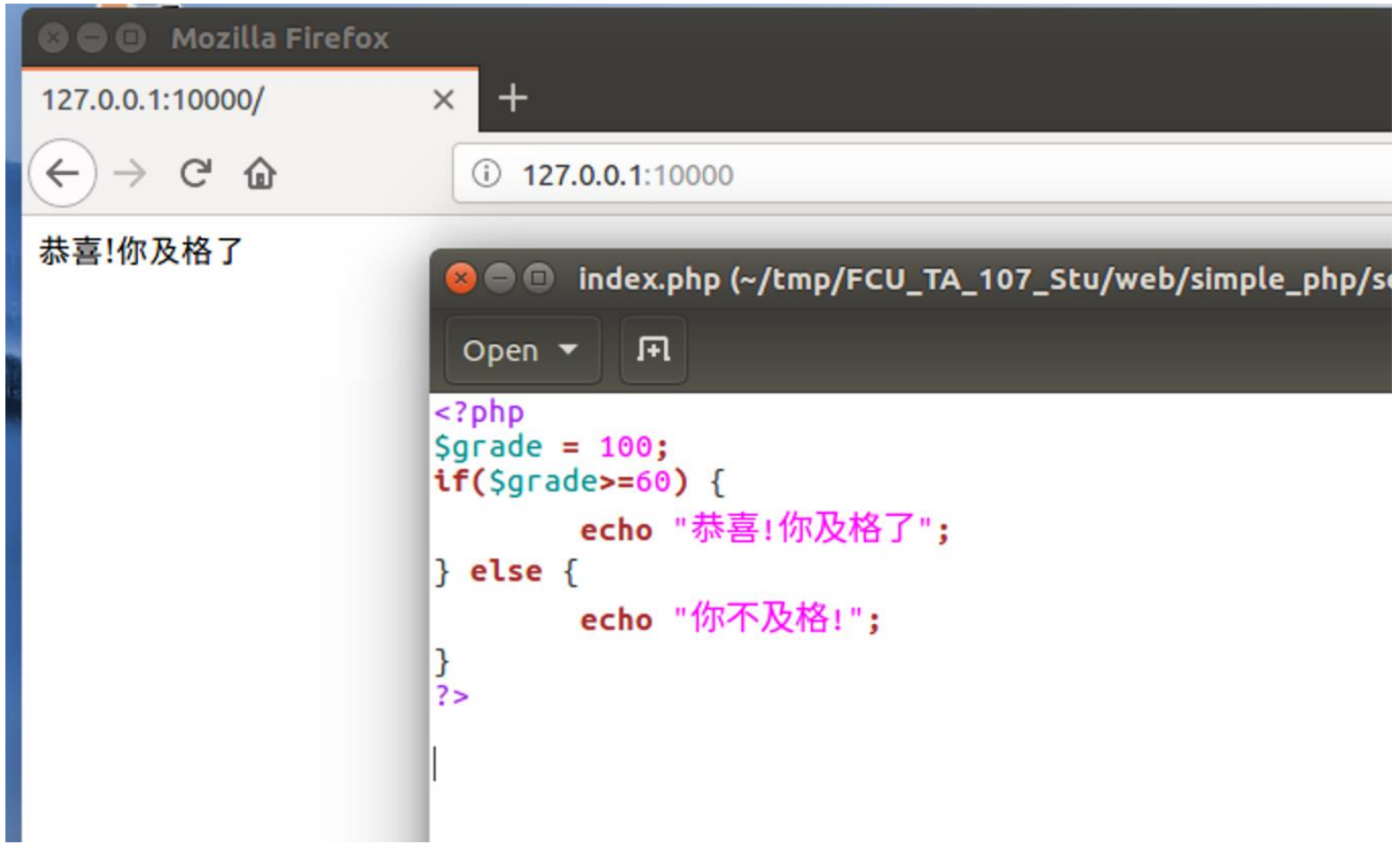

Mozilla Firefox
127.0.0.1:10000/
127.0.0.1:10000
Hello World!
index.php (~/tmp/FCU_TA_107_Stu/web/simple_php/server) - gedit
Open
<?php
$a = "Hello World!";
echo $a;
?>

變數

- 不用型態 字串可以用「.」來做相加



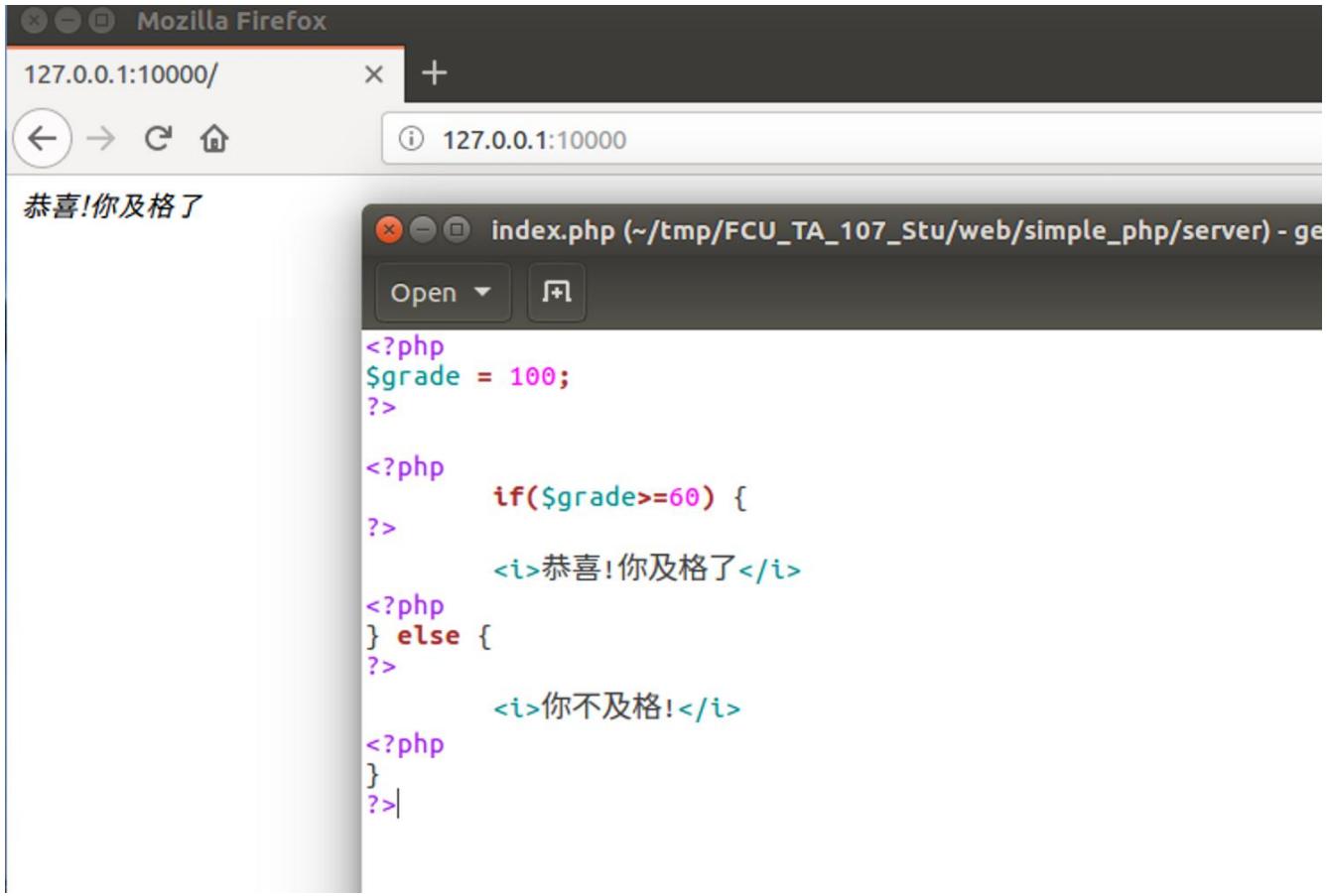
If else


Mozilla Firefox
127.0.0.1:10000/
127.0.0.1:10000
恭喜!你及格了
index.php (~/tmp/FCU_TA_107_Stu/web/simple_php/s
Open
<?php
$grade = 100;
if($grade>=60) {
echo "恭喜!你及格了";
} else {
echo "你不及格!";
}
?>

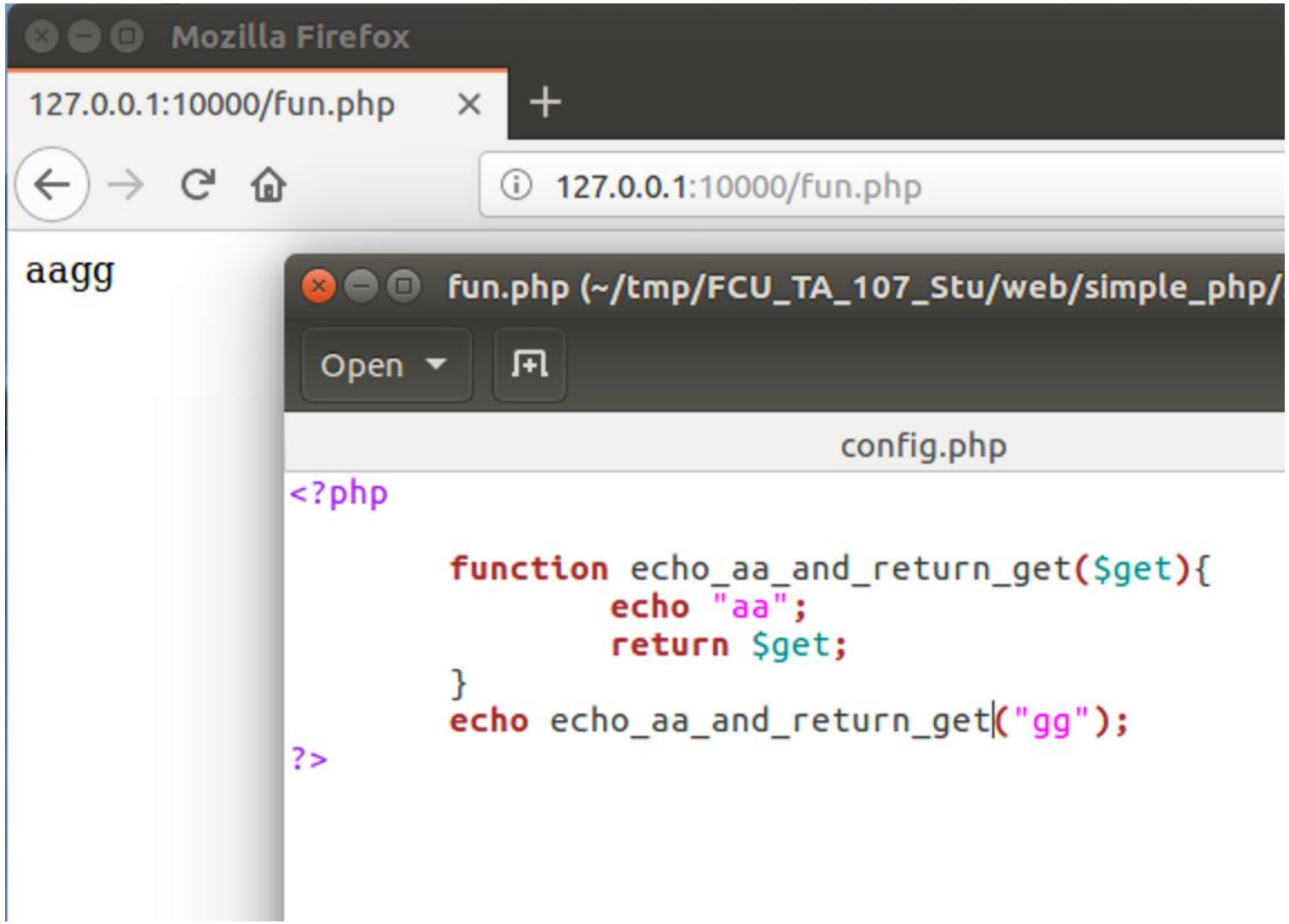
If else

http://127.0.0.1:10000/ifelse.php

- 可以分開 讓他選擇顯示什麼

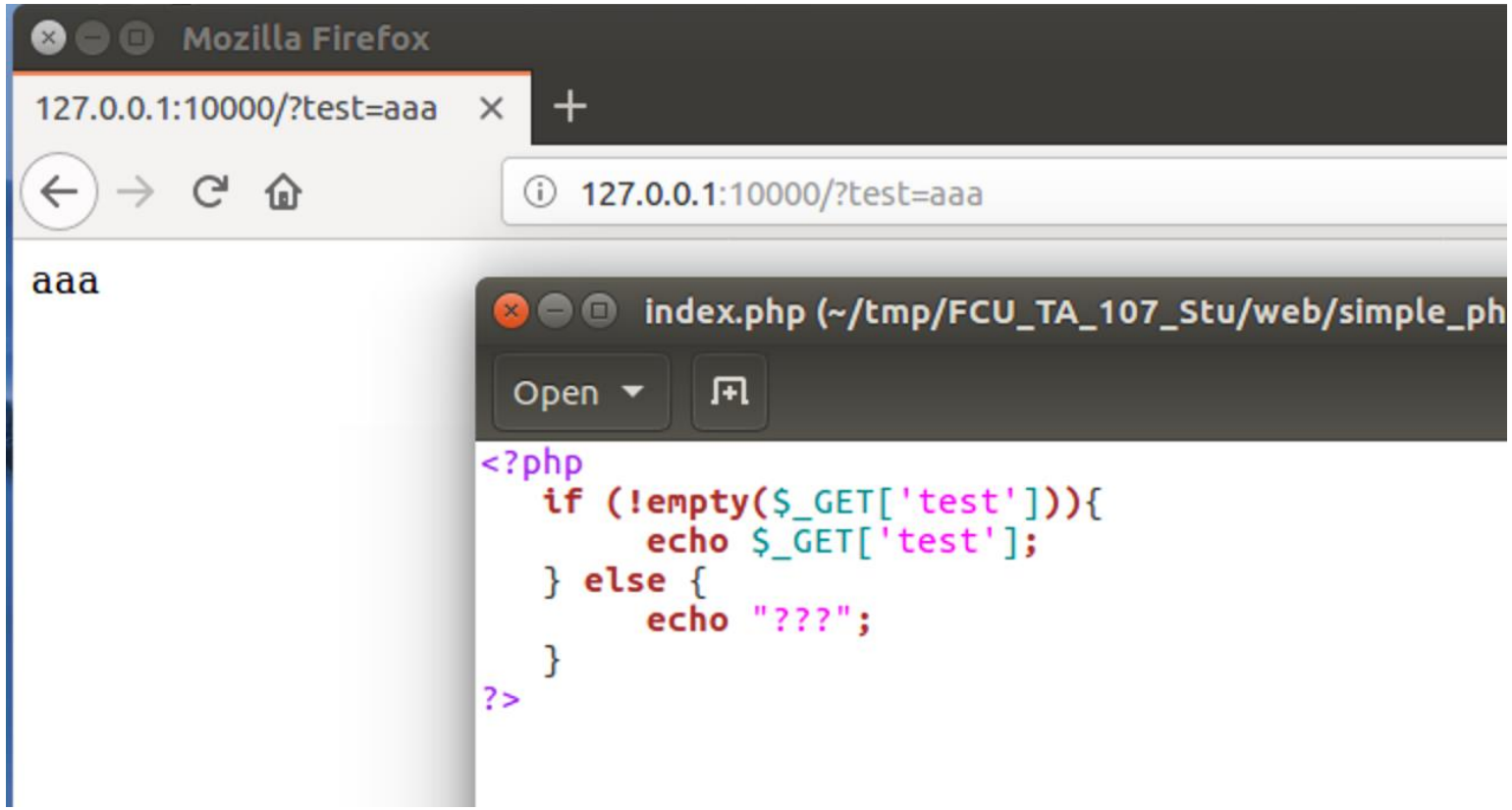


Function


Mozilla Firefox
127.0.0.1:10000/fun.php
127.0.0.1:10000/fun.php
aagg
fun.php (~/tmp/FCU_TA_107_Stu/web/simple_php/
Open
config.php
<?php
function echo_aa_and_return_get($get){
echo "aa";
return $get;
}
echo echo_aa_and_return_get("gg");
?>

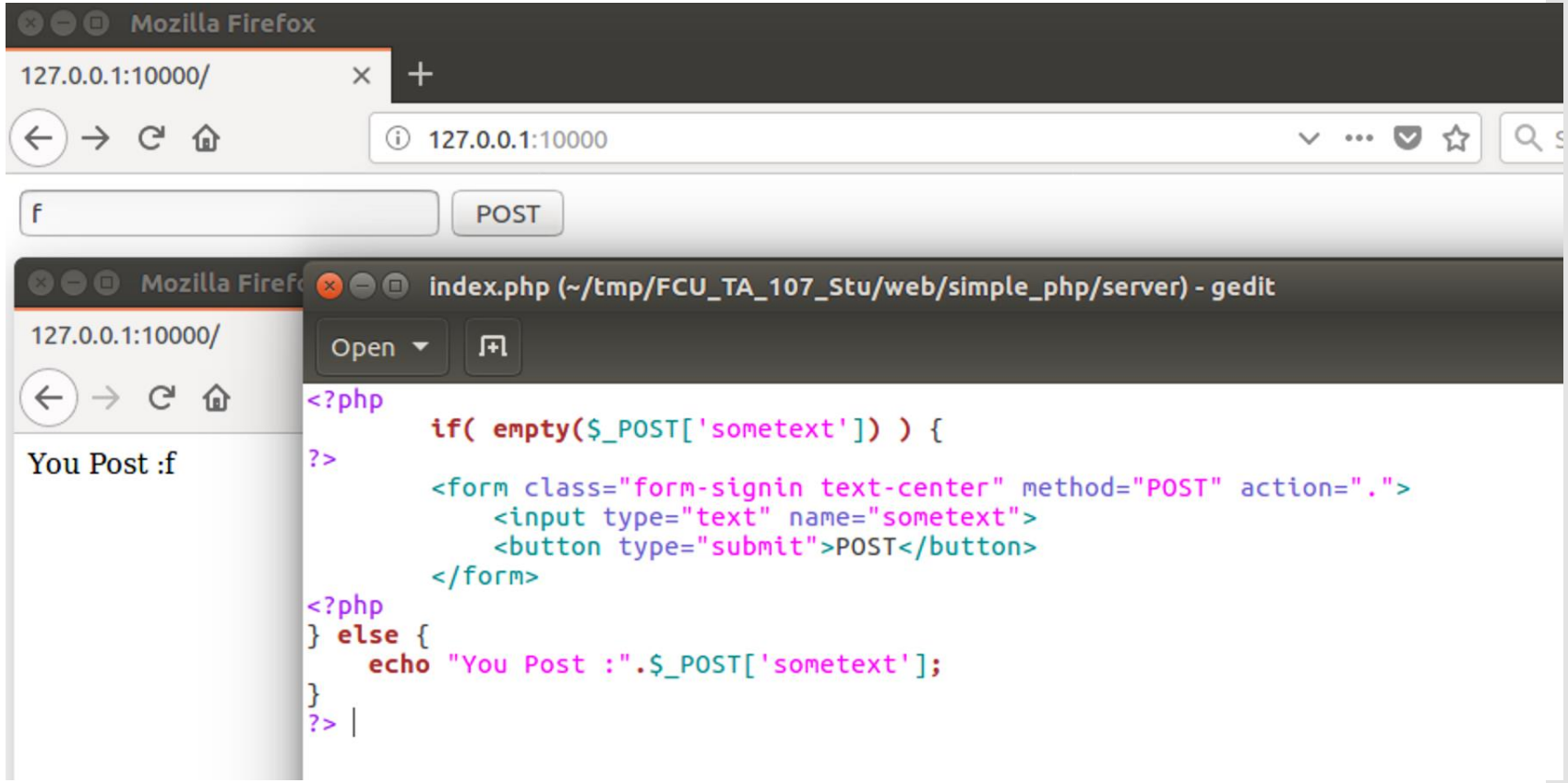
$_GET

- url 傳進來的參數會放在「$_GET」這變數裡
- 通常會用 empty() 或 isset() 檢查他是否有值，避免程式錯誤



$_POST

- POST 傳進來的參數會放在「$_POST」這變數裡
- 通常會用 empty() 或 isset()檢查他是否有值，避免程式錯誤



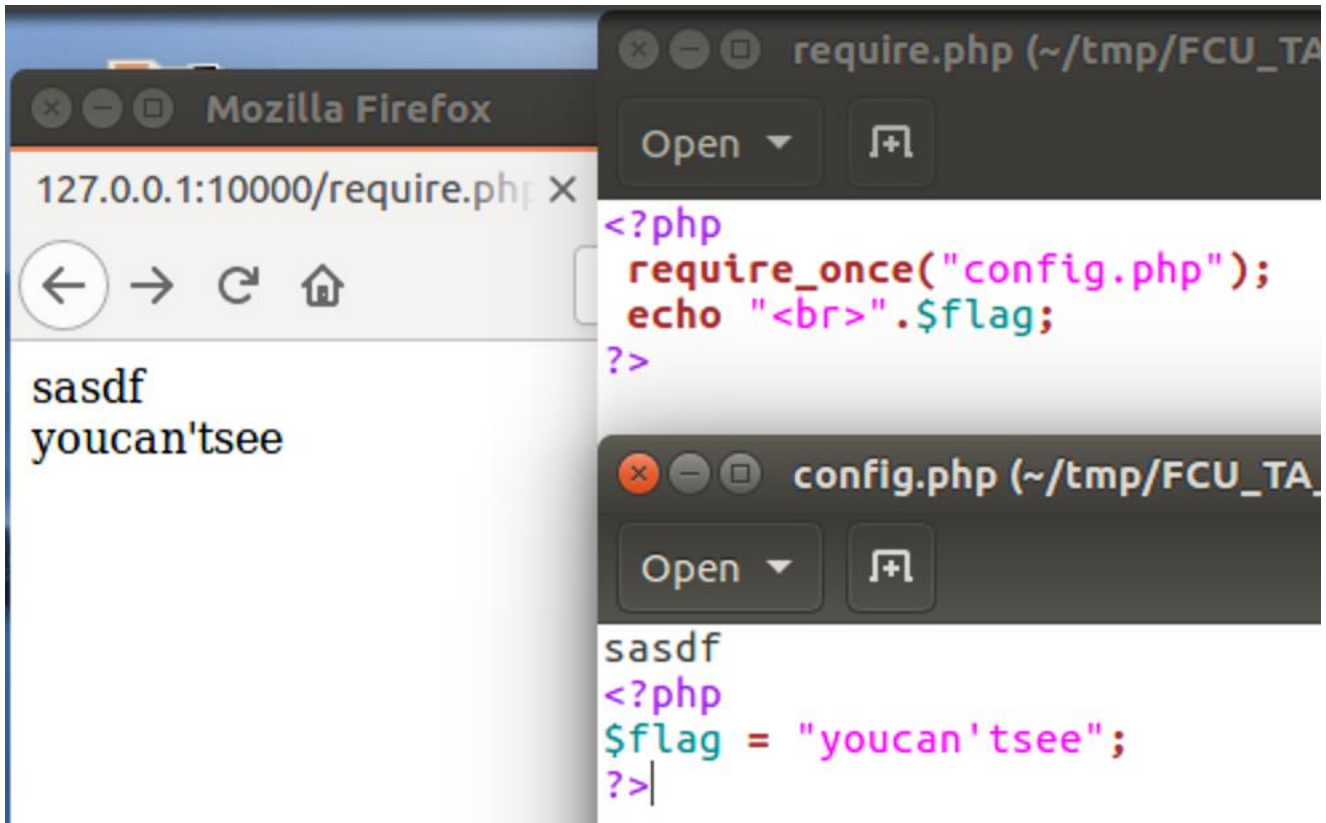
include
include_once
require
require_once

include 和 include_once

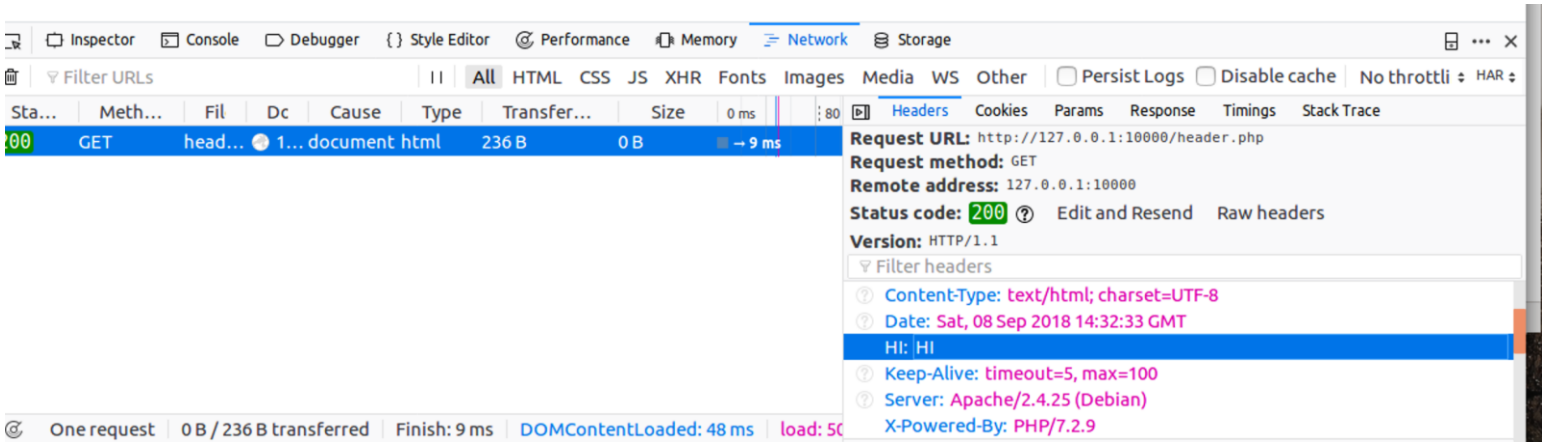
都是用來引入檔案，後者可避免重複引入，故建議用後者。引不到檔案會出現錯誤息，但程式不會停止。

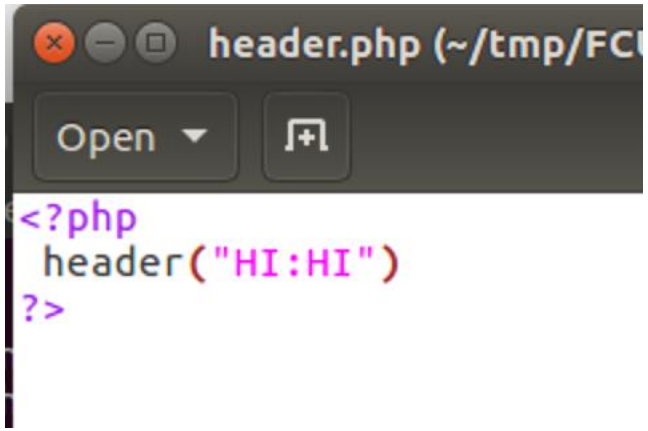
require 和 require_once

都是用來引入檔案，後者可避免重複引入，故建議用後者。引不到檔案會出現錯誤息，而且程式會停止執行。



headers


Inspector
Console
Debugger
Style Editor
Performance
Memory
Network
Storage
Filter URLs
All HTML CSS JS XHR Fonts Images Media WS Other
Persist Logs
Disable cache
No throttli
HAR
Sta... Meth... Fil Dc Cause Type Transfer... Size
GET head... 1... document html 236 B 0 B → 9 ms
Headers Cookies Params Response Timings Stack Trace
Request URL: http://127.0.0.1:10000/header.php
Request method: GET
Remote address: 127.0.0.1:10000
Status code: 200
Edit and Resend
Raw headers
Version: HTTP/1.1
Filter headers
Content-Type: text/html; charset=UTF-8
Date: Sat, 08 Sep 2018 14:32:33 GMT
HI: HI
Keep-Alive: timeout=5, max=100
Server: Apache/2.4.25 (Debian)
X-Powered-By: PHP/7.2.9
One request
0 B / 236 B transferred
Finish: 9 ms
DOMContentLoaded: 48 ms

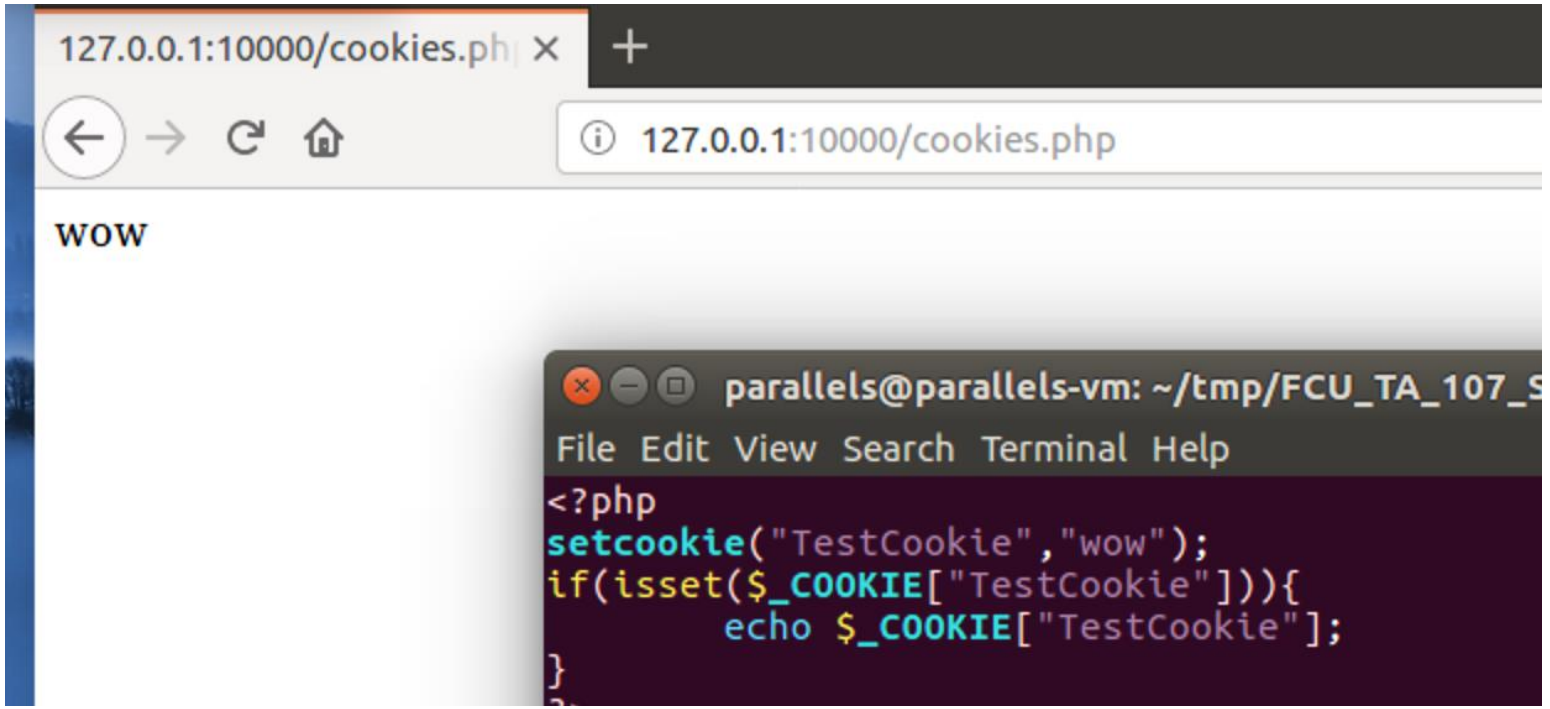

header.php (~/tmp/FC
Open
<?php
header("HI:HI")
?>

redirect

```php
<?php
        header("Location: http://abc.com/login"); //301頁面轉跳
?>
```

cookies

- 第一次因為cookies還沒設定 所以不會回任何訊息
- 第二次之後cookies就一直在

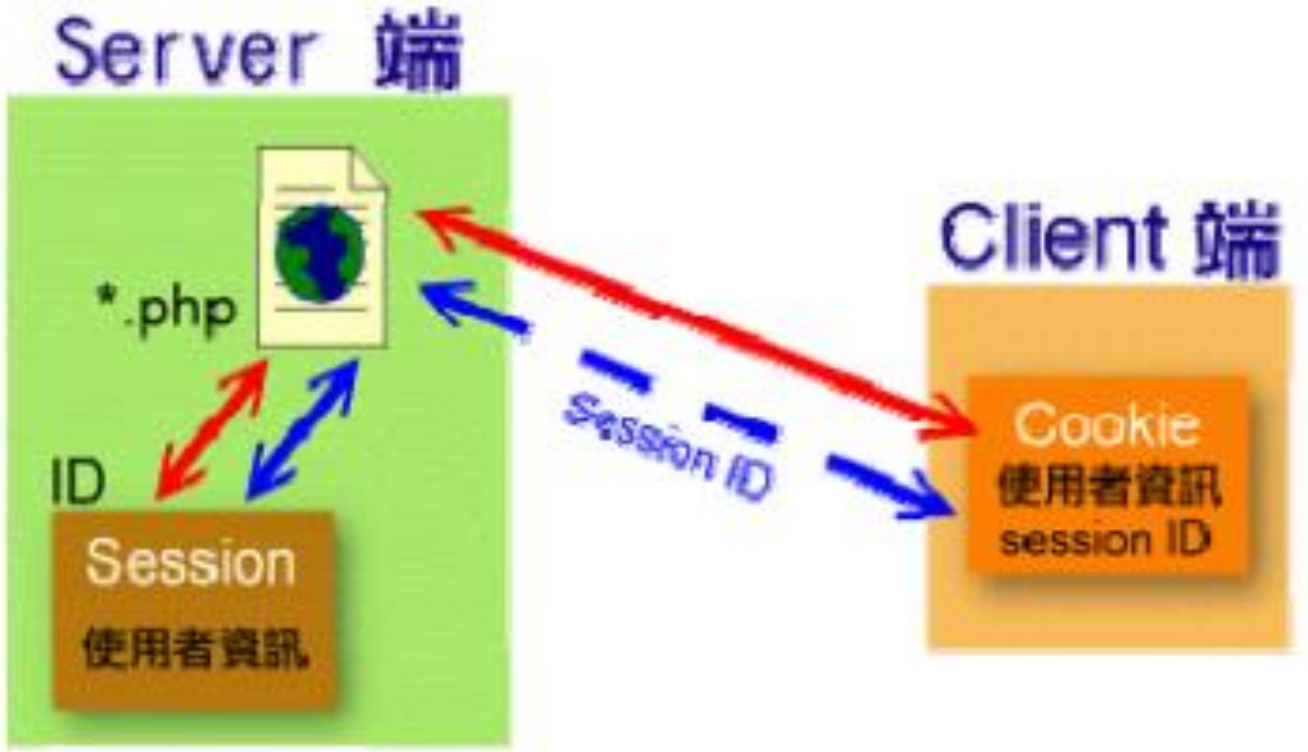


程式碼解析

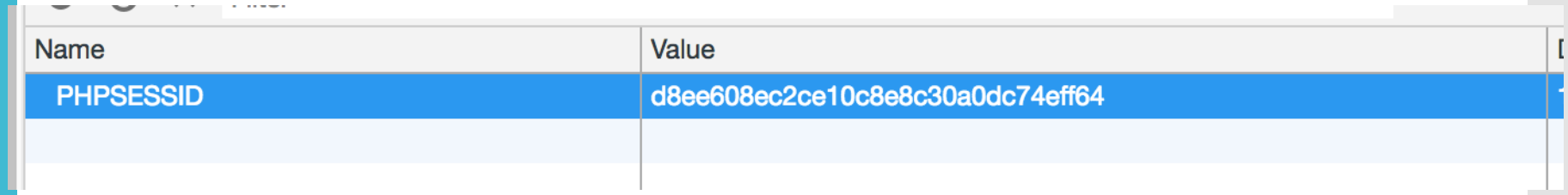
```php
<?php
$flag = "CTF{Dont_use_cookies_as_a_flag!!}";
$text = "U are not admin , no flag!!";
if (isset($_COOKIE['user'])) {
    $cookie = unserialize($_COOKIE['user']);
    if ($cookie['admin'] === true) {
        $text = $flag;
    }
} else {
    setcookie('user', serialize(['user_id' => 23333, 'admin' => false]), time()+3600);
}
?>
<!DOCTYPE html>
```

Php serialize與unserialize
在以前常用於把資料打包放在client上
但如果把不該給使用者控制的資料放在上面
就會導致權限管控問題

所以資訊要使用「session」
客戶端cookies只會放一個「session ID」
其他資訊全部存在伺服器資料庫



Session ID 會過期
並且是隨機產生的
過期就需要再次登入

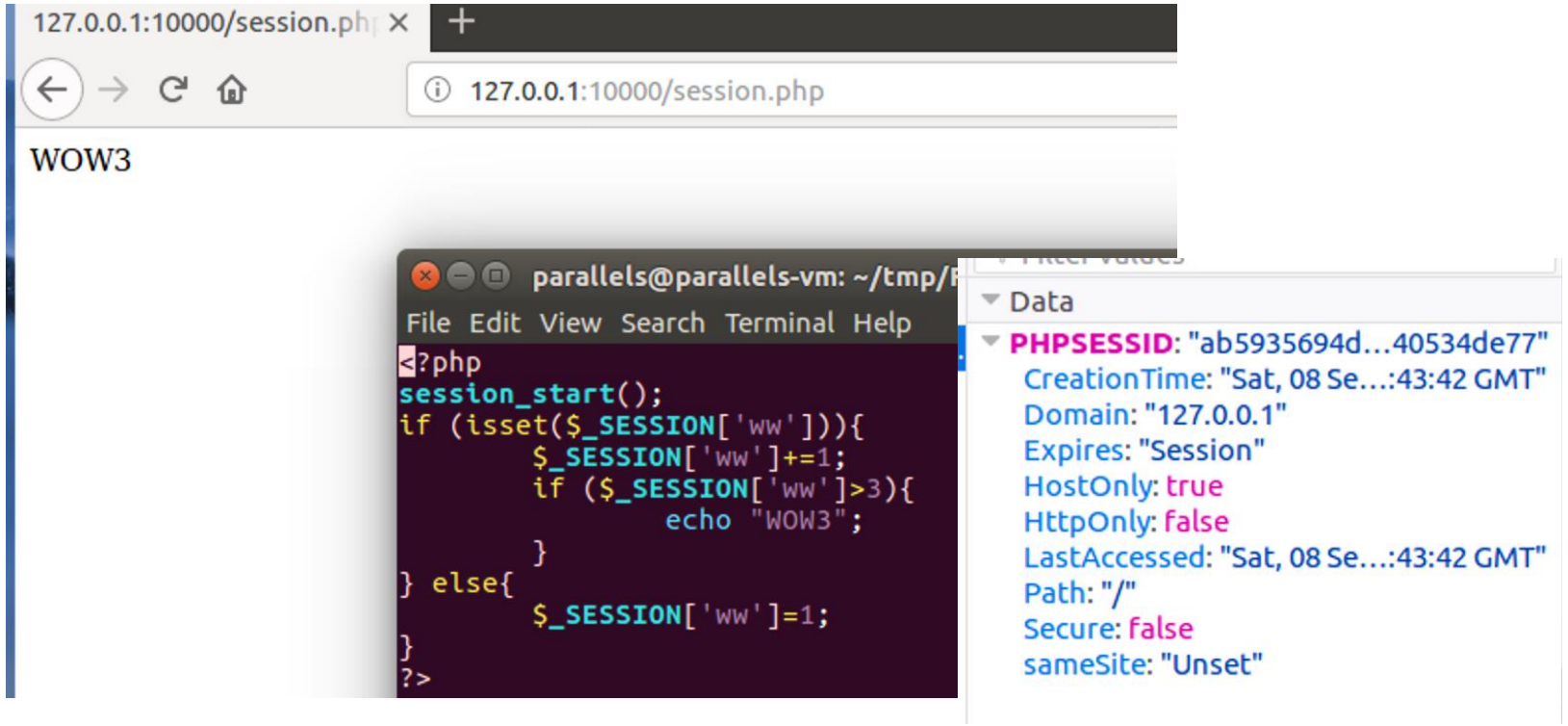


Name	Value
PHPSESSID	d8ee608ec2ce10c8e8c30a0dc74eff64

但同樣的 Session ID被知道
一樣能盜用他人帳號

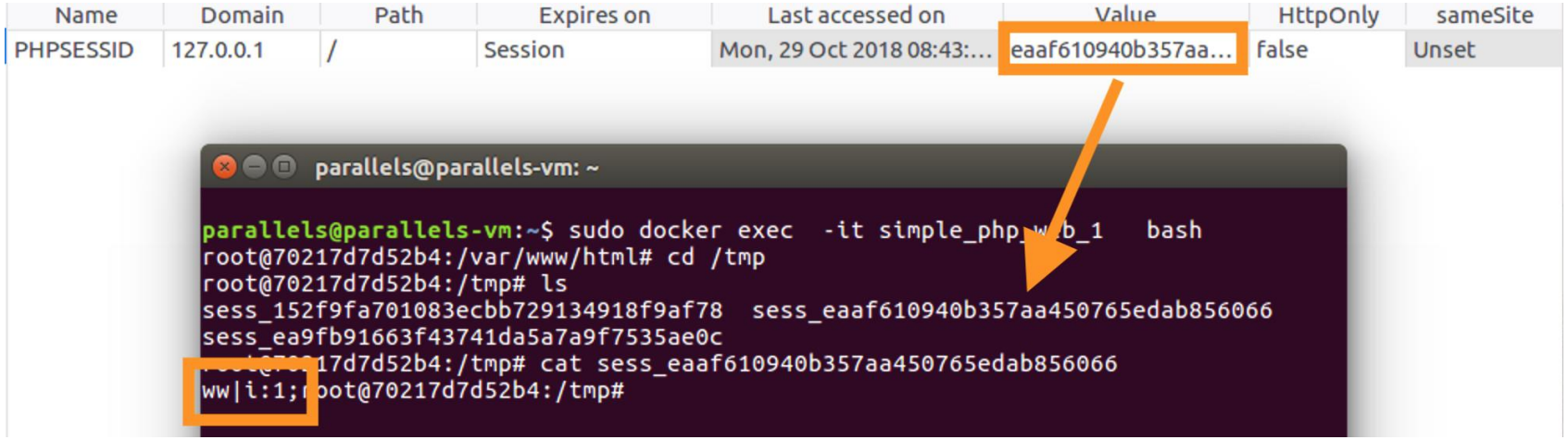
session

- 你無法直接設定ww值 因為cookies永遠只會有sessionid
- 而沒有ww，所以你一定要開三次才看得到WOW3



session

- 至於session資料存在伺服器的哪裡呢？
- 在/tmp裡面
- 輸入 sudo docker exec –it simple_php_web_1 bash 進入docker
- 然後cd /tmp 查看
- 會看到很多session檔案
- cat 檔案



SQL 連線

```php
<?php
$servername = "localhost";
$username = "username";
$password = "password";
$dbname = "myDB";

// Create connection
$conn = new mysqli($servername, $username, $password, $dbname);
// Check connection
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {
        echo "id: " . $row["id"]. " - Name: " . $row["firstname"]. " " . $row["lastname"]. "
<br>";
    }
} else {
    echo "0 results";
}
$conn->close();
?>
```

SQL 連線

範例表格

UID	NAME	EMAIL	PASSWORD
1	RicharLin	Richar@richarlin.tw	1234567

使用 fetch_array() ：

```
array(
    [0] => '1'
    [UID] => '1'
    [1] => 'RicharLin'
    [Name] => 'RicharLin'
    [2] => 'Richar@richarlin.tw'
    [EMAIL] => 'Richar@richarlin.tw'
    [3] => '1234567'
    [PASSWORD] => '1234567'
)
```

SQL 連線

範例表格

UID	NAME	EMAIL	PASSWORD
1	RicharLin	Richar@richarlin.tw	1234567

使用 fetch_array() ：

```
array(
    [0] => '1'
    [UID] => '1'
    [1] => 'RicharLin'
    [Name] => 'RicharLin'
    [2] => 'Richar@richarlin.tw'
    [EMAIL] => 'Richar@richarlin.tw'
    [3] => '1234567'
    [PASSWORD] => '1234567'
)
```

SQL 連線

使用 fetch_assoc() ：

```
array(
    [UID] => '1'
    [Name] => 'RicharLin'
    [EMAIL] => 'Richar@richarlin.tw'
    [PASSWORD] => '1234567'
)
```

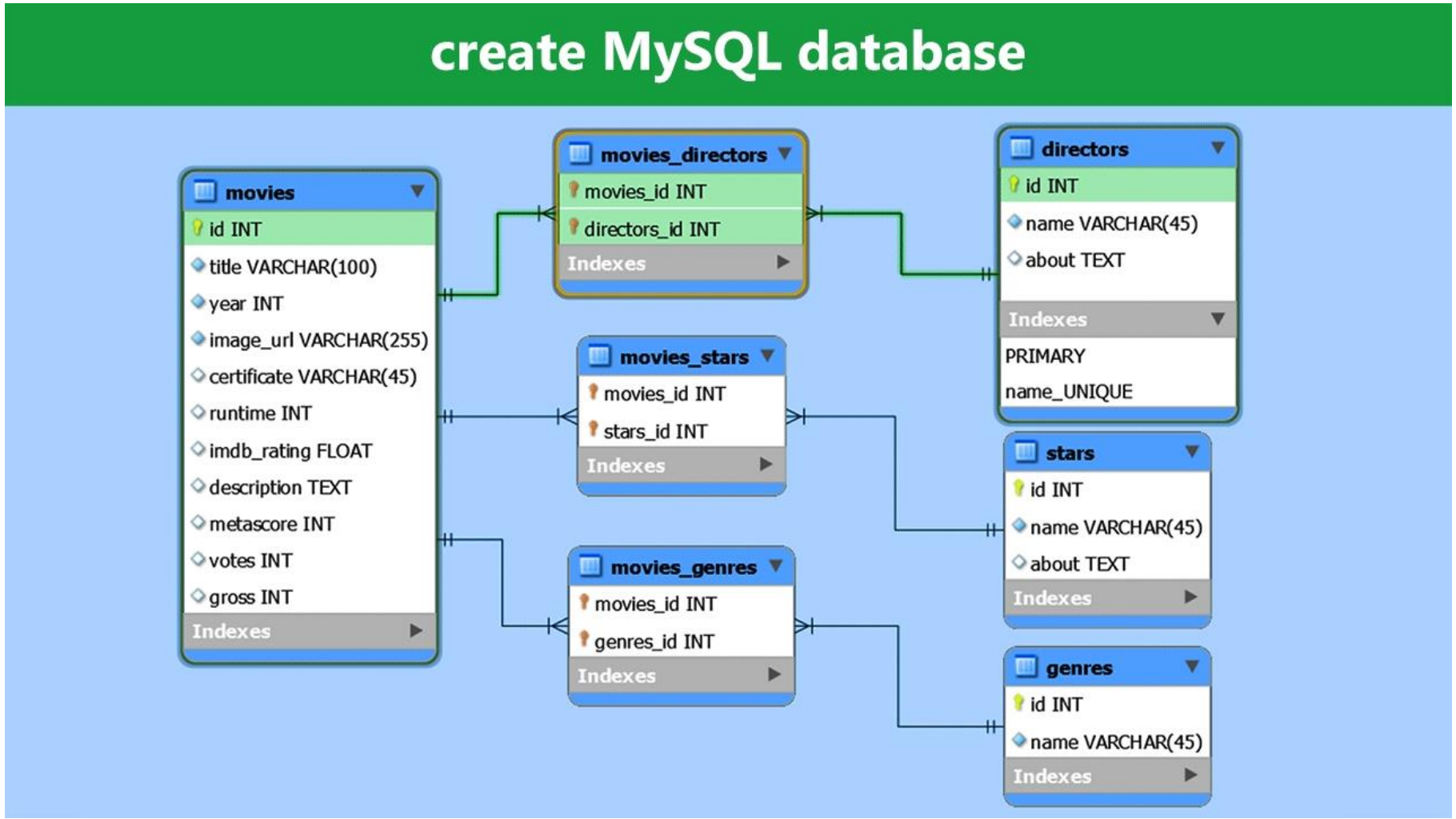
使用 fetch_row() ：

```
array(
    [0] => '1'
    [1] => 'RicharLin'
    [2] => 'Richar@richarlin.tw'
    [3] => '1234567'
)
```

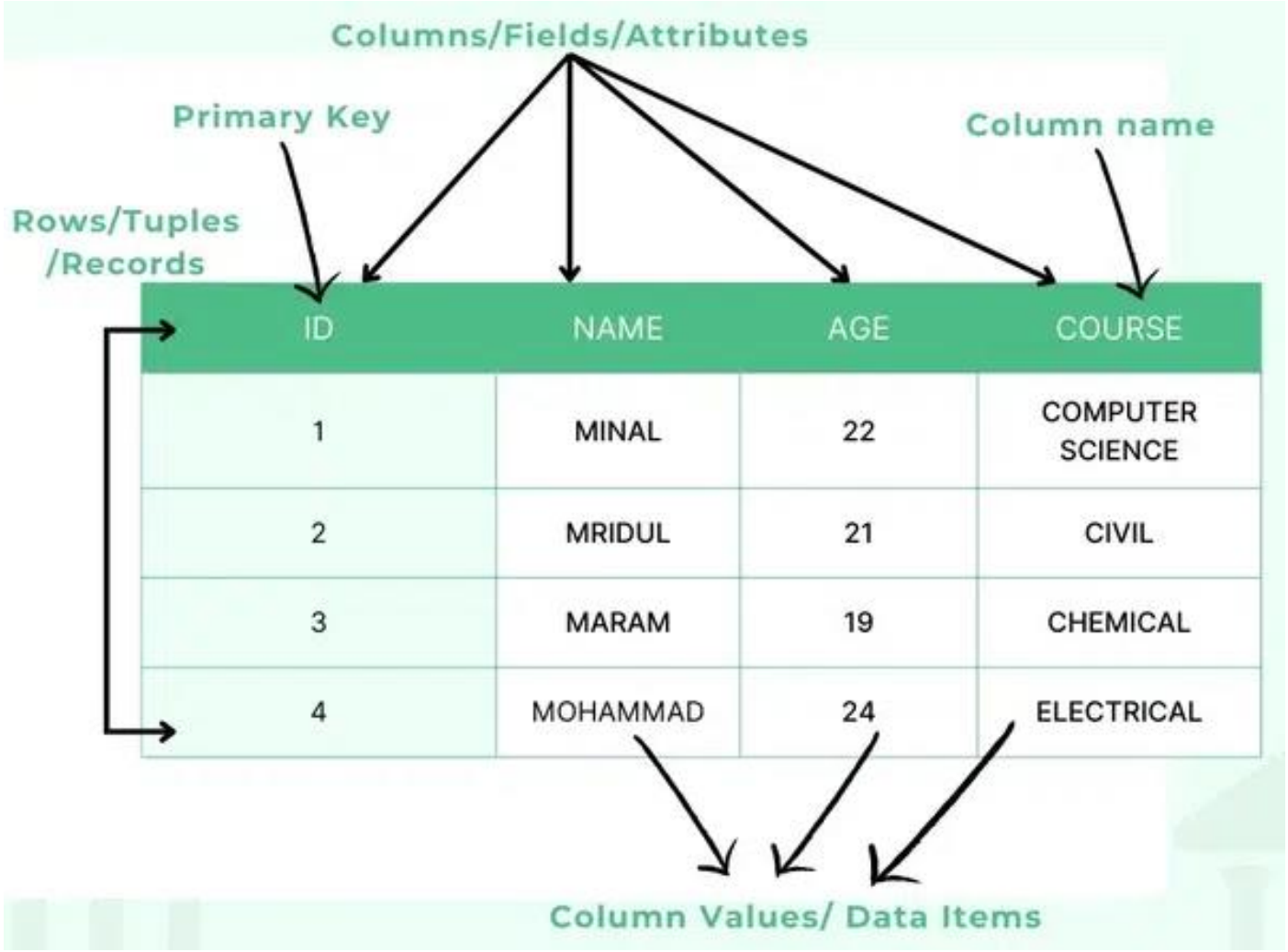
SQL

https://sqliteonline.com/

什麼是資料庫(database)



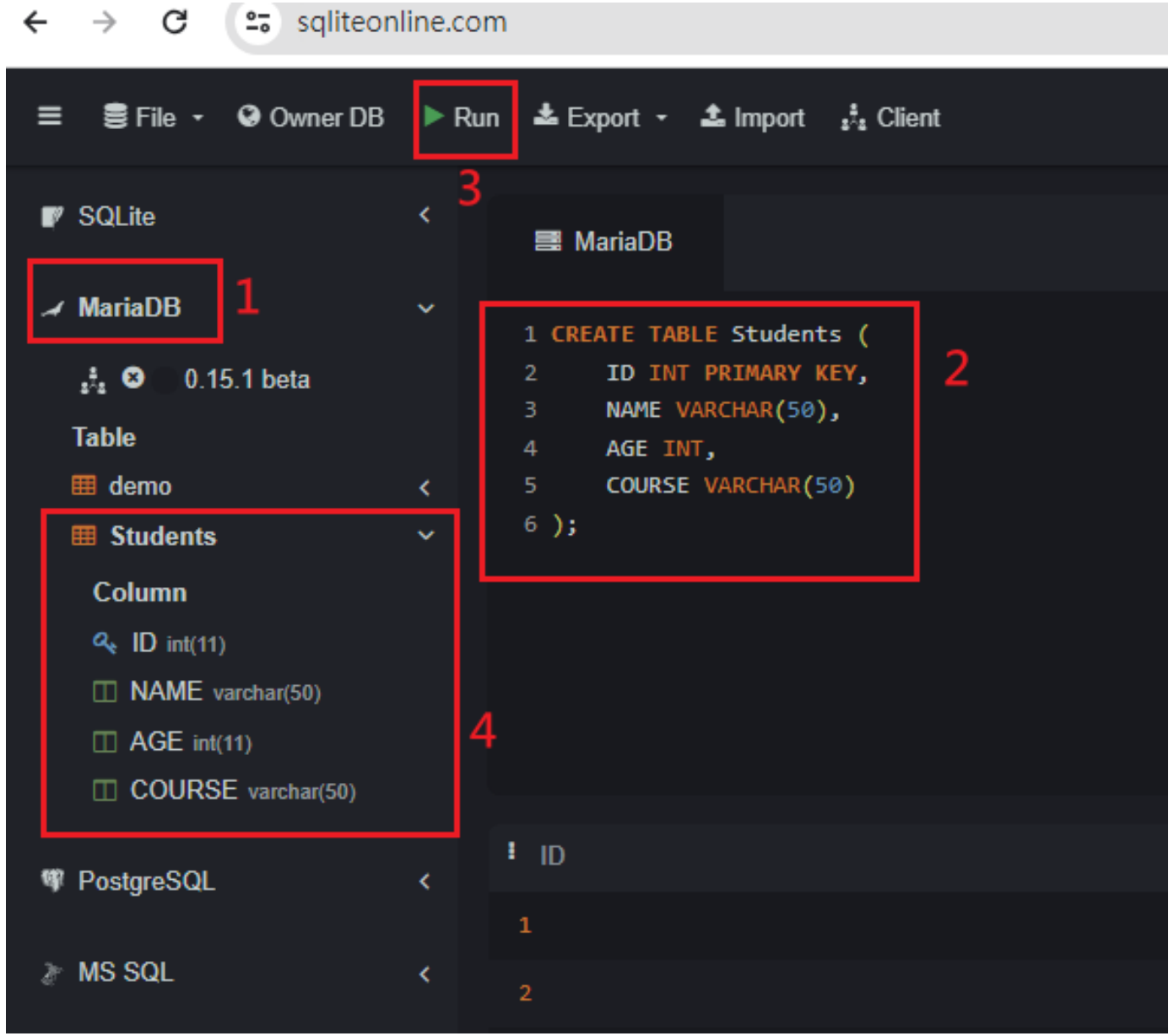
資料庫中表(table)的元件



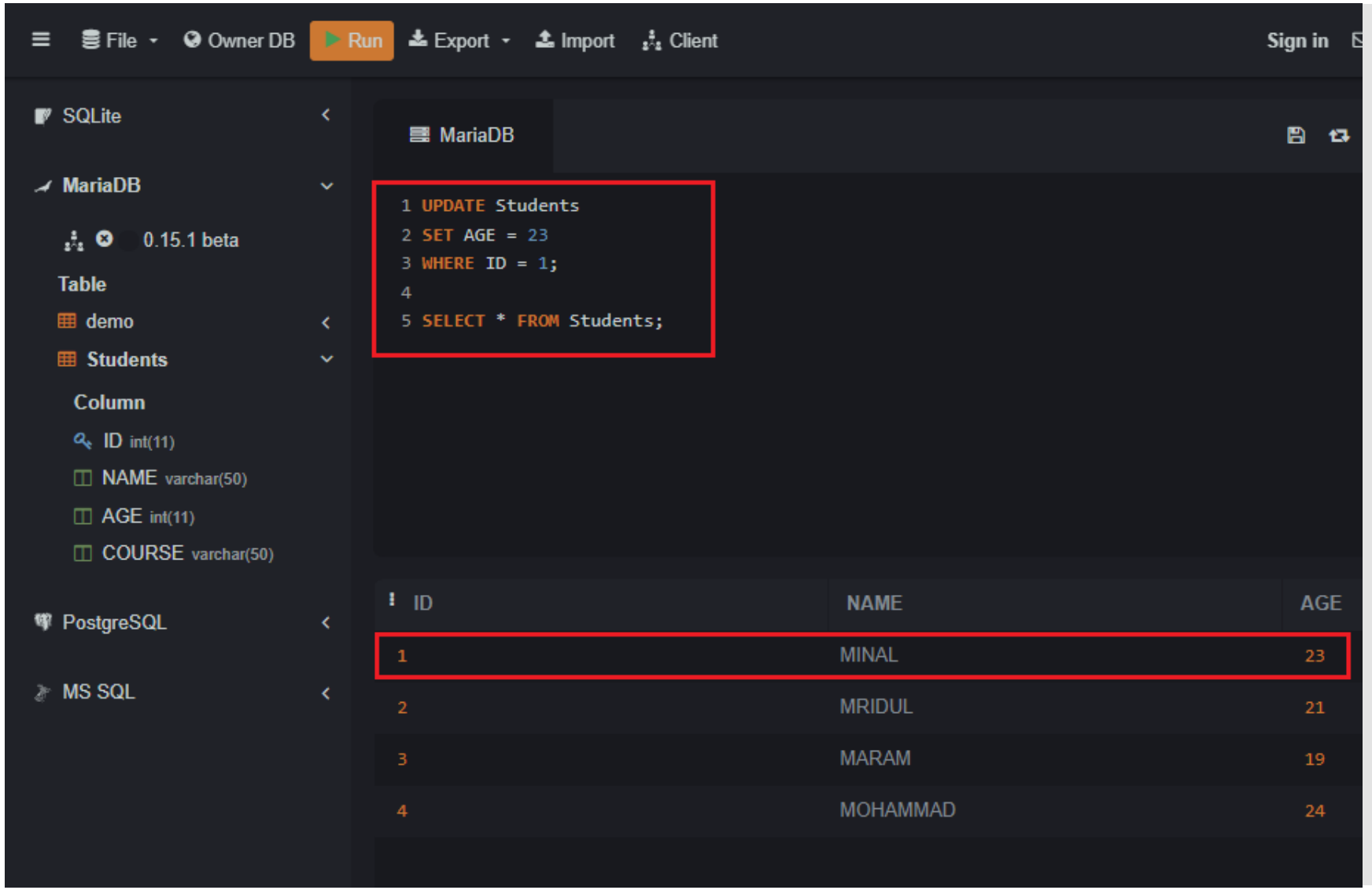
一些常見的資料庫


MariaDB
Foundation
PostgreSQL
MySQL
mongoDB
IBM
DB2
Microsoft
SQL Server
ORACLE

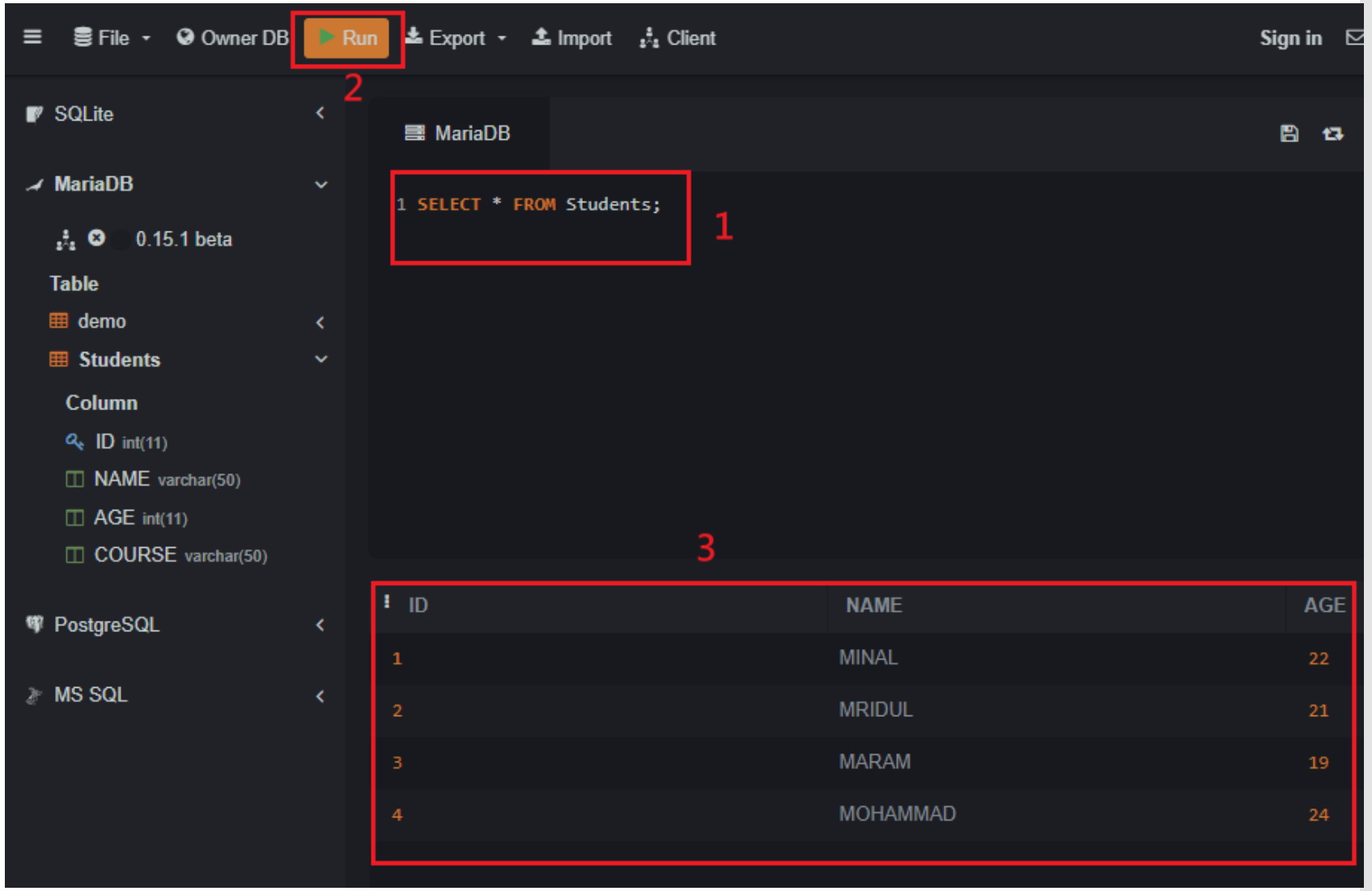
Sql建立資料表語法


sqliteonline.com
File
Owner DB
Run
Export
Import
Client
3
SQLite
MariaDB
1
0.15.1 beta
Table
demo
Students
Column
ID int(11)
NAME varchar(50)
AGE int(11)
COURSE varchar(50)
4
PostgreSQL
MS SQL
MariaDB
1 CREATE TABLE Students (
2 ID INT PRIMARY KEY,
3 NAME VARCHAR(50),
4 AGE INT,
5 COURSE VARCHAR(50)
6);
2
ID
1
2

新增資料


File
Owner DB
Run
Export
Import
Client
Sign in
SQLite
MariaDB
0.15.1 beta
Table
demo
Students
Column
ID int(11)
NAME varchar(50)
AGE int(11)
COURSE varchar(50)
PostgreSQL
MS SQL
MariaDB
1 UPDATE Students
2 SET AGE = 23
3 WHERE ID = 1;
4
5 SELECT * FROM Students;
ID
NAME
AGE
1
MINAL
23
2
MRIDUL
21
3
MARAM
19
4
MOHAMMAD
24

查詢


File
Owner DB
Run
Export
Import
Client
Sign in
2
SQLite
MariaDB
0.15.1 beta
Table
demo
Students
Column
ID int(11)
NAME varchar(50)
AGE int(11)
COURSE varchar(50)
PostgreSQL
MS SQL
MariaDB
1 SELECT * FROM Students;
1
3
ID
NAME
AGE
1
MINAL
22
2
MRIDUL
21
3
MARAM
19
4
MOHAMMAD
24

查詢語法

```
SELECT [* | DISTINCT | Top n] <欄位串列>
FROM (資料表名稱{<別名>} | JOIN資料表名稱)
[WHERE <條件式>]
[GROUP BY <群組欄位> ]
[HAVING <群組條件>]
[ORDER BY <欄位> [ASC | DESC]]
```

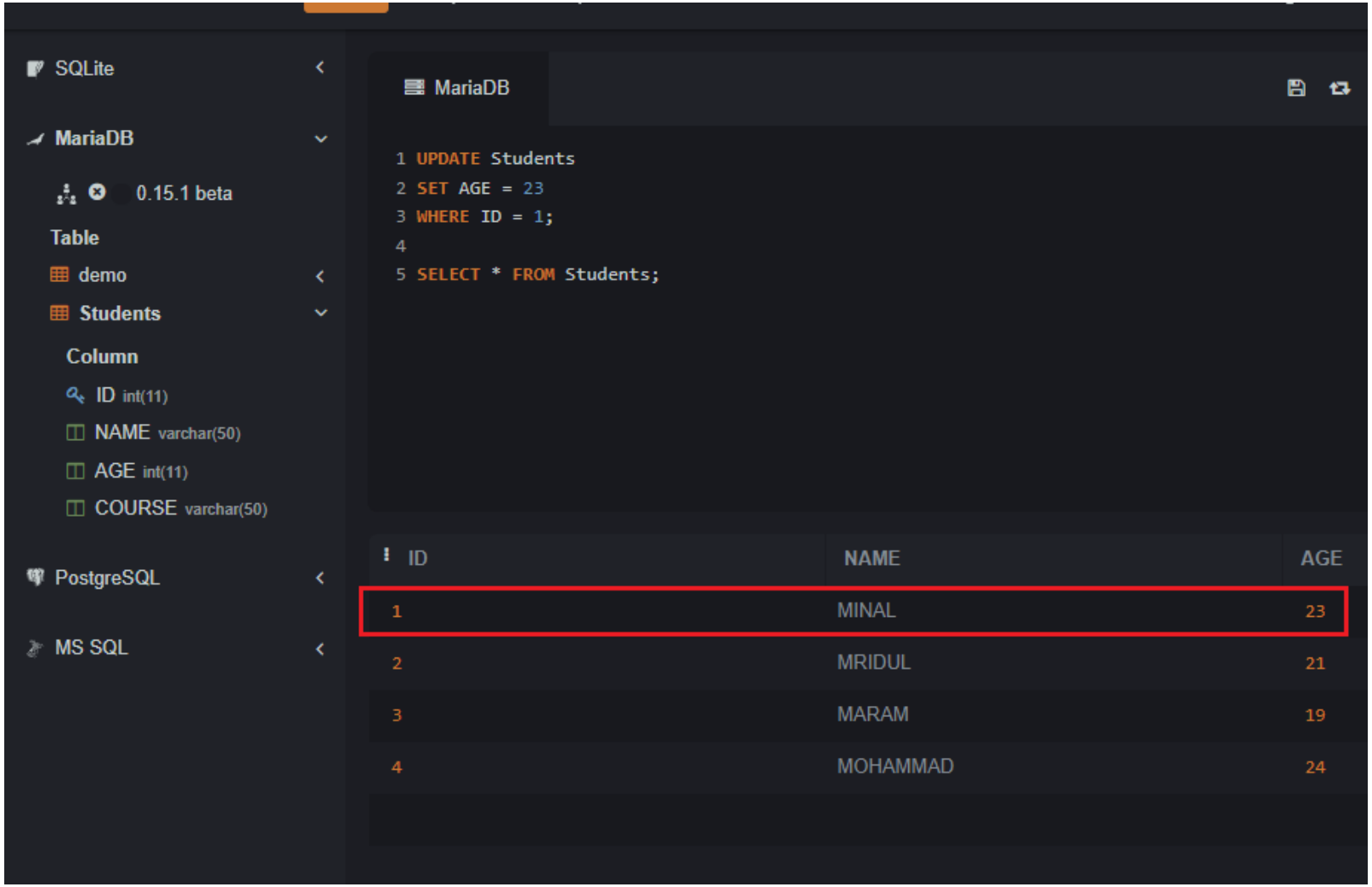
查詢的範例

- 查詢所有不同的專業（major）
  - `SELECT DISTINCT major FROM students;`
- 查詢前 10 位學生的資料
  - `SELECT TOP 10 * FROM students;`
- 查詢專業為 "Mathematics" 的學生，並且只顯示前 5 名成績最好的學生
  - `SELECT TOP 5 * FROM students WHERE major = 'Mathematics' ORDER BY grade DESC;`
- 查詢所有學生資料，按成績（grade）降序排列，如果成績相同則按名字（name）升序排列
  - `SELECT * FROM students ORDER BY grade DESC, name ASC;`

where的模糊查詢

- Like :模糊相似條件
  - Where 系所LIKE '資管%’
- IN :集合條件
  - Where課程代號IN('C001','C002’)
- Between……And 範圍條件
  - Where成績Between 60 And 80

更新單一欄位


SQLite
MariaDB
0.15.1 beta
Table
demo
Students
Column
ID int(11)
NAME varchar(50)
AGE int(11)
COURSE varchar(50)
PostgreSQL
MS SQL
MariaDB
1 UPDATE Students
2 SET AGE = 23
3 WHERE ID = 1;
4
5 SELECT * FROM Students;
ID
NAME
AGE
1
MINAL
23
2
MRIDUL
21
3
MARAM
19
4
MOHAMMAD
24

更新多個欄位

```sql
UPDATE Students
SET AGE = 22, COURSE = 'MECHANICAL'
WHERE ID = 2;


SELECT * FROM Students;
```

ID	NAME	AGE	COURSE
1	MINAL	23	COMPUTER SCIENCE
2	MRIDUL	22	MECHANICAL
3	MARAM	19	CHEMICAL
4	MOHAMMAD	24	ELECTRICAL

刪除單一記錄

MariaDB

```
1 DELETE FROM Students
2 WHERE ID = 3;
3
4
5 SELECT * FROM Students;
```

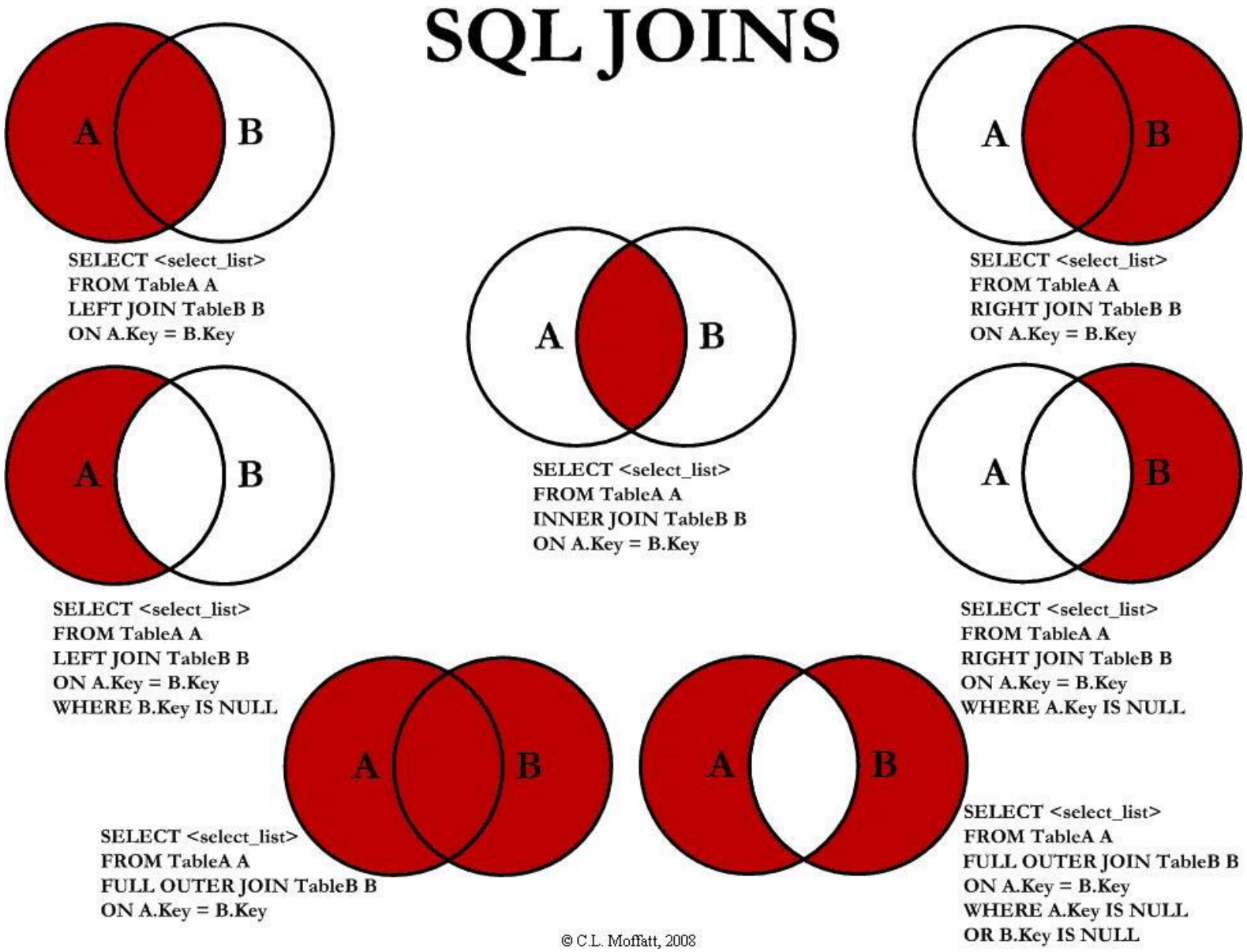
ID	NAME	AGE	COURSE
1	MINAL	23	COMPUTER SCIENCE
2	MRIDUL	22	MECHANICAL
4	MOHAMMAD	24	ELECTRICAL

全删

```
1 DELETE FROM Students;
2
```

join

SELECT column1, column2, ...
FROM table1
JOIN table2 ON condition;



操作Database基本指令

命令	說明
show databases;	顯示所有資料庫
create database <name>;	創建資料庫，並指定其名稱 (注意：名稱不能空格)
drop database <name>;	刪除指定名稱的資料庫
use <database name>;	指定使用哪個資料庫(切換資料庫)
select database();	查詢當前使用的資料庫是哪一個

注意:
刪掉 資料庫裡的 Students 這個表**DROP TABLE `FCU`.`Students`;**

是 ` 而不是 ''這會被當成字串''這才會被當成資料庫或表或欄位的名稱`

註解

```
-- 單行註解 注意 -- 後要空白
select 1 AS A#單行註解;
/*註解*/
/*
多行註解
*/
```

雖然上面 # 那個沒變色但是他的確是註解

SQL Injection

從開始到放棄(喂

admin' or 1=1#'

```
SELECT * from Users
  where user = 'admin' or 1=1#'' and password = md5('meow');
```

admin'/**/or/**/1/**/=/**/1;
#

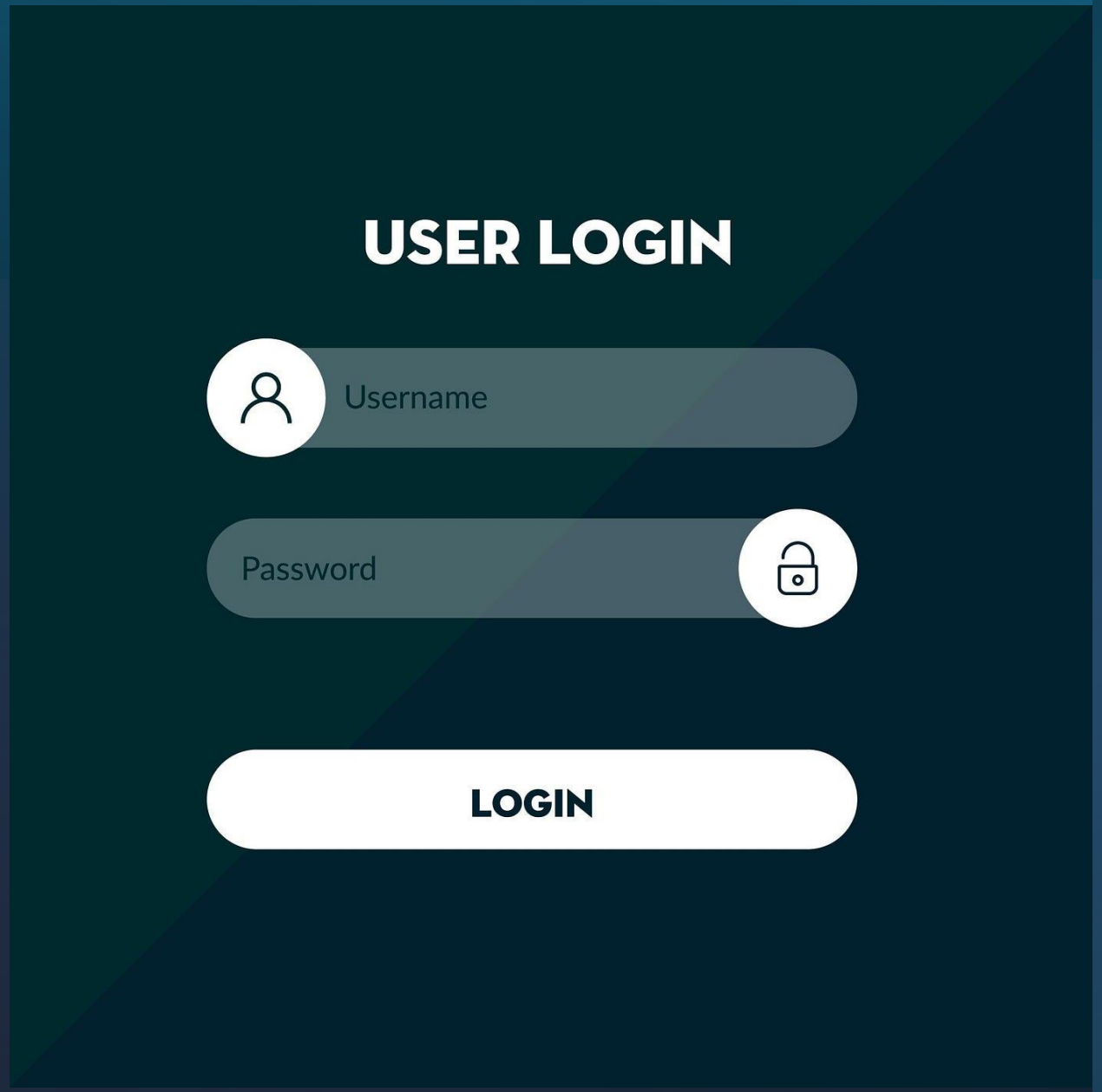
一些同義詞語法

- 空白
  - /* 註解 */
- true
  - 87=87
- false
  - 1=0
- and
  - &&
- or
  - ||

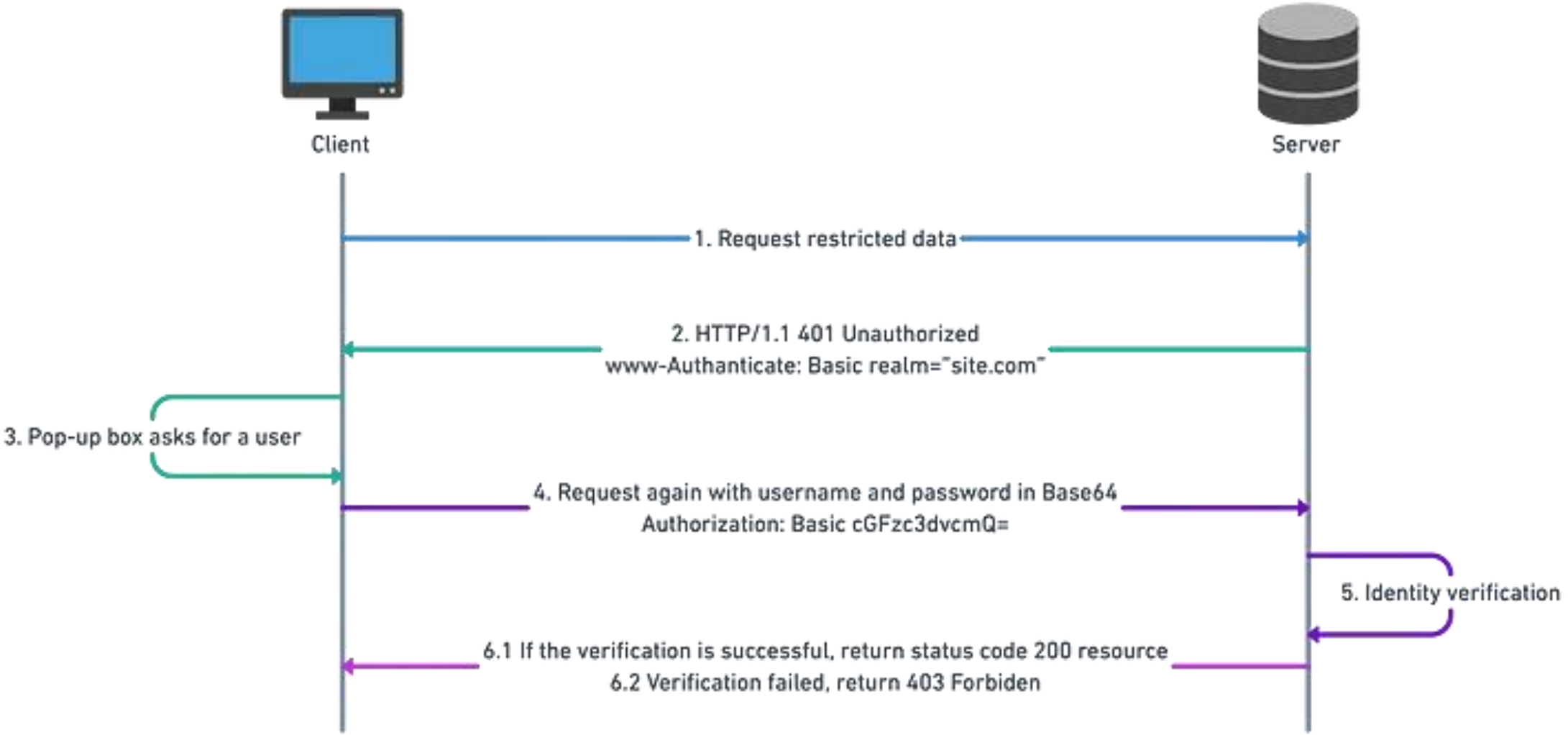
全部的帳密出來啦


顯示第 0 - 4 列 (總計 5 筆, 查詢花費 0.0004 秒。)
SELECT * from Users where user = 'admin' or 1=1#'' and password = md5('meow');
全部顯示
資料列數:
25
搜尋資料列:
搜尋此資料表
+ 選項
id
user
password
編輯 複製 刪除 1 admin 0ef189449c087a9b1577db34a10b4fb010077ae8
編輯 複製 刪除 2 root ceaf14fdcd7a80564ecf19144c25a4bf6db57258
編輯 複製 刪除 3 meow a4a29b5575fdb0a1d7bbed5c93861849c2417db0
編輯 複製 刪除 4 hax0r 6a4c0d00ffe971dd1ccea54e5c395f4271abd164
編輯 複製 刪除 5 corgi 08ee497fad7c7ffd34678105c4827d8304bc52e4

登入驗證

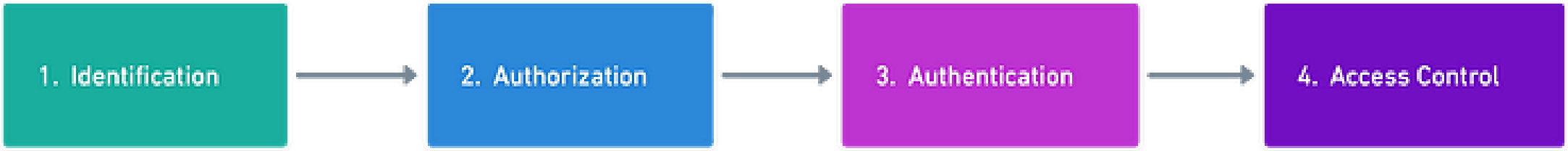

USER LOGIN
Username
Password
LOGIN

在登入頁面背後 **HTTP**做這樣的事



網站如何登入

- 將過程簡單劃分4個階段



登入過程- identification身份證明

- 身份證
- 使用者名和密碼
- 用戶的手機
- 用戶的電子郵件
- 使用者的生物特徵

登入過程- authorization授權

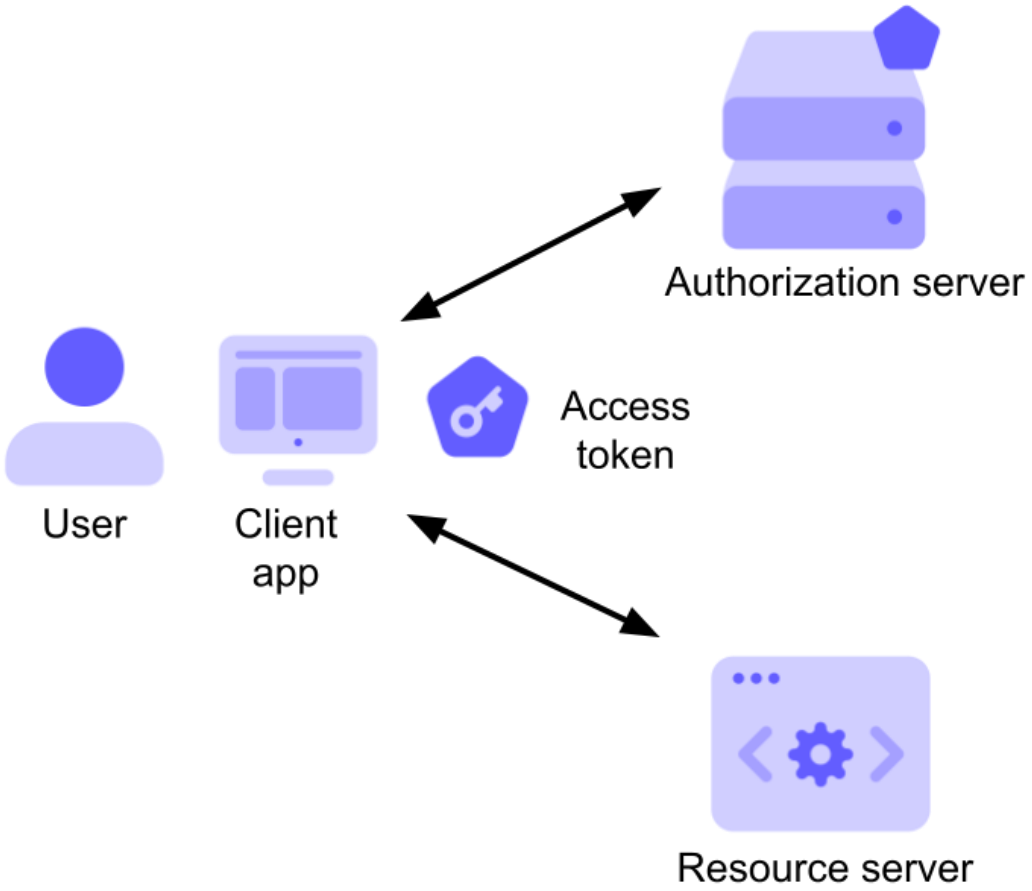
實際場景

- 銀行發行的銀行卡，允許使用者從自動取款機取款或購物。
- 門禁卡由物業管理公司分發，允許居民進入建築物或進入某些區域
- 鑰匙由房東提供，允許租戶進入他們的家

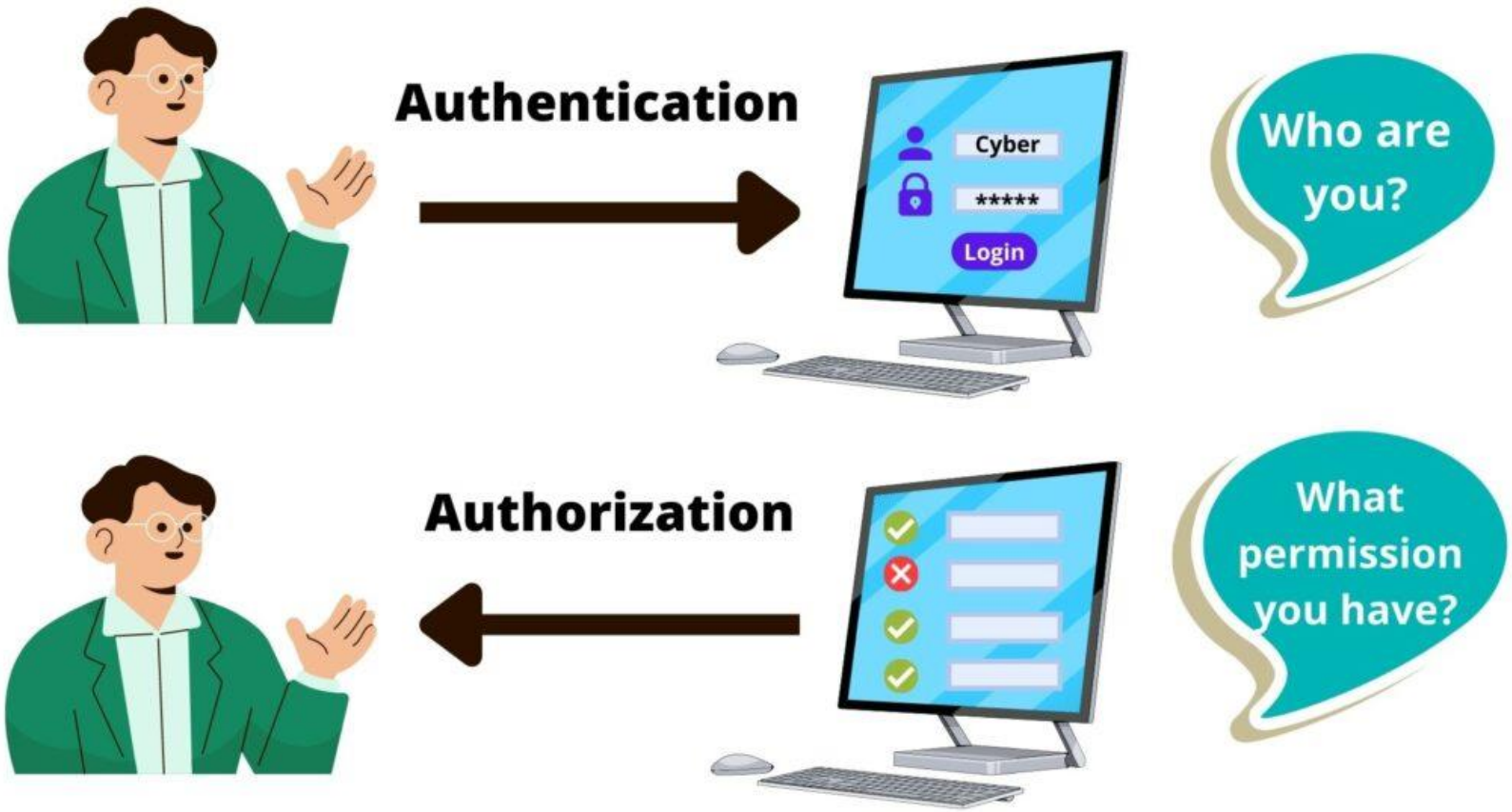
網路上

- Sessions
- Cookie
- tokens

登入過程- **authentication** 身分驗證

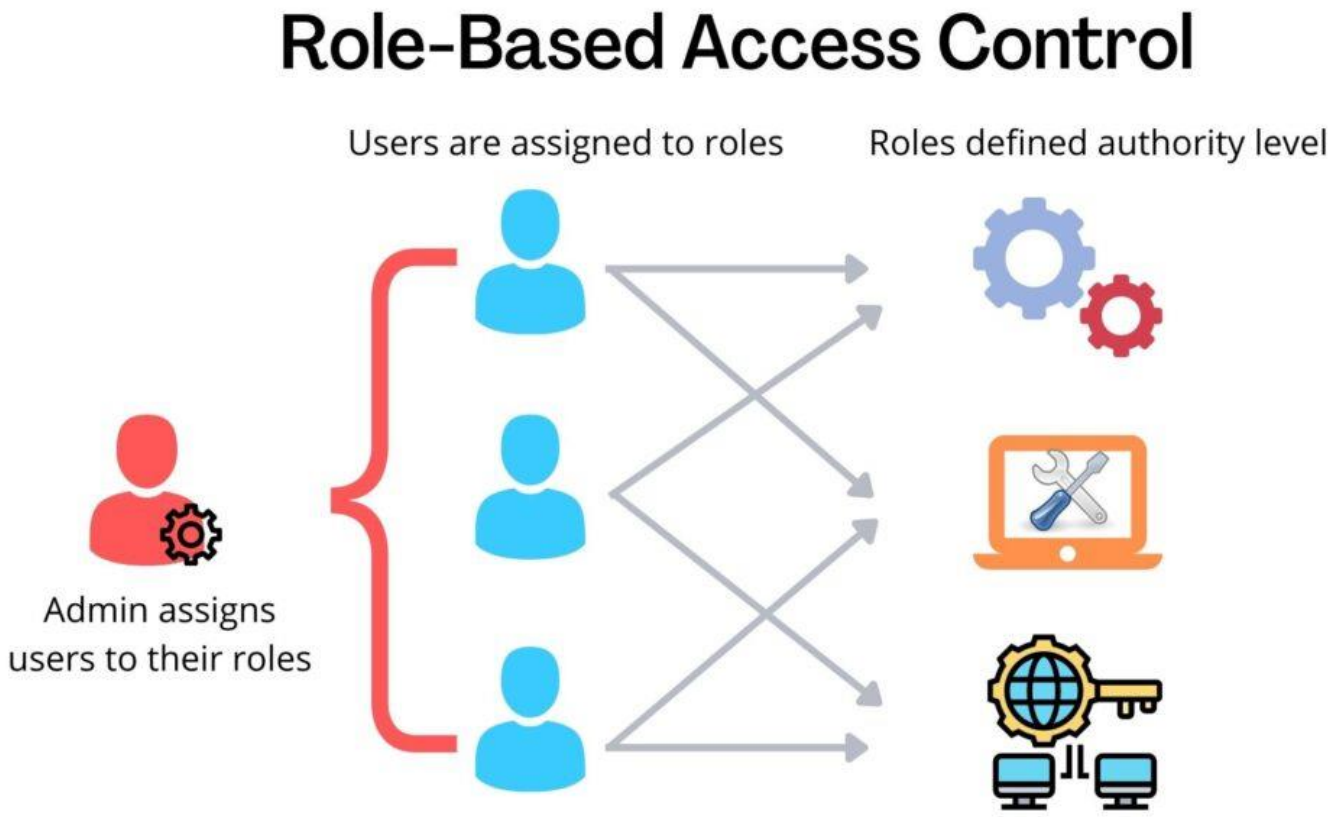


身份驗證和授權比較


Authentication
Cyber
*****
Login
Who are you?
Authorization
What permission you have?

登入過程- access control訪問控制

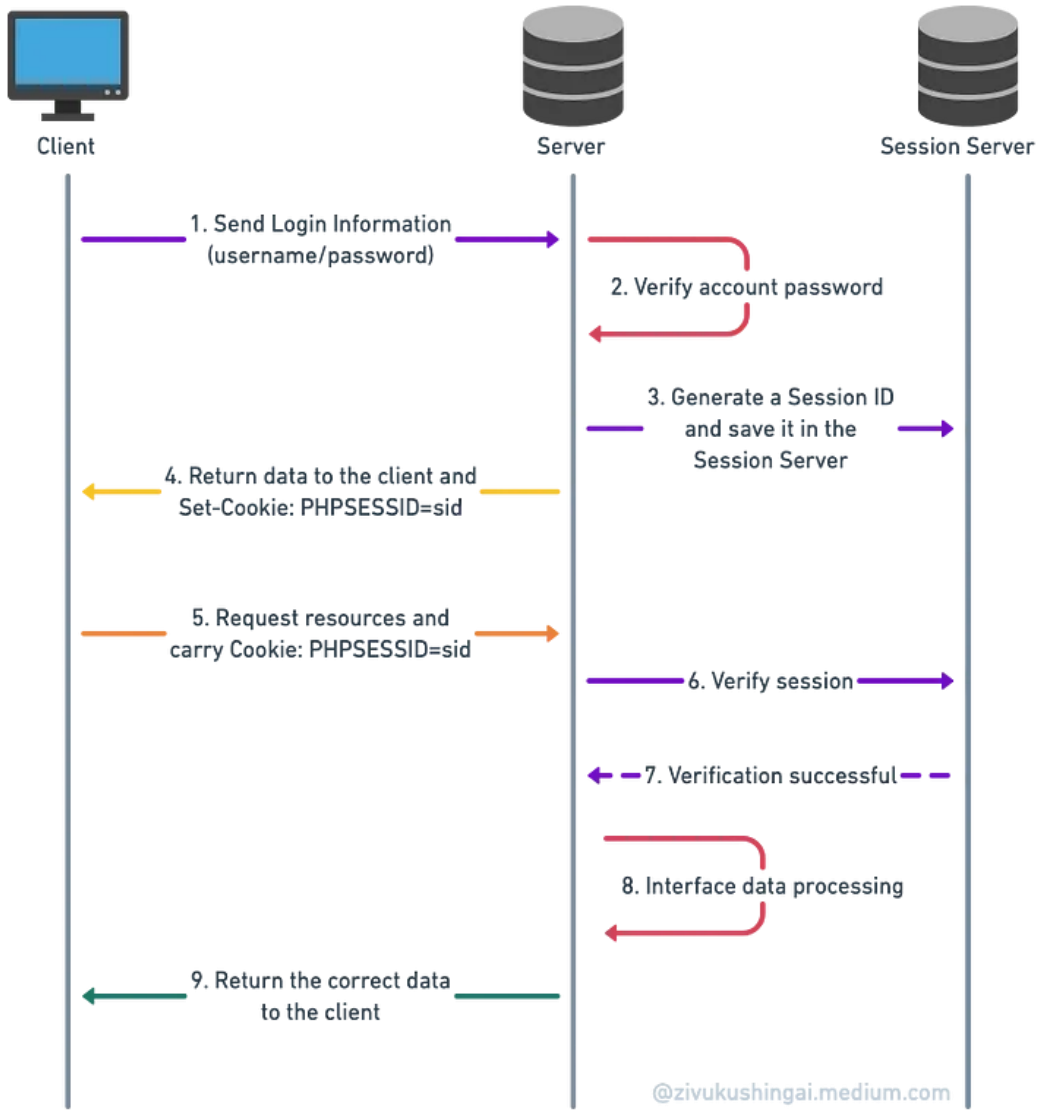
- 它有助於保護系統和數據免受未經授權的訪問。
- 使許可權保持最新可能很困難



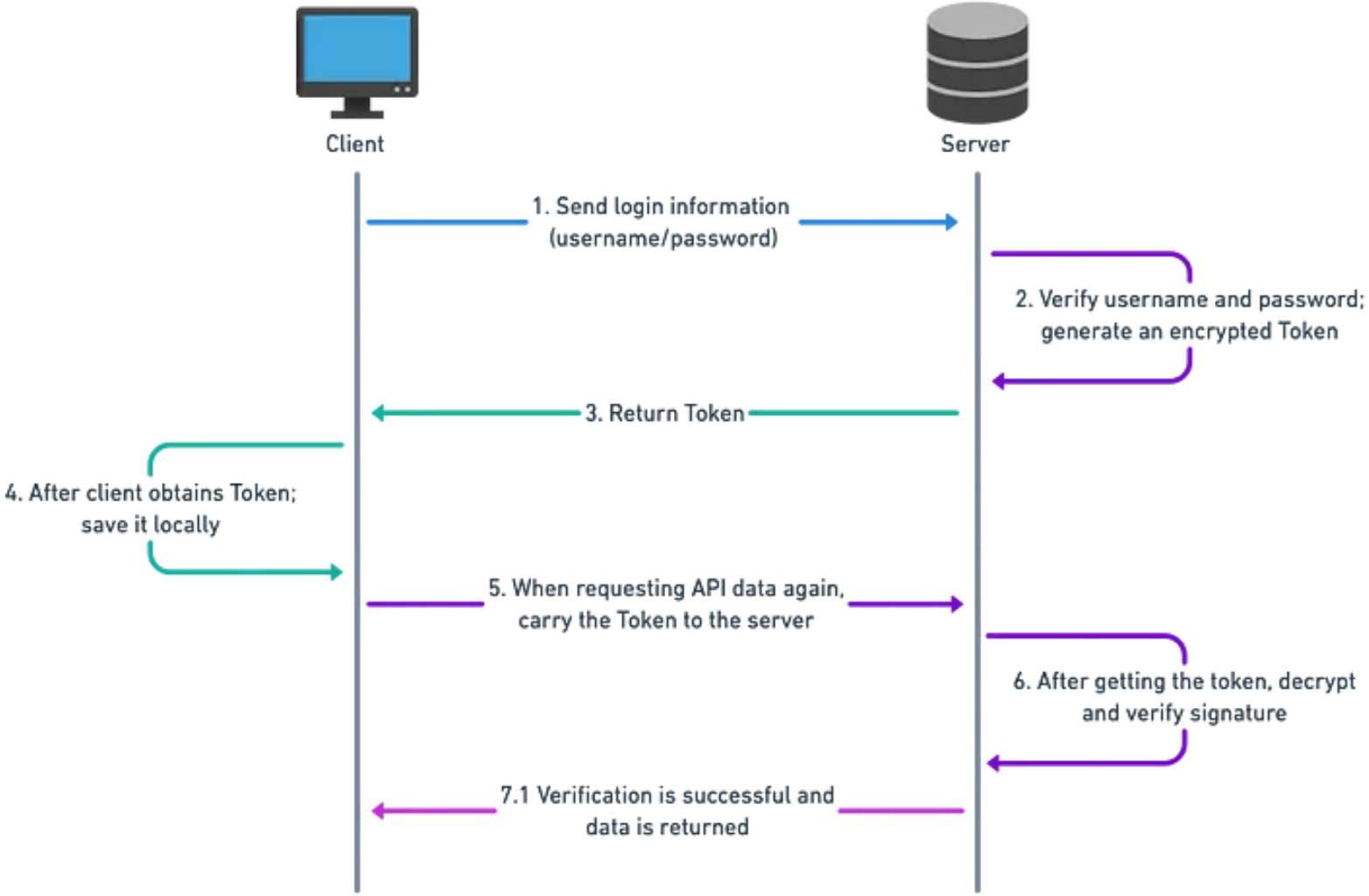
有哪些授權機制

- Session&cookie
- token

Session-Cookie Authentication身份驗證流程圖



Token Authentication令牌認證



什麼是token?

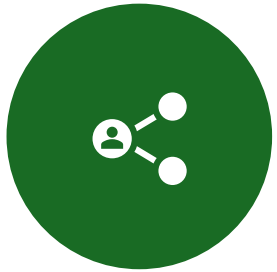
- 當客戶端存取伺服器時，伺服器會在通過驗證後為其頒發令牌。之後，用戶端可以攜帶令牌訪問伺服器，伺服器只需要驗證令牌的有效性即可。

令牌由以下元件組成

- UID（唯一使用者識別碼）
- time（目前時間的時間戳）
- 符號（簽名，使用雜湊演算法將權杖的前幾個字元壓縮成固定長度的十六進位字串）

Tokens的優點

無狀態和可擴展的伺服器：權杖機制消除了在伺服器上存儲會話資訊的需要，因為權杖本身包含與使用者相關的資訊。這允許在多個服務之間輕鬆共享用戶狀態。



支援行動裝置：權杖相容APP行動裝置。

增強的安全性：權杖可有效緩解 CSRF 攻擊，因為它們不依賴於 COOKIE。

跨程式調用支援：與 COOKIE 不同，權杖可用於跨域訪問而不會出現問題。

Tokens的缺點

1

所需的協調：權杖實現需要前端和後端之間的協調。

2

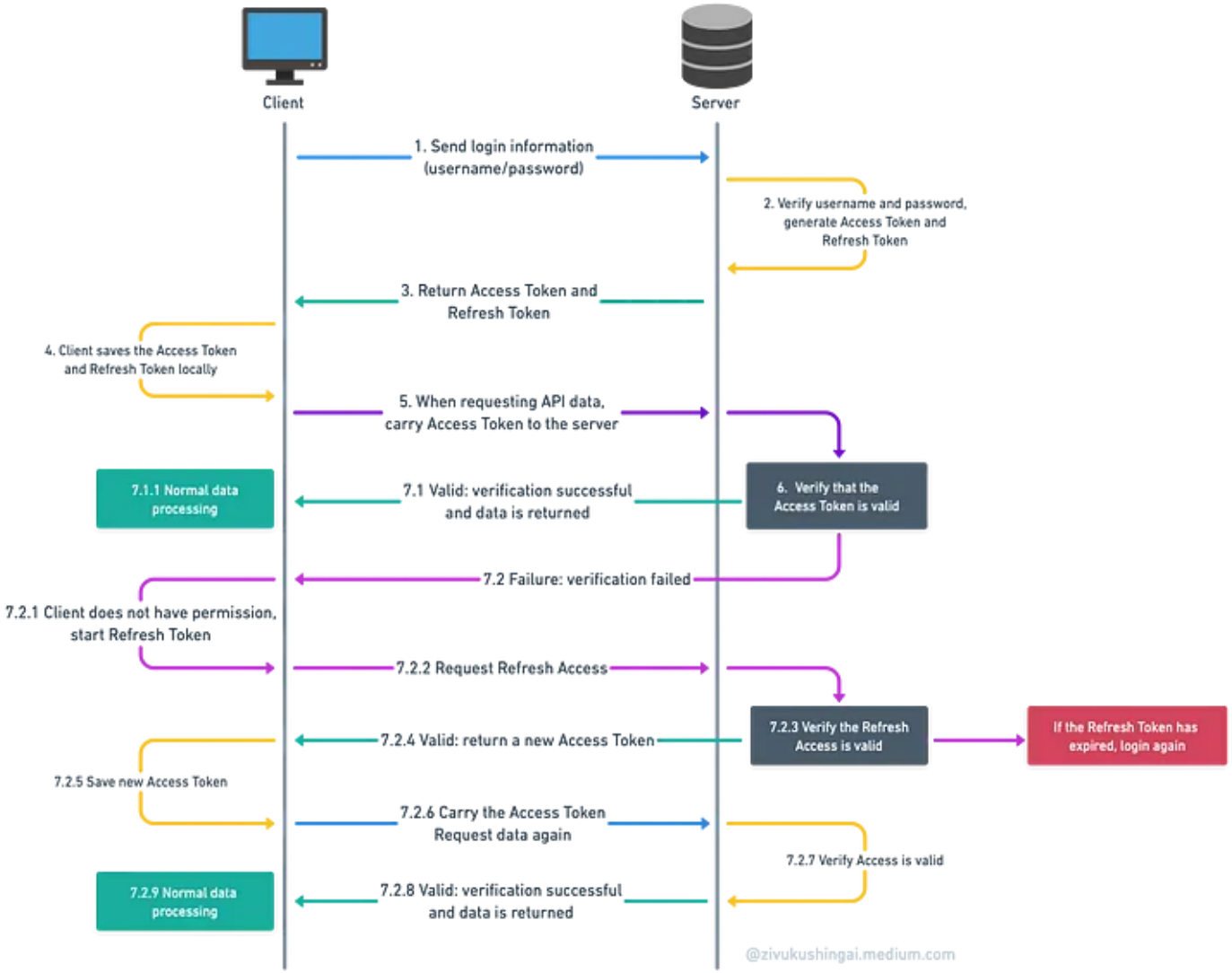
帶寬消耗：與session IDs (sids)相比，權杖通常具有更大的大小，從而導致流量和頻寬使用量增加。

3

性能影響：雖然權杖驗證消除了對資料庫或遠端服務訪問的需要，但它仍然需要加密和解密等操作，這可能會影響性能。

4

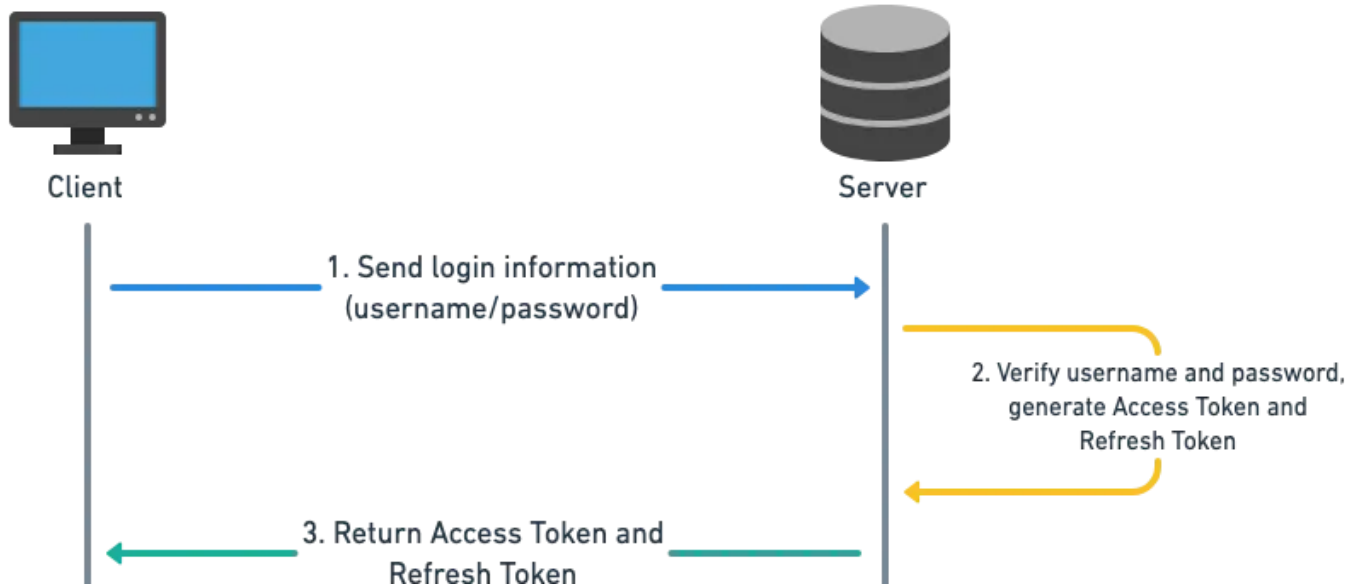
有效期短：為了最大限度地降低權杖被盜的風險，權杖通常設置了較短的有效期。刷新權杖通常用於解決此問題。



更新權杖 **refresh token**

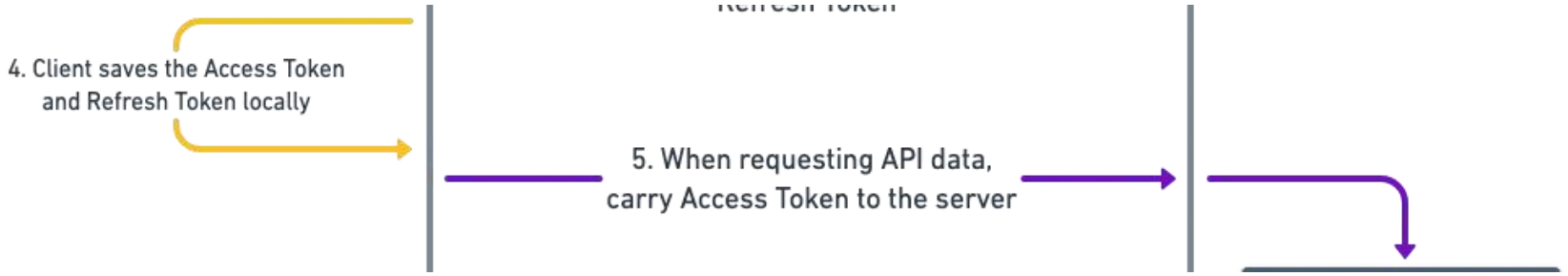
更新權杖過程(1/4)

- 用戶端：客戶端輸入使用者名和密碼以請求登錄驗證。
- 伺服器：收到請求后，伺服器會驗證使用者名和密碼。如果驗證成功，伺服器將向用戶端頒發訪問令牌和刷新令牌。



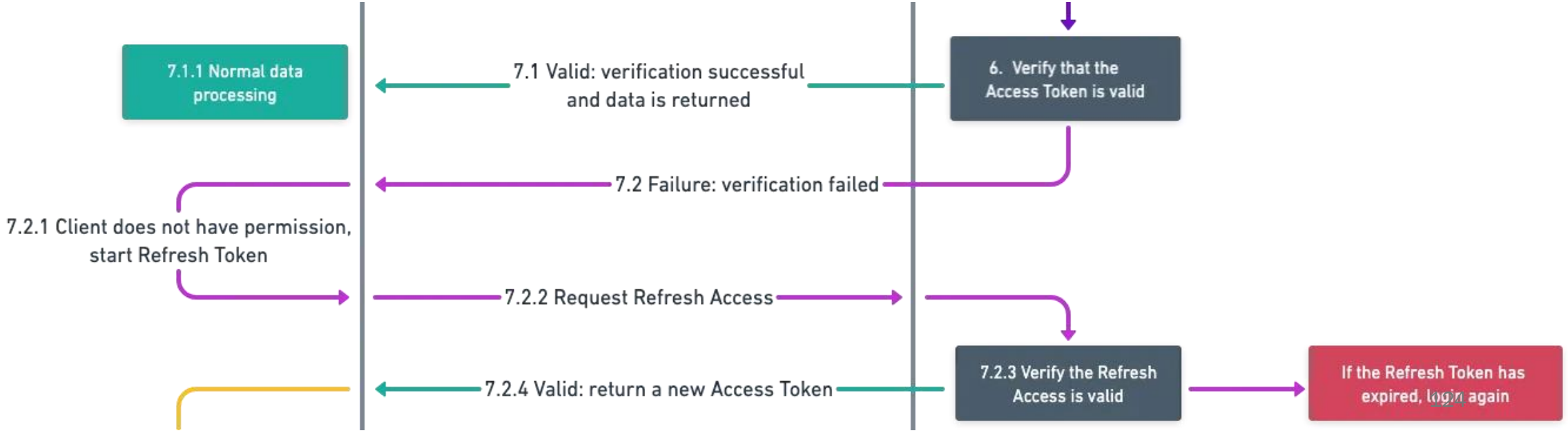
更新權杖過程(2/4)

- 用戶端：用戶端將訪問令牌和刷新令牌存儲在本地。
- 用戶端發送請求：請求數據時，用戶端在請求中包含訪問令牌並將其發送到伺服器。



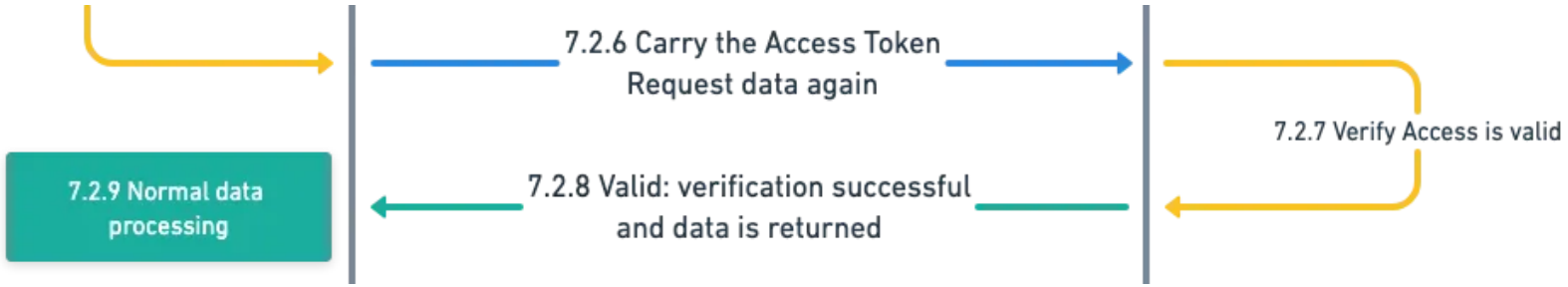
更新權杖過程(3/4)

- Server: 伺服器：
  - 驗證訪問令牌是否有效：正常返回數據
  - 驗證訪問令牌是否已過期：拒絕請求



更新權杖過程(4/4)

- 用戶端（訪問令牌已過期）：用戶端將刷新令牌重新發送到伺服器。
- 用戶端：客戶端在請求中包含新的訪問令牌，並將請求重新發送到伺服器。

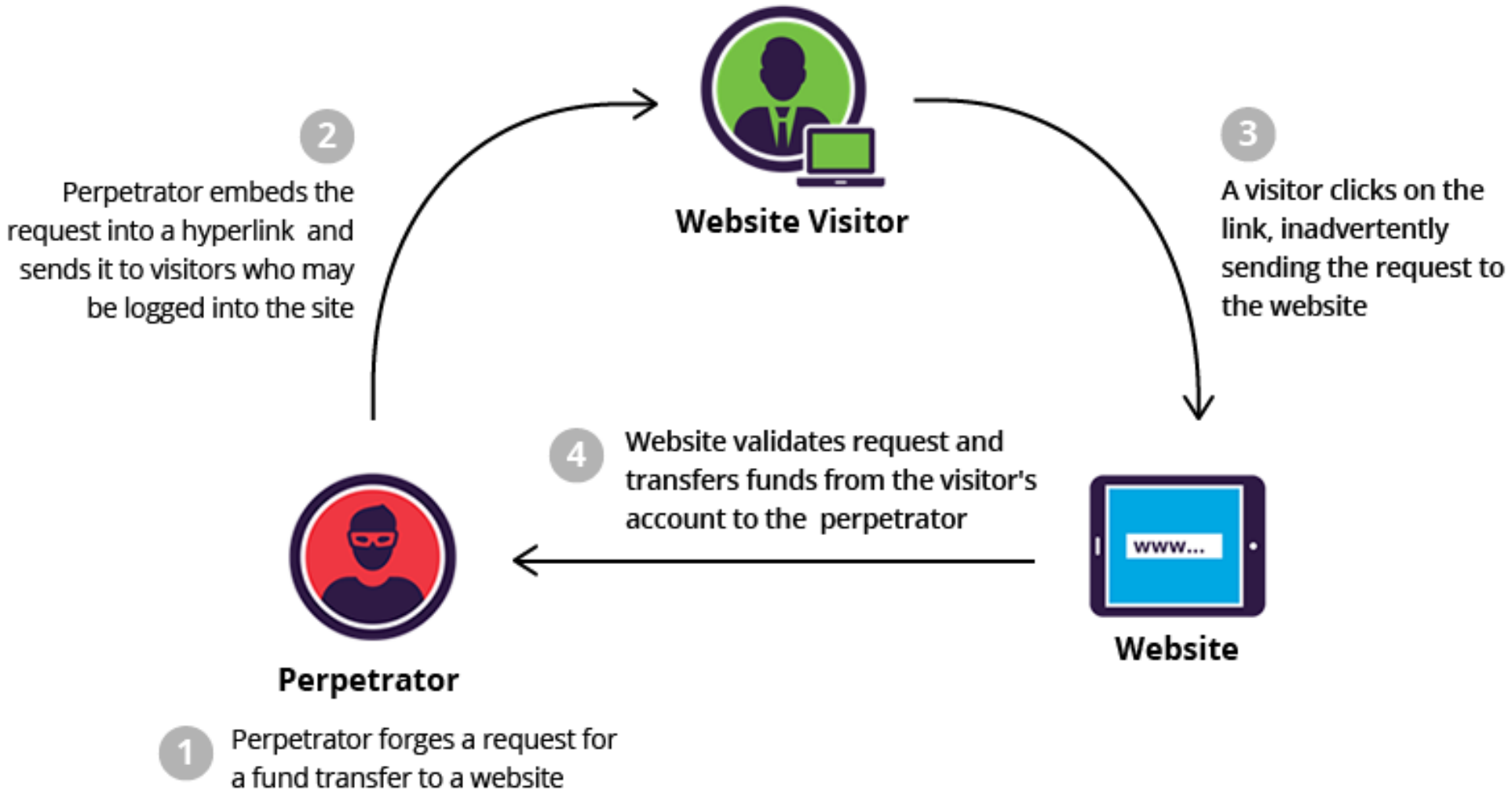


Cross Site Request Forgery (CSRF)

What is CSRF?

- 跨站請求偽造，也被稱為 one-click attack 或 session riding，通常縮寫為 CSRF 或 XSRF。

舉例:

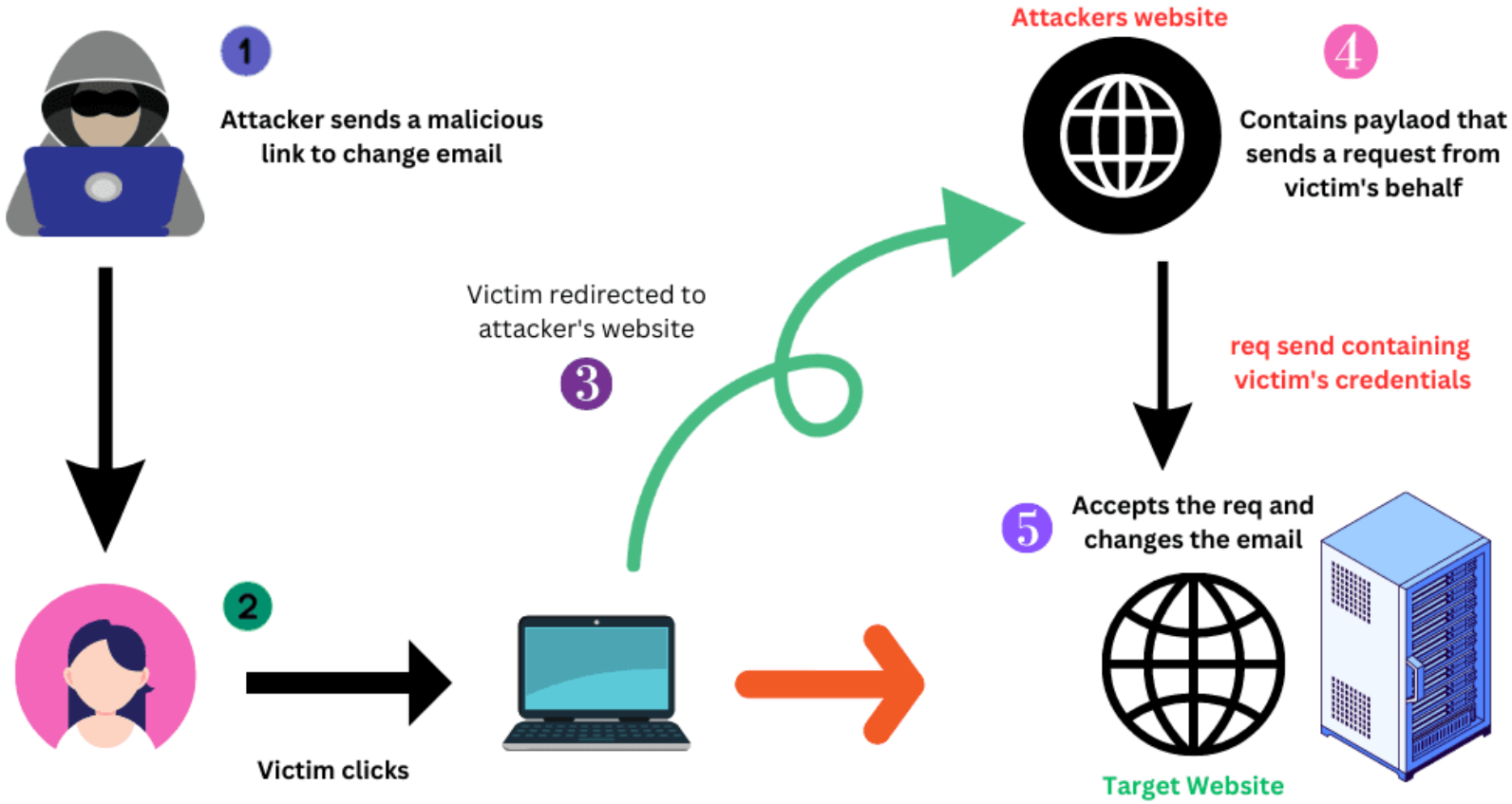

2
Perpetrator embeds the request into a hyperlink and sends it to visitors who may be logged into the site
Website Visitor
3
A visitor clicks on the link, inadvertently sending the request to the website
4
Website validates request and transfers funds from the visitor's account to the perpetrator
www...
Website
Perpetrator
1
Perpetrator forges a request for a fund transfer to a website

為什麼會產生CSRF漏洞

- 開發者編寫web不夠謹慎
- web瀏覽器對於cookie和http身份驗證還有session資訊的處理存在著缺陷

- 導致
  - 攻擊者透過盜用使用者的身份，悄悄發送一個請求或是執行一些惡意行為

發現CSRF


1
Attacker sends a malicious link to change email
Attackers website
4
Contains paylaod that sends a request from victim's behalf
Victim redirected to attacker's website
3
req send containing victim's credentials
5
Accepts the req and changes the email
2
Victim clicks
Target Website

同源政策 (Same Origin Policy) 與跨網域 (CORS)

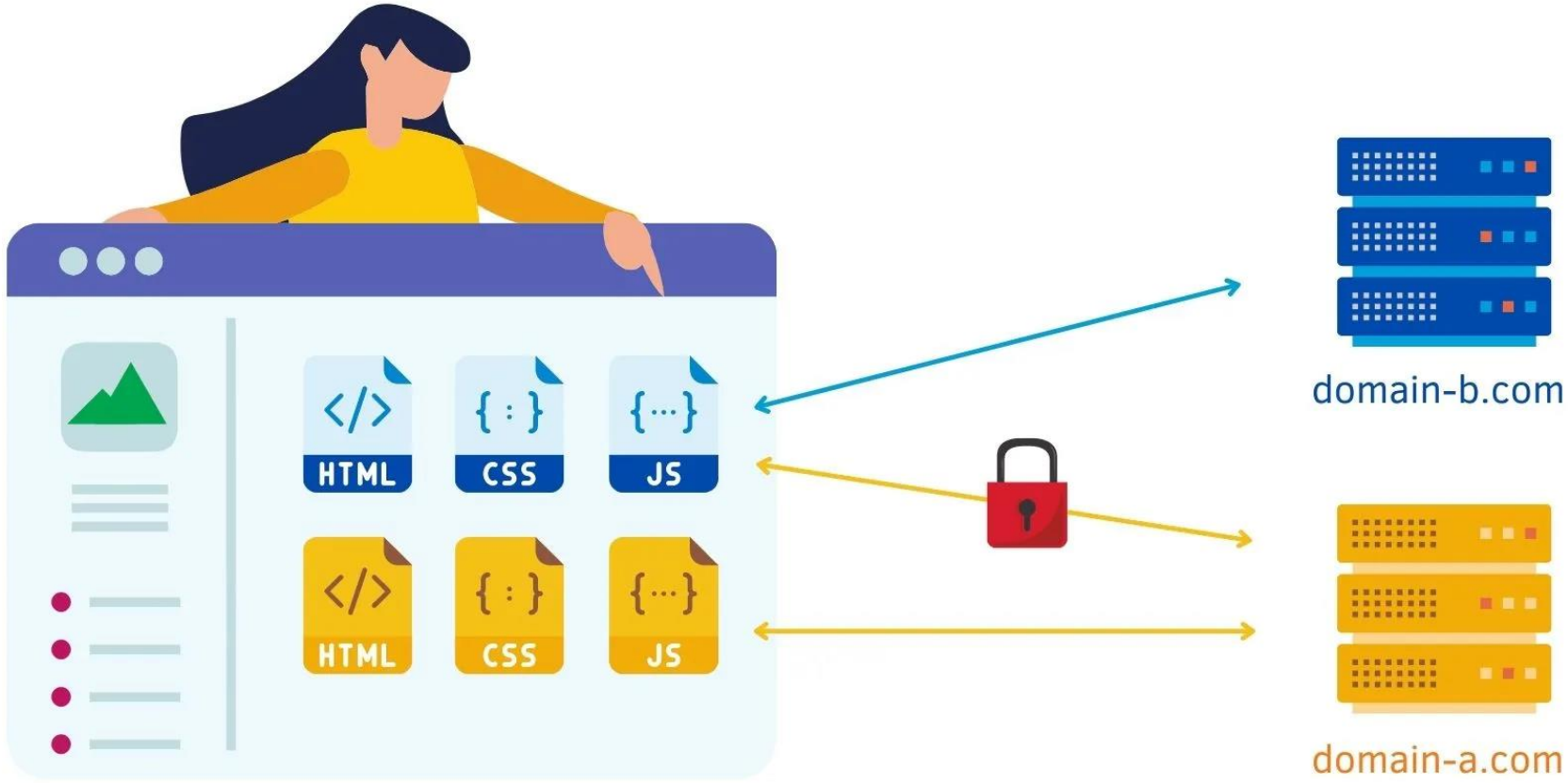
你可能在開發時遇到

```
Access to fetch at 'http://localhost:8000' from origin 'https://www.xxx.com'
has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is
present on the requested resource. If an opaque serves your needs, set the
request's mode to 'no-cors' to fetch the resource with CORS disabled.
```

```
Access to fetch at 'http://localhost:8000' from origin 'https://www.xxx.com'
has been blocked by CORS policy: The 'Access-Control-Allow-Origin' header
containers multiple values 'https://www.xxx.com', but only one is allowed.
Have the server send the header with a valid value, or, if an opaque serves
your needs, set the request's mode to 'no-cors' to fetch the resource with
CORS disabled.
```

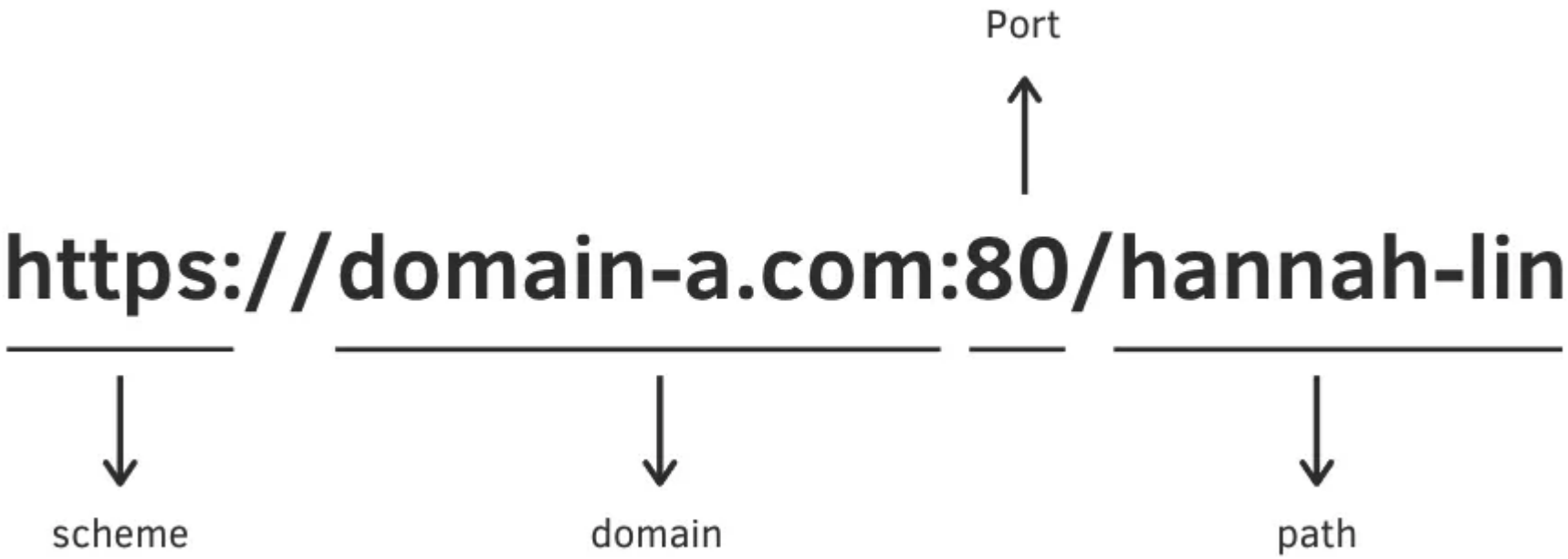
```
Access to fetch at 'http://localhost:8000' from origin 'https://www.xxx.com'
has been blocked by CORS policy: The 'Access-Control-Allow-Origin' header
has a value 'https://www.xxx.com'that is not equal to the supplied origin.
Have the server send the header with a valid value, or, if an opaque serves
your needs, set the request's mode to 'no-cors' to fetch the resource with
CORS disabled.
```

同源政策 (Same Origin Policy)



怎麼判斷同源?

- 只要 scheme、domain、port 一樣就會被視為同源 (same-origin)，其他則是不同源



但還是有例外

- 跨來源嵌入通常被允許 (embed)
  - <img>、<video>
- 跨來源寫入通常被允許 (writes)
  - <form>
- **跨來源讀取通常被禁止 (*reads*)**
  - domain-a.com 不能讀取 domain-b.com 的 cookie、XMLHttpRequest ，Fetch API 也都無法被讀取，會回報錯誤

CORS (Cross-Origin Resource Sharing)

- Access-Control-Allow-Origin: *

- // 還是會針對特定網站開權限
- Access-Control-Allow-Origin: https://foo.example
- // 可以設定允許哪些 method，defult 是全部方法
- Access-Control-Request-Method: POST, GET
- // 允许 client side 带 cookie 等驗證，defult 是 false
- Access-Control-Allow-Credentials: true

Poxy

Poxy 反向代理

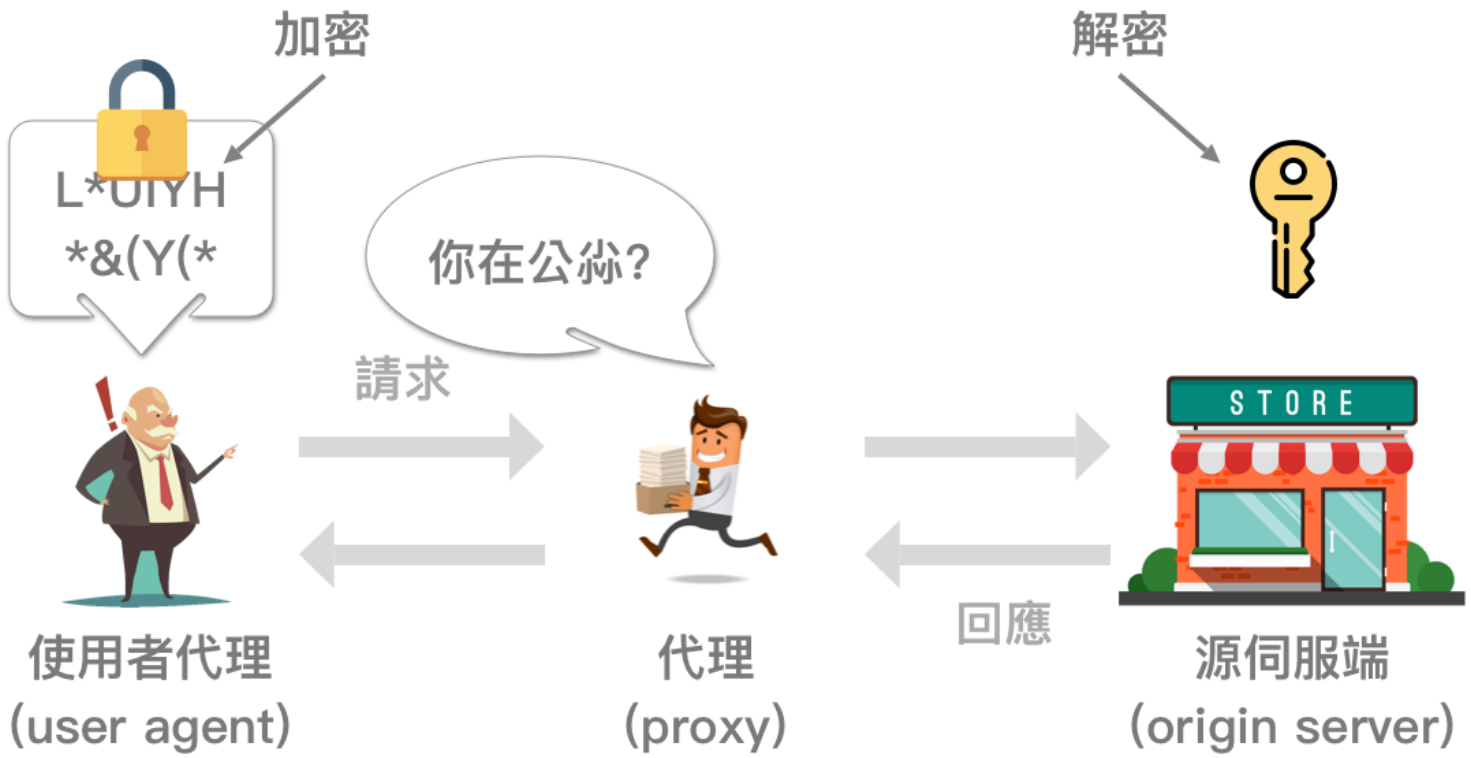
- 透過緩存，減少回應時間、減輕源站負擔
- 隱藏源站真實IP地址，避免遭受攻擊
- 過濾流量，確保源站安全
- 負載均衡，擴展業務、排除故障伺服器

Poxy -nginx

```
17  http {
34
35      server {
36          listen       80;
37          server_name  localhost;
38
39          #charset koi8-r;
40
41          #access_log  logs/host.access.log  main;
42          #root "C:\Users\WMSLAB\Downloads\led-light-v3\dist";
43          #index index.html;
44
45          location / {
46              #root   html;
47              #index  index.html index.htm;
48              root "C:\Users\WMSLAB\Downloads\led-light-v3\dist";
49              index index.html;
50
51          }
52          location /api {
53              proxy_pass http://localhost:3000/;
54              proxy_http_version 1.1;
55              proxy_set_header Host $http_host;
56              proxy_set_header Upgrade $http_upgrade;
57              proxy_set_header Connection 'upgrade';
58              proxy_set_header Host $host;
59              proxy_cache_bypass $http_upgrade;
60          }
```

TLS Problem

- 欲使用 傳輸層安全協議 (TLS) 進行安全連線 (i.e., HTTPS)，Client 與 Server 在建立連線 (TCP 三向交握) 後，需進行 TLS/SSL handshake 交換加密資訊



其他 驗證方式

JWT MFA

- 簡單穩定的方式來表示身份驗證和授權數據。這篇簡單介紹什麼是JWT和為什麼要使用JWT認證機制。

- 在測試後端 API 時，在註冊或登入成功後，我們會看到後端伺服器回傳一組稱為「token」的字串，現在讓我們簡單介紹一下什麼是 token，以及後端採用的 JWT 實作機制：

JWT

- JWT（JSON Web Token）
- **JWT 的主要作用是方便客戶端與伺服器之間的身份驗證。使用 JWT 可以在不需要每次登入的情況下，在客戶端與伺服器之間安全地傳遞封裝身份信息。它還可以用於許多其他用途，例如串接多個服務，並將數據在服務間安全地傳遞。**

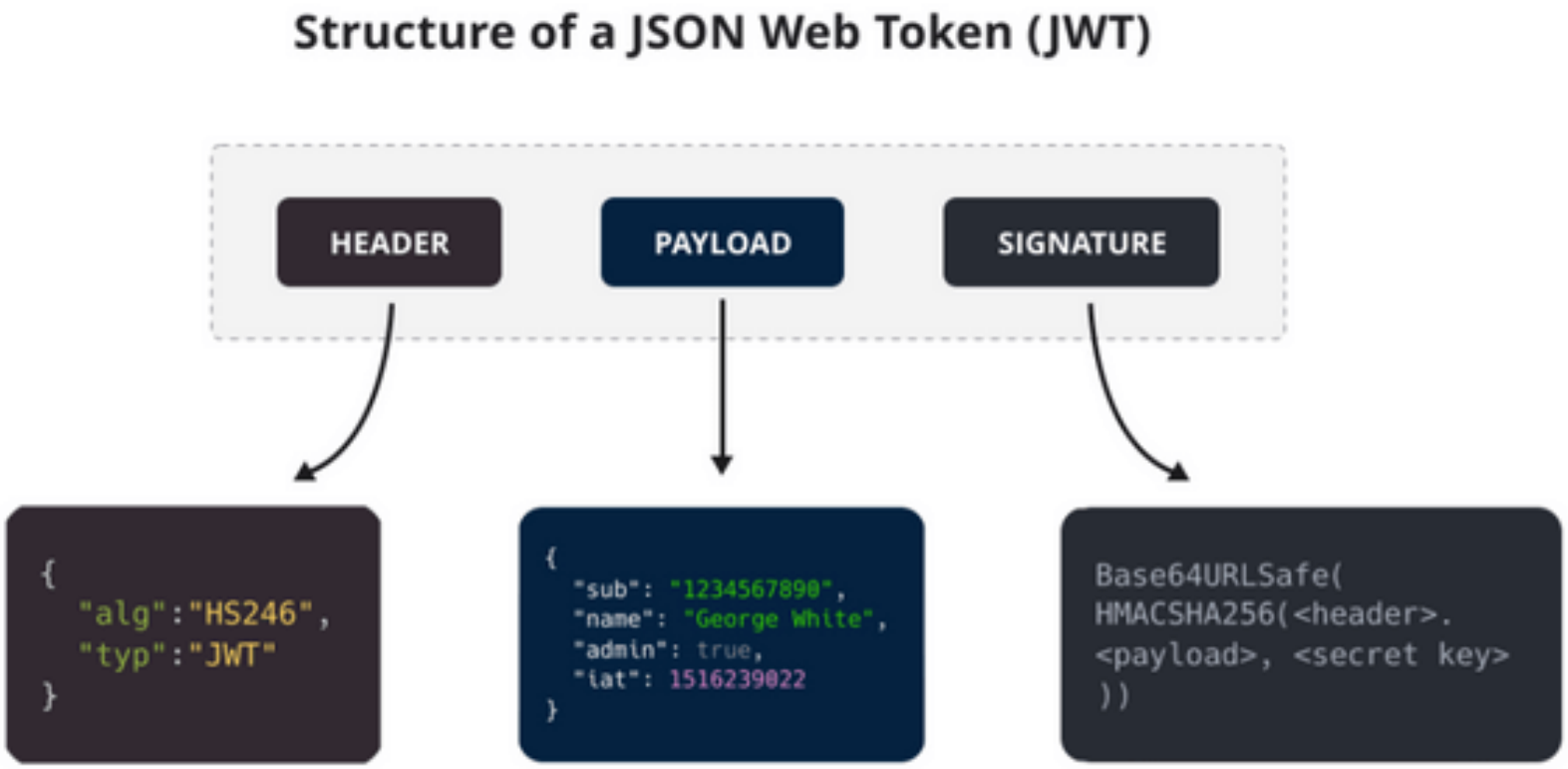
JWT 原理

- 驗證成功後發出的憑證包含
  - Header – 標記 token 的類型與雜湊函式名稱。
  - Payload – 要攜帶的資料，例如 user_id 與時間戳記，也可以指定 token 的過期時間。
  - Signature – 根據 Header 和 Payload，加上密鑰 (secret) 進行雜湊，產生一組不可反解的亂數，當成簽章，用來驗證 JWT 是否經過篡改。

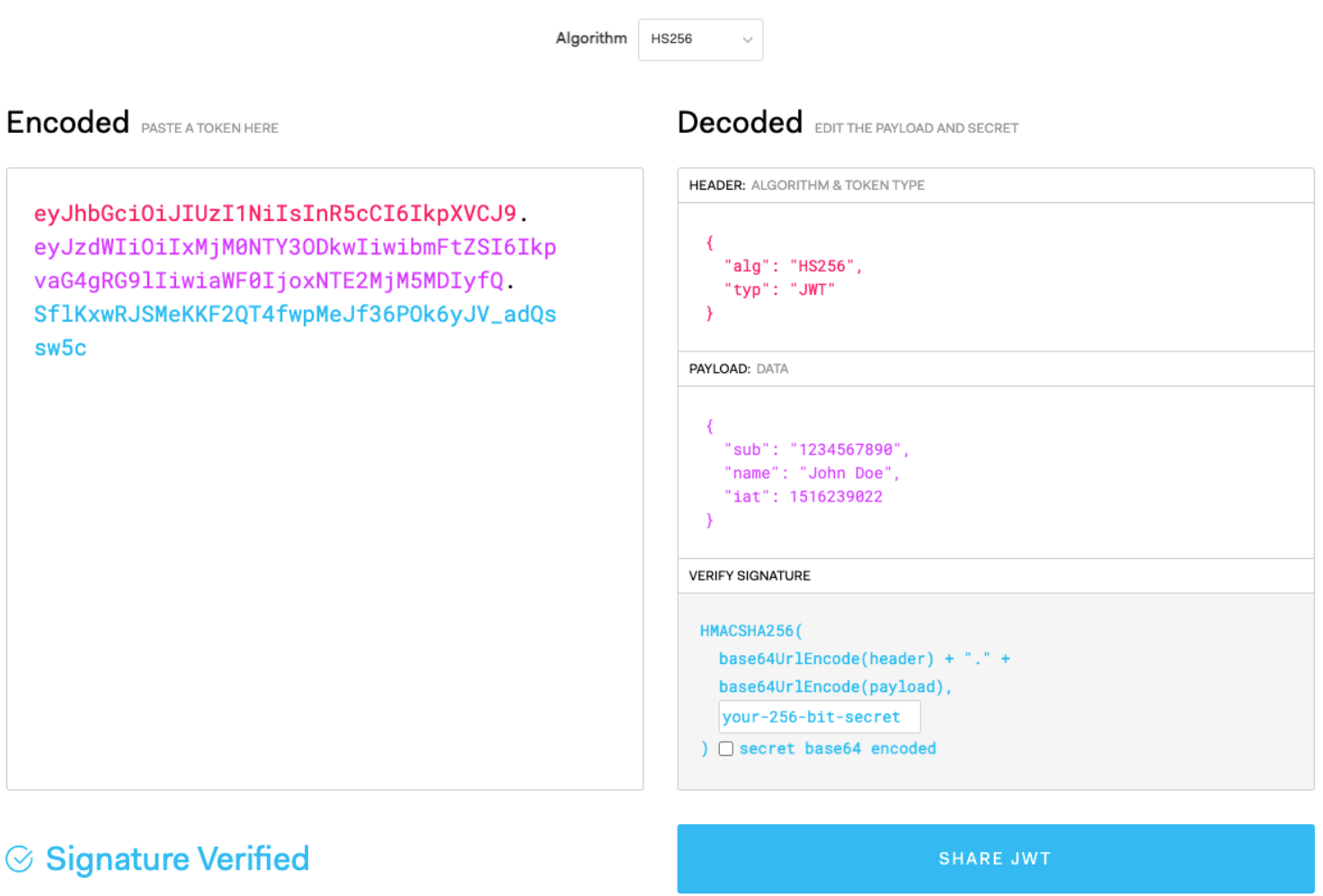
JWT產生的是一個字串

- JSON Web Tokens(JWT)，它就是個字串，格式如下：

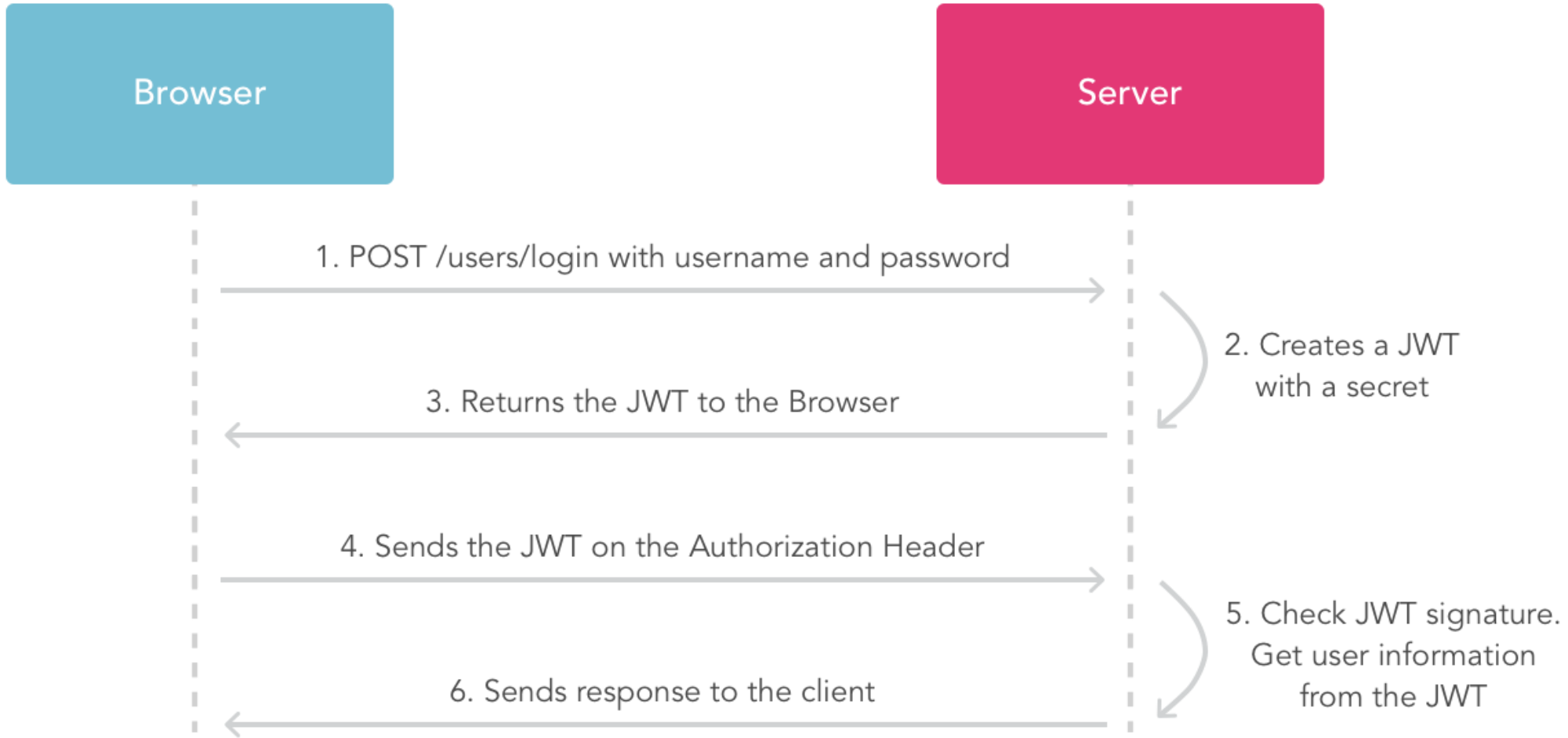
  HEADER.PAYLOAD.SIGNATURE



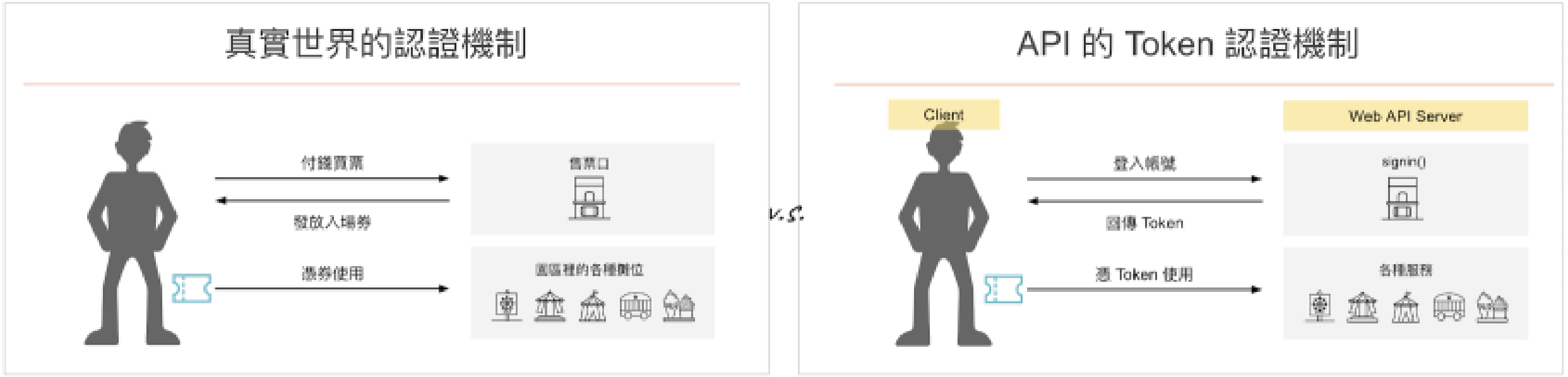
解碼示範:https://jwt.io/



Server 收到後會去檢查 JWT token 是否有效，在發 Request 時，在 Header 加入 Authorization: <token>



JWT認證


真實世界的認證機制
付錢買票
發放入場券
售票口
憑券使用
園區裡的各種設位
v.s.
API 的 Token 認證機制
Client
Web API Server
登入帳號
回傳 Token
signin()
憑 Token 使用
各種服務

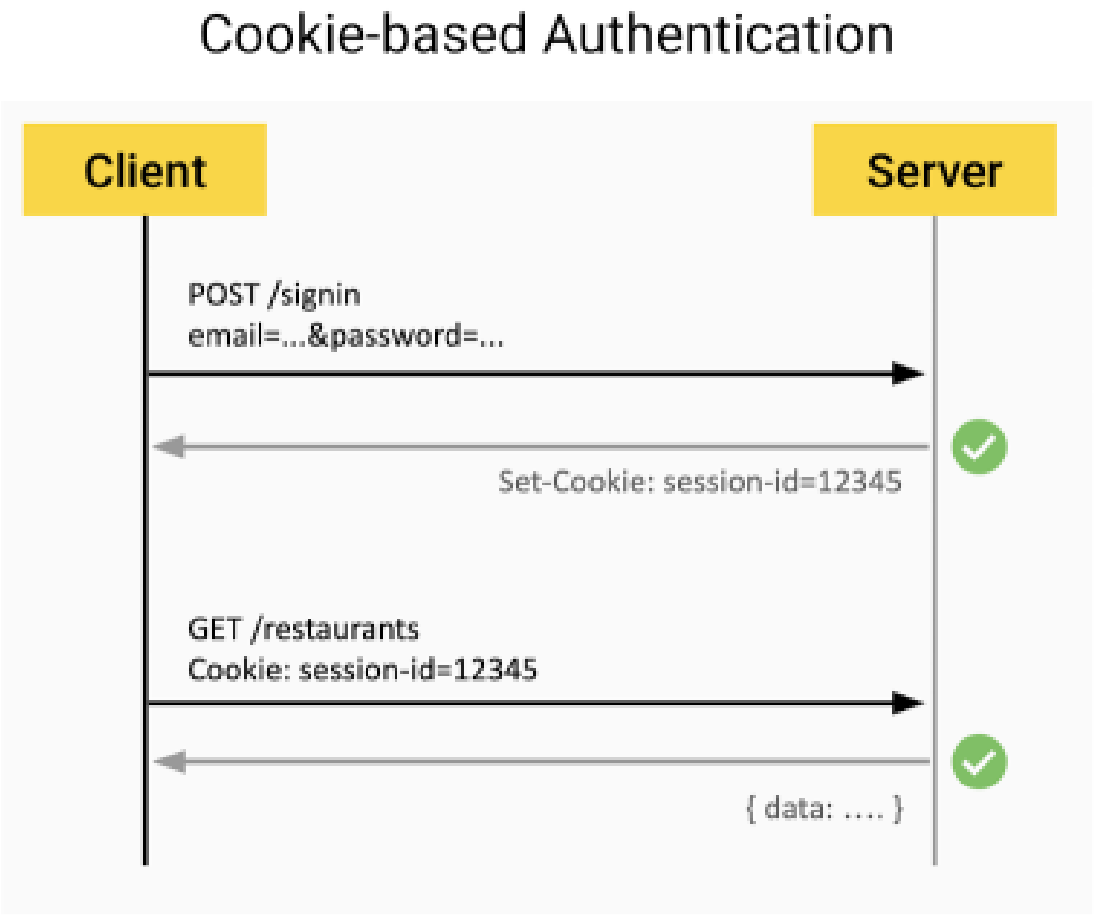
如何防止 **JWT** 被篡改？

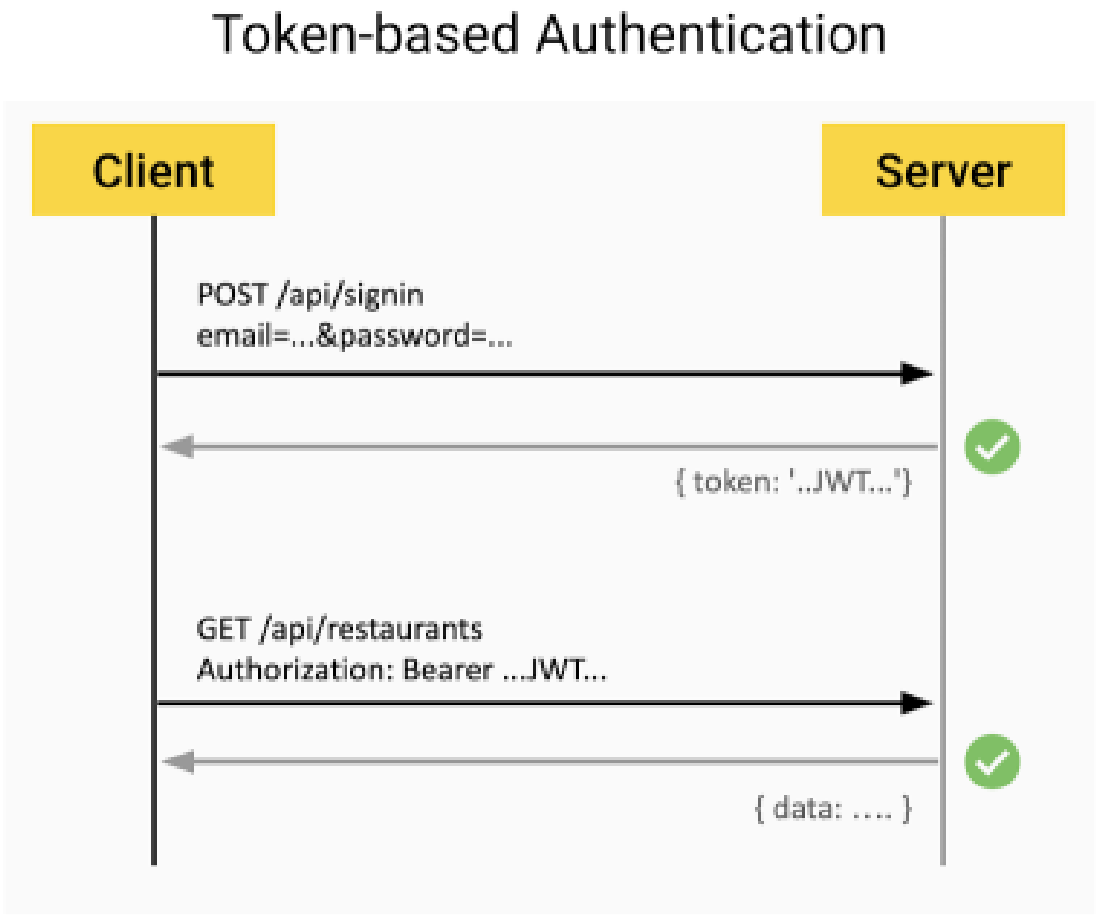
- 因為服務端拿到 JWT 之後，會解析出其中包含的 Header、Payload 以及 Signature 。服務端會根據 Header、Payload、金鑰再次生成一個 Signature。拿新生成的 Signature 和 JWT 中的 Signature 作對比，如果一樣就說明 Header 和 Payload 沒有被修改。

建議

- 使用安全係數高的加密演算法
- 使用成熟的開源庫
- JWT 存放在 localStorage 中而不是 Cookie 中，避免 CSRF 風險
- 一定不要將隱私資訊存放在 Payload 當中
- 金鑰一定保管好，一定不要洩露出去。JWT 安全的核心在於簽名，簽名安全的核心在金鑰
- Payload 要加入 exp （JWT 的過期時間），永久有效的 JWT 不合理。並且，JWT 的過期時間不易過長

比較cookie與JWT的驗證流程



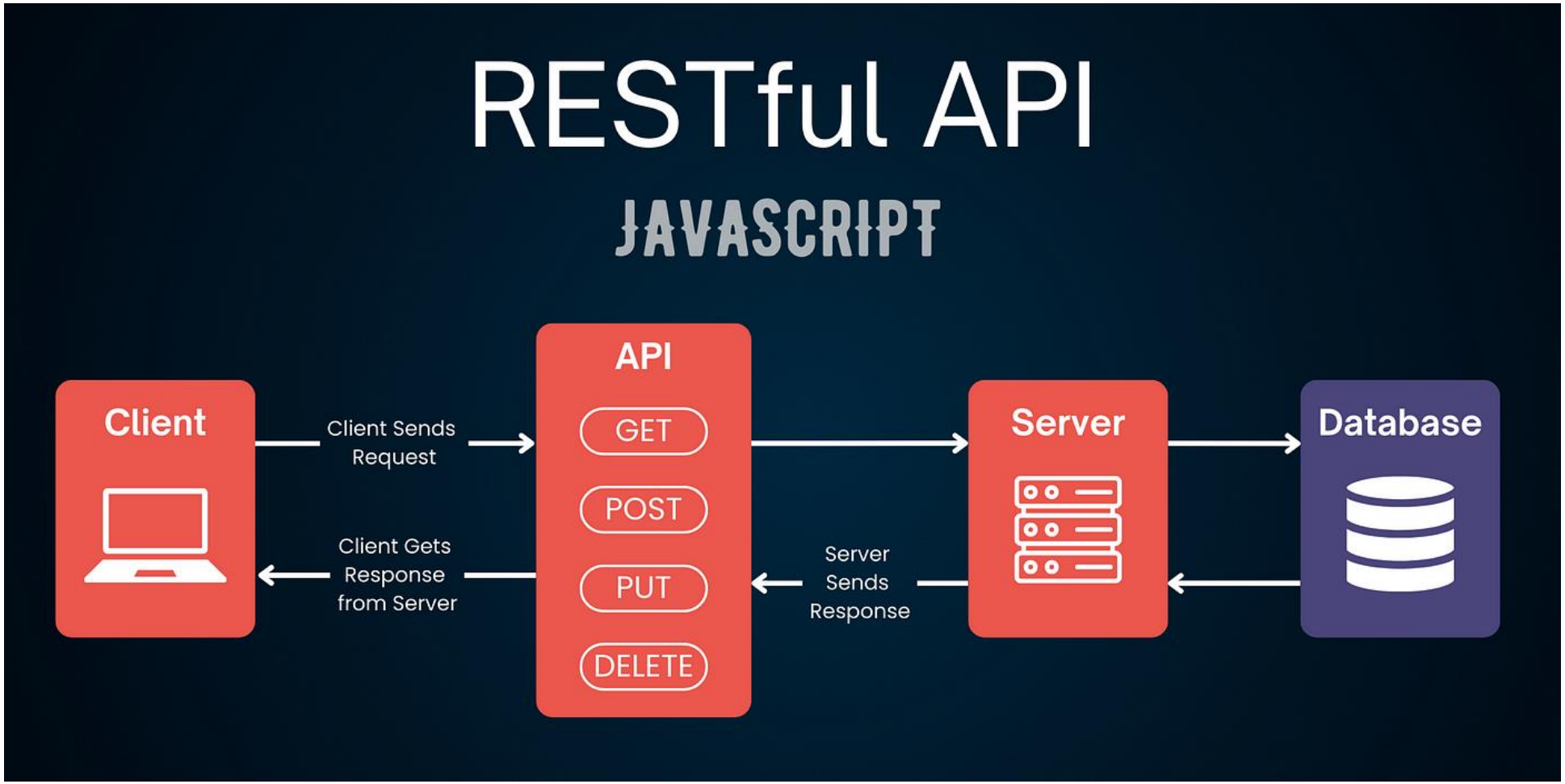


Multifactor Authentication (MFA) Model
多重身份驗證（MFA）模型

- 減慢攻擊者速度
- 機制
  - SMS
  - 身份驗證器應用程式
  - SecurityTokens

推薦的開發風格

RESTful


RESTful API
JAVASCRIPT
Client
Client Sends Request
Client Gets Response from Server
API
GET
POST
PUT
DELETE
Server Sends Response
Server
Database

一般的 API vs RESTful API

一般的 API

```
獲得資料GET     /getData
新增資料POST    /createData
刪除資料DELETE  /deleteData/1
```

RESTful API

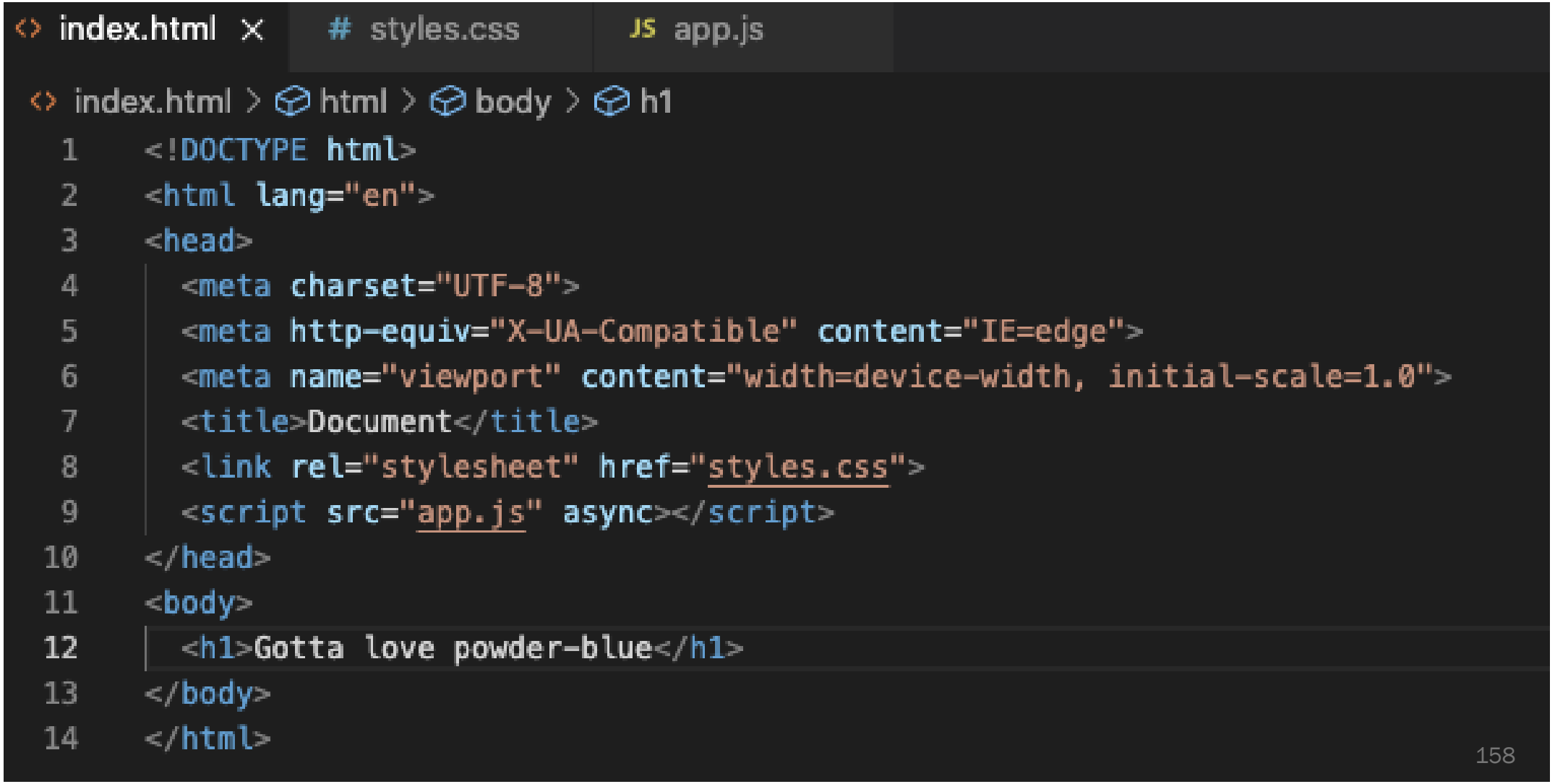
```
獲得資料GET     /data
新增資料POST    /data
刪除資料DELETE  /data/1
```

如何實現 **RESTful API**

URL	HTTP Verb	POST Body	Result
/api/movies	GET	empty	Returns all movies
/api/movies	POST	JSON String	New movie Created
/api/movies/:id	GET	empty	Returns single movie
/api/movies/:id	PUT	JSON string	Updates an existing movie
/api/movies/:id	DELETE	empty	Deletes existing movie

Web滲透

Html,css, javascript架構



網站滲透

- 淺談滲透測試前的資訊收集
- 文件洩露
- SQL注入
- XSS
- CSRF
- 暴力破解
- 文件上傳
- 邏輯漏洞

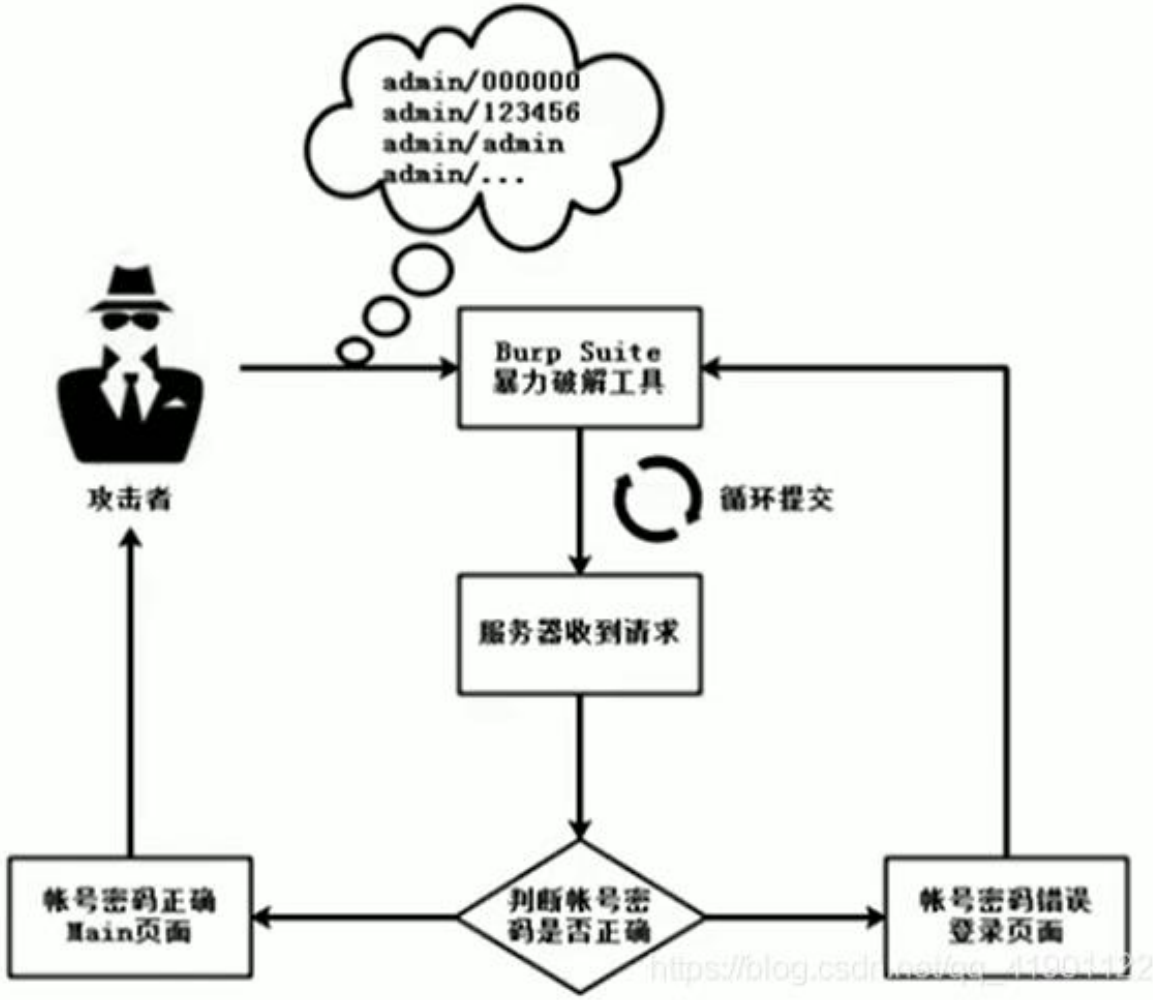
網站滲透-文件洩露

機敏資料洩漏

指管理員將網站備份檔案或是敏感資訊檔存放在某個網站目錄下，駭客可通過暴力破解等方法獲取該備份檔案，導致網站敏感資訊洩露

網站滲透-暴力破解

- 因為伺服器端沒有做限制，導致攻擊者可以通過暴力破解的手段破解使用者所需要的資訊，如用戶名、密碼、驗證碼等。暴力破解需要有龐大的字典，暴力破解的關鍵在於字典的大小。



網站滲透-文件上傳

什麼是檔案上傳漏洞？
當網站允許使用者上傳文件到其檔案系統，且未對檔名、類型、內容或大小進行足夠的驗證時，就可能產生檔案上傳漏洞。如果不正確地實施這些限制，即使是簡單的圖像上傳功能也可能被用來上傳任意和潛在的危險文件。這可能允許遠程代碼執行的伺服器端腳本文件。

網站滲透-邏輯漏洞

主要有幾個原因

無條件信任使用者輸入的內容
過濾使用者輸入，但是沒有過濾完全，導致被繞過

商業邏輯漏洞影響
任意修改系統邏輯
影響到使用系統的使用者，如個資外洩

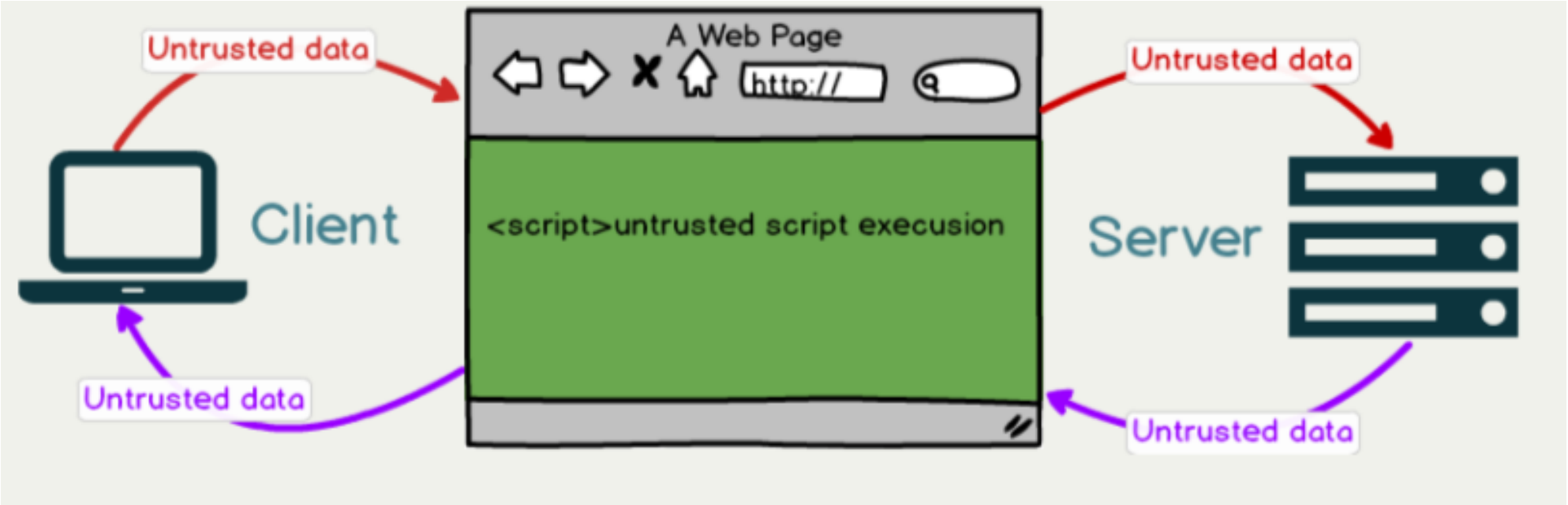
XSS

XSS

- Cross-Site Scripting(跨站腳本)
- 攻擊者可以使用 XSS 向毫無戒心的使用者發送惡意腳本
- 最終使用者的瀏覽器無法知道該腳本不應被信任，並且會執行該腳本
- 因為它認為腳本來自受信任的來源，所以惡意腳本可以訪問瀏覽器保留並用於該網站的任何 cookie、會話令牌或其他敏感信息
- 這些腳本甚至可以重寫 HTML 頁面的內容

Web 應用程式容易受到 XSS 攻擊...

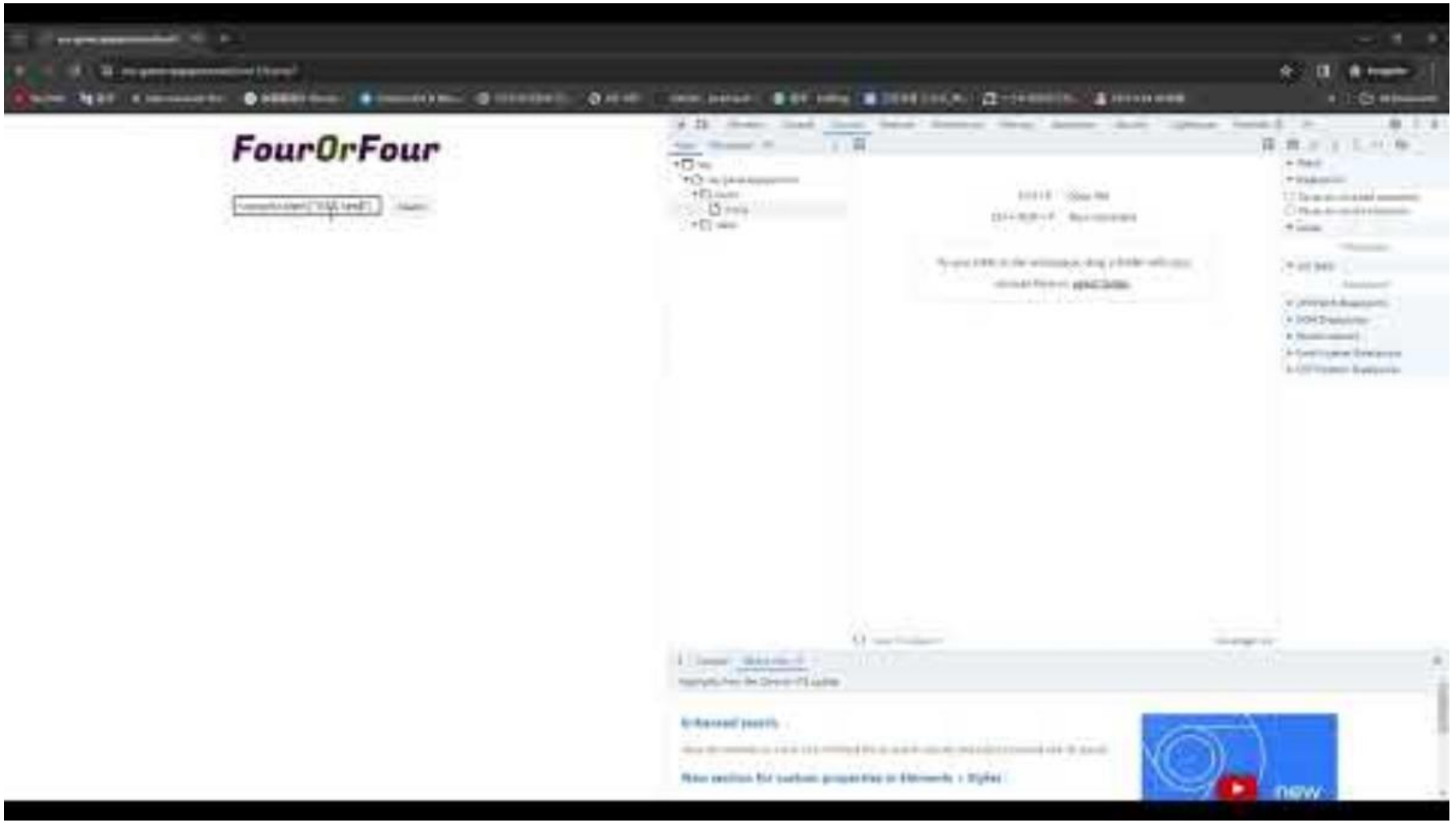
- 將不受信任的資料（通常來自 HTTP 請求）包含到動態內容中
- 然後發送給網路用戶，而無需事先驗證惡意內容



Xss 會有以下行為

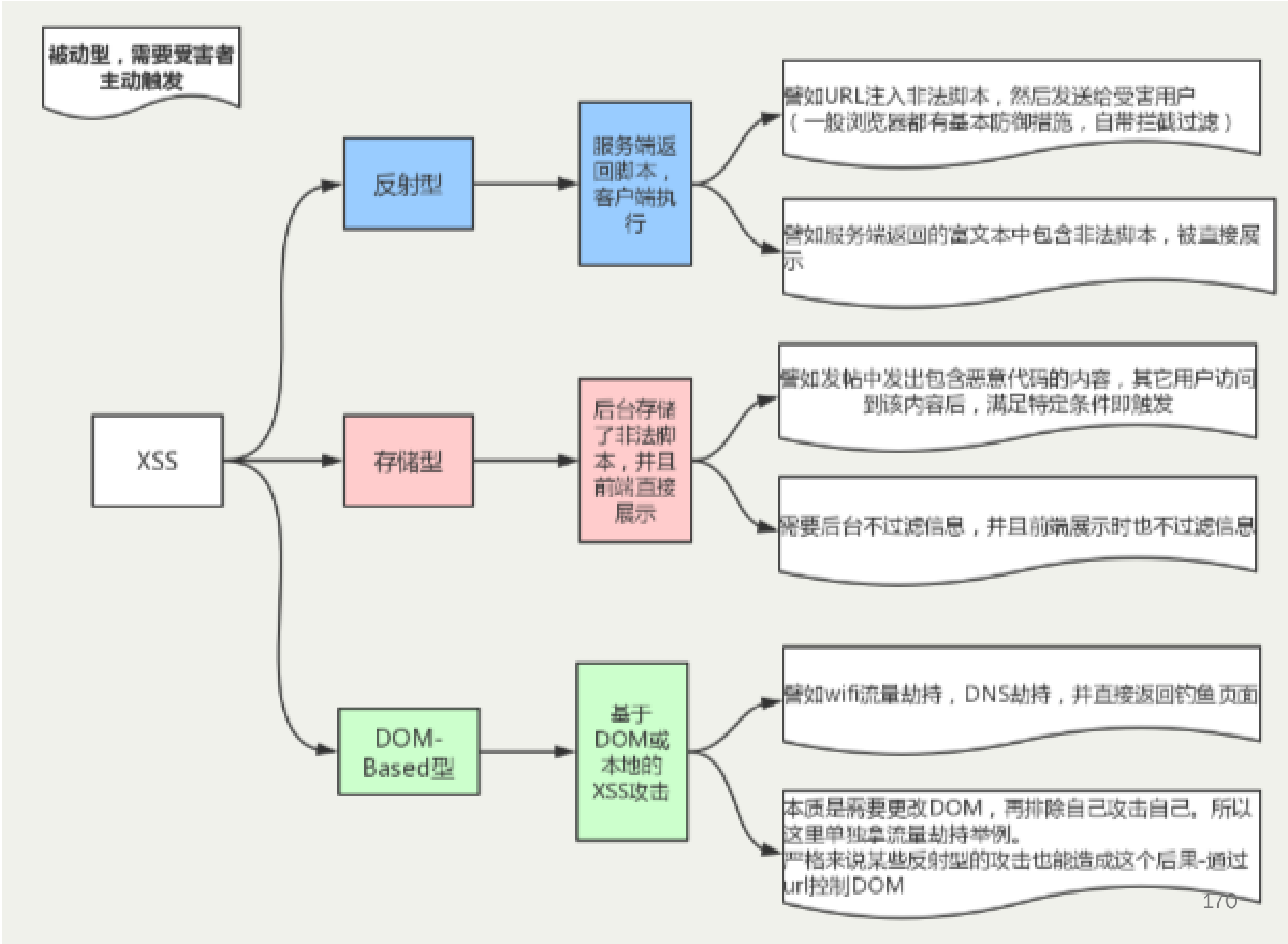
- Steal user's session 竊取使用者會話
- 竊取敏感數據
- 重寫網頁
- 將使用者重新導向到惡意網站

來看例子了解**XSS**過程



再看一個例子

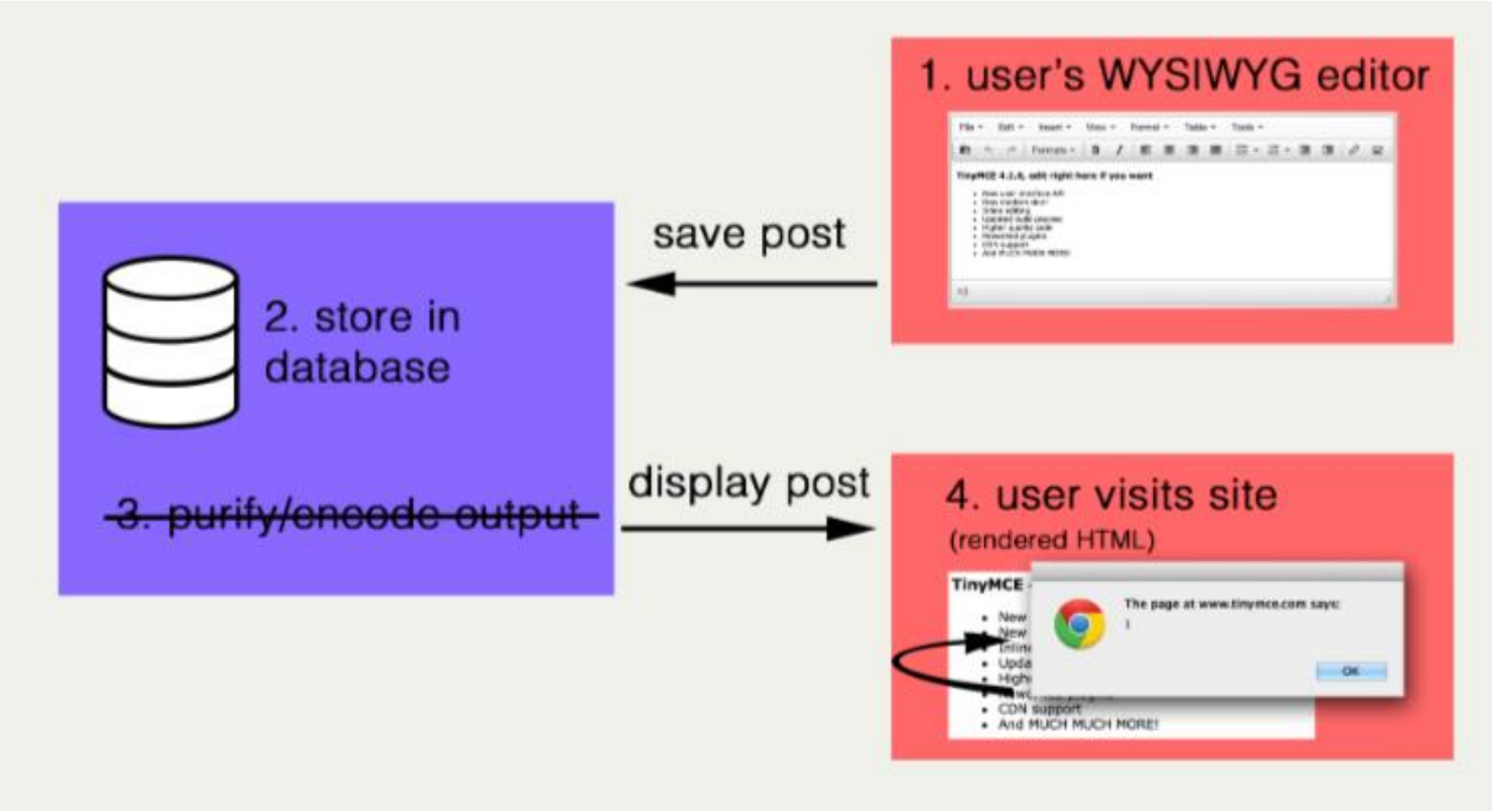
XSS種類


被动型，需要受害者主动触发
XSS
反射型
存储型
DOM-Based型
服务端返回脚本，客户端执行
后台存储了非法脚本，并且前端直接展示
基于DOM或本地的XSS攻击
譬如URL注入非法脚本，然后发送给受害用户（一般浏览器都有基本防御措施，自带拦截过滤）
譬如服务端返回的富文本中包含非法脚本，被直接展示
譬如发帖中发出包含恶意代码的内容，其它用户访问到该内容后，满足特定条件即触发
需要后台不过滤信息，并且前端展示时也不过滤信息
譬如wifi流量劫持，DNS劫持，并直接返回钓鱼页面
本质是需要更改DOM，再排除自己攻击自己。所以这里单独拿流量劫持举例。
严格来说某些反射型的攻击也能造成这个后果-通过url控制DOM

來玩一下吧

- https://xss-game.appspot.com/

xss存在的原因


1. user's WYSIWYG editor
save post
2. store in database
3. purify/encode output
display post
4. user visits site
(rendered HTML)
The page at www.tinymce.com says:
OK

XSS 的防範

> 对输入(和URL参数)进行过滤，对输出进行编码。
> Cookie 设置 http-only

编程语言
防御代码

- nodejs
- php
- java
- ruby
- flask
- spring boot

Content-Security-Policy（CSP）和X-XSS-Protection的定義

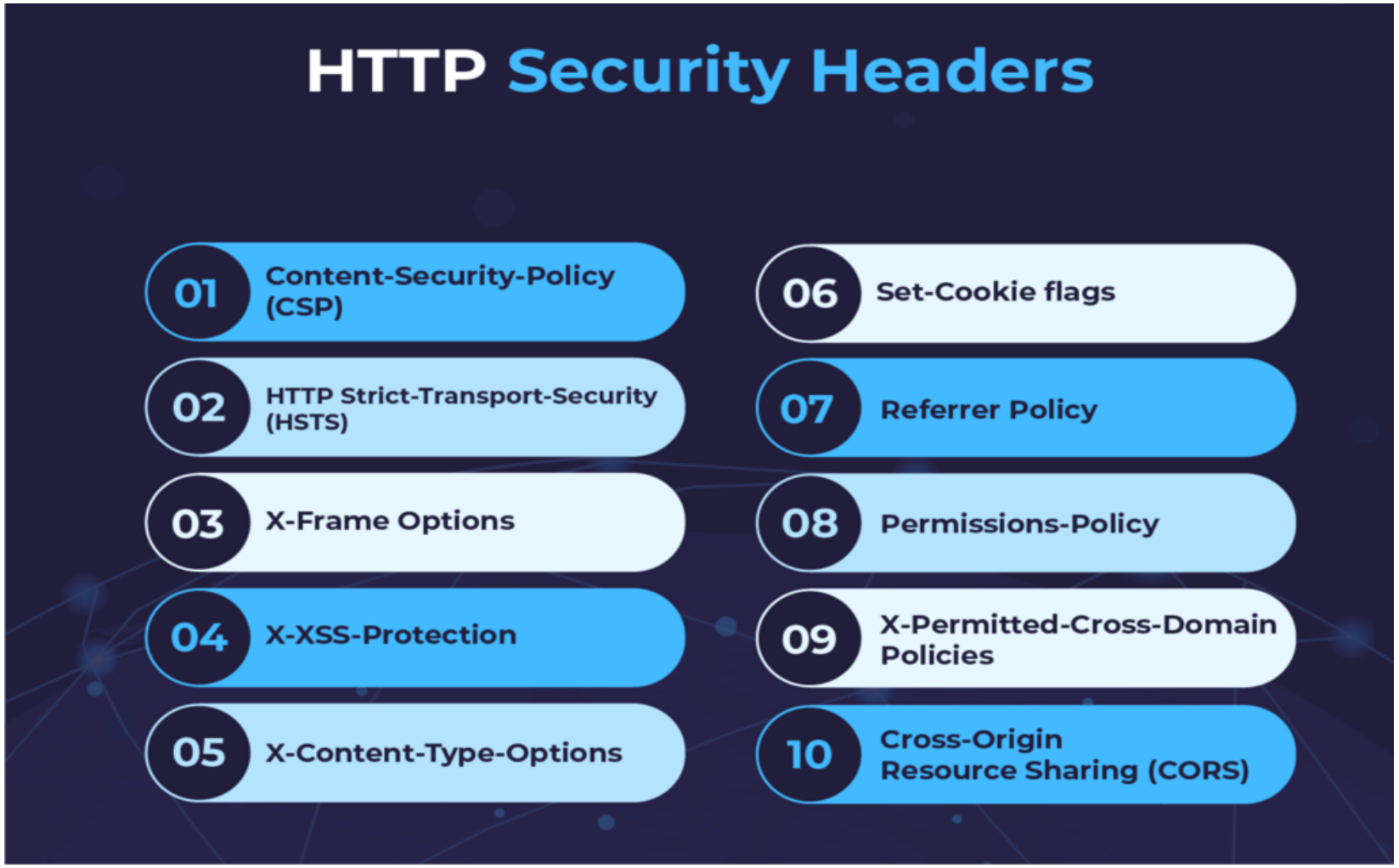
- X-XSS-Protection
  - 主要是針對反射式XSS的預防
- Content Security Policy（CSP）
  - 禁止瀏覽器從非預期的來源，載入惡意指令稿

舉例:php如何達成Content-Security-Policy（CSP）

```php
<?php
header("Content-Security-Policy: default-src 'none'; img-src 'self' data:;");
?>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>test</title>
</head>
<body>
    <p>test</p>
    <img src="example.jpg">
</body>
</html>
```

安全配置

HTTP標頭安全配置


HTTP Security Headers
01 Content-Security-Policy (CSP)
02 HTTP Strict-Transport-Security (HSTS)
03 X-Frame Options
04 X-XSS-Protection
05 X-Content-Type-Options
06 Set-Cookie flags
07 Referrer Policy
08 Permissions-Policy
09 X-Permitted-Cross-Domain Policies
10 Cross-Origin Resource Sharing (CORS)

Apache Web Server安全配置

- **禁用不必要的模塊**
- 編輯httpd.conf文件，註釋掉不需要的模塊：
- #LoadModule autoindex_module modules/mod_autoindex.so
  #LoadModule status_module modules/mod_status.so

禁用危險函數

- 禁用不必要的PHP函數
- 編輯php.ini文件，禁用危險函數：
  - disable_functions = exec,passthru,shell_exec,system

錯誤報告設置

- 在生產環境中禁用錯誤顯示。
- 編輯php.ini文件，設置錯誤報告：
  - display_errors = Off log_errors = On error_log = "C:/xampp/php/logs/php_errors.log"

使用Prepared Statements

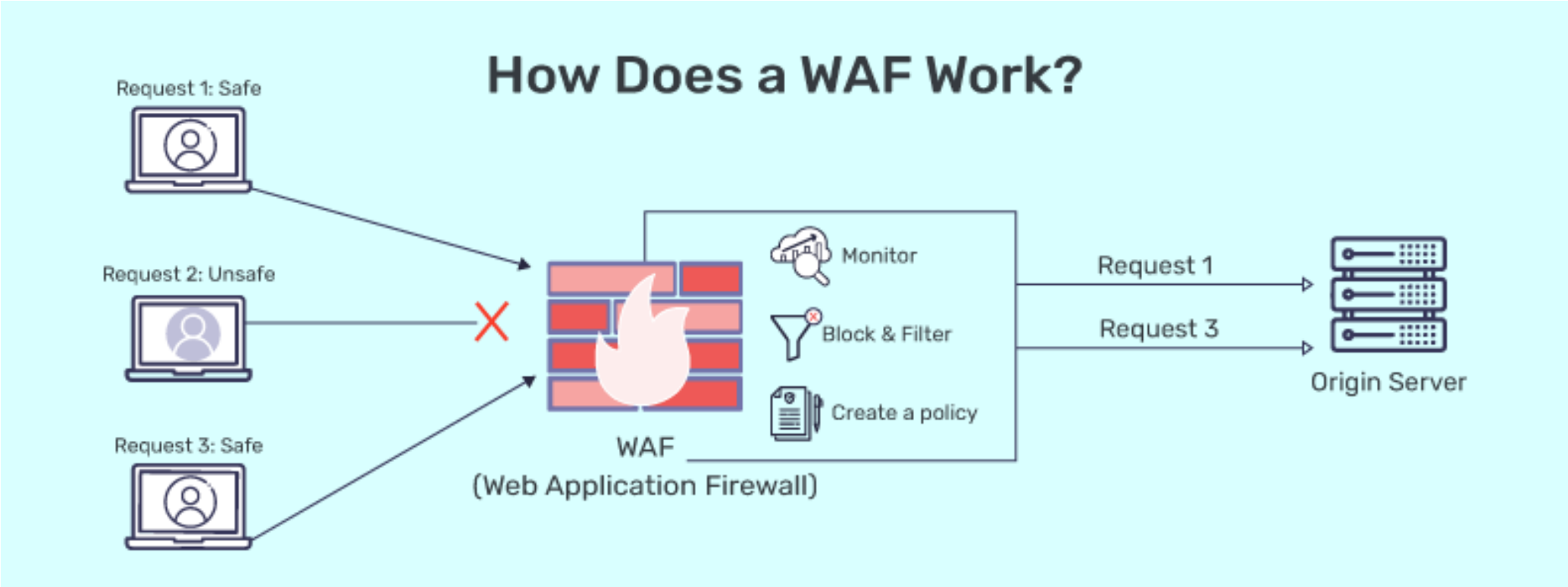
- 防止SQL注入攻擊。

```
<?php
$stmt = $conn->prepare("SELECT * FROM users WHERE username=? AND password=?"); $stmt->bind_param("ss", $username, $password); $stmt->execute(); $result = $stmt->get_result();
?>
```

保護機制

WAF

- 使用Web應用防火牆（ WAF ）



Waf如何過濾流量

- 積極的安全模型 – 它涉及明確定義允許的模式和行為，僅允許已知的合法流量並拒絕所有其他模式，透過白名單方法增強安全性。
- 負安全模型 - 它識別並阻止已知的惡意模式或簽名，假設任何與預定義攻擊模式匹配的流量都是惡意的並拒絕訪問，透過黑名單方法提供安全性。
- 進階功能 - 利用機器學習/人工智慧演算法和威脅情報來主動防禦複雜的威脅。

WAF部署方式

- 內聯或橋接模式 – WAF 可以內嵌部署在 Web 應用程式和網路之間。在橋接模式下，它監視流量而不直接攔截流量，使其成為被動觀察者。
- 基於雲端的 WAF – 基於雲端的 WAF 解決方案由第三方雲端服務供應商託管和管理。它們為 Web 應用程式提供可擴展且靈活的安全性，而無需本地硬體。這對於雲端託管應用程式特別有用。
- 基於設備的 WAF – WAF 設備是安裝在網路基礎架構內部的實體設備。它們為具有特定硬體需求或合規性需求的組織提供專門的本地化保護。