CSE480: Web Security

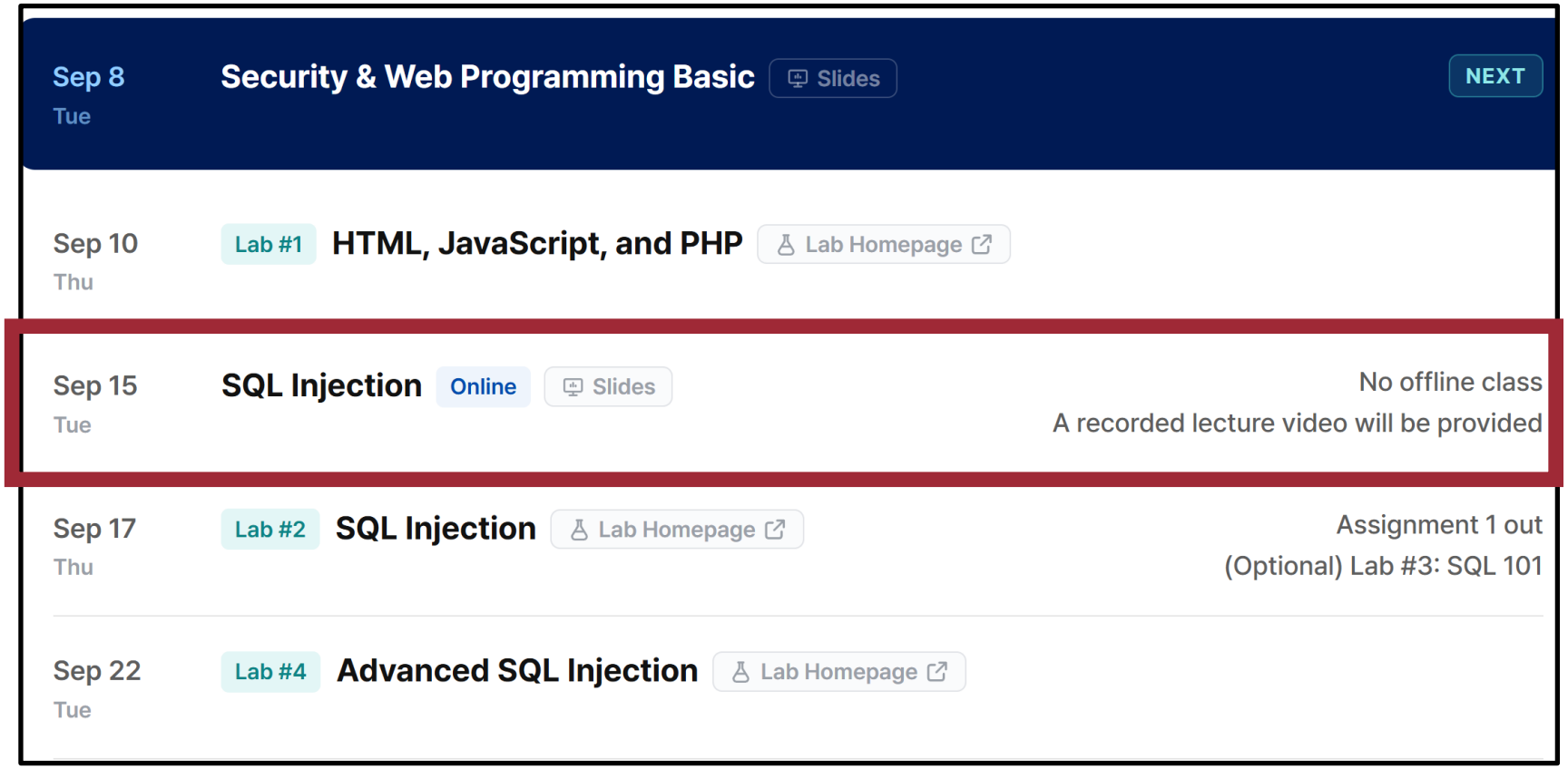
2. Security & Web Programming Basics

Seongil Wi

Department of Computer Science and Engineering

Class on 9/15: Online Class

- There will be **NO** offline class on September 15
- Instead, a recorded lecture video will be provided (at Blackboard)
  - This video will be available **ONLY** on September 15



Class on 9/17 & 9/22: Lab Sessions

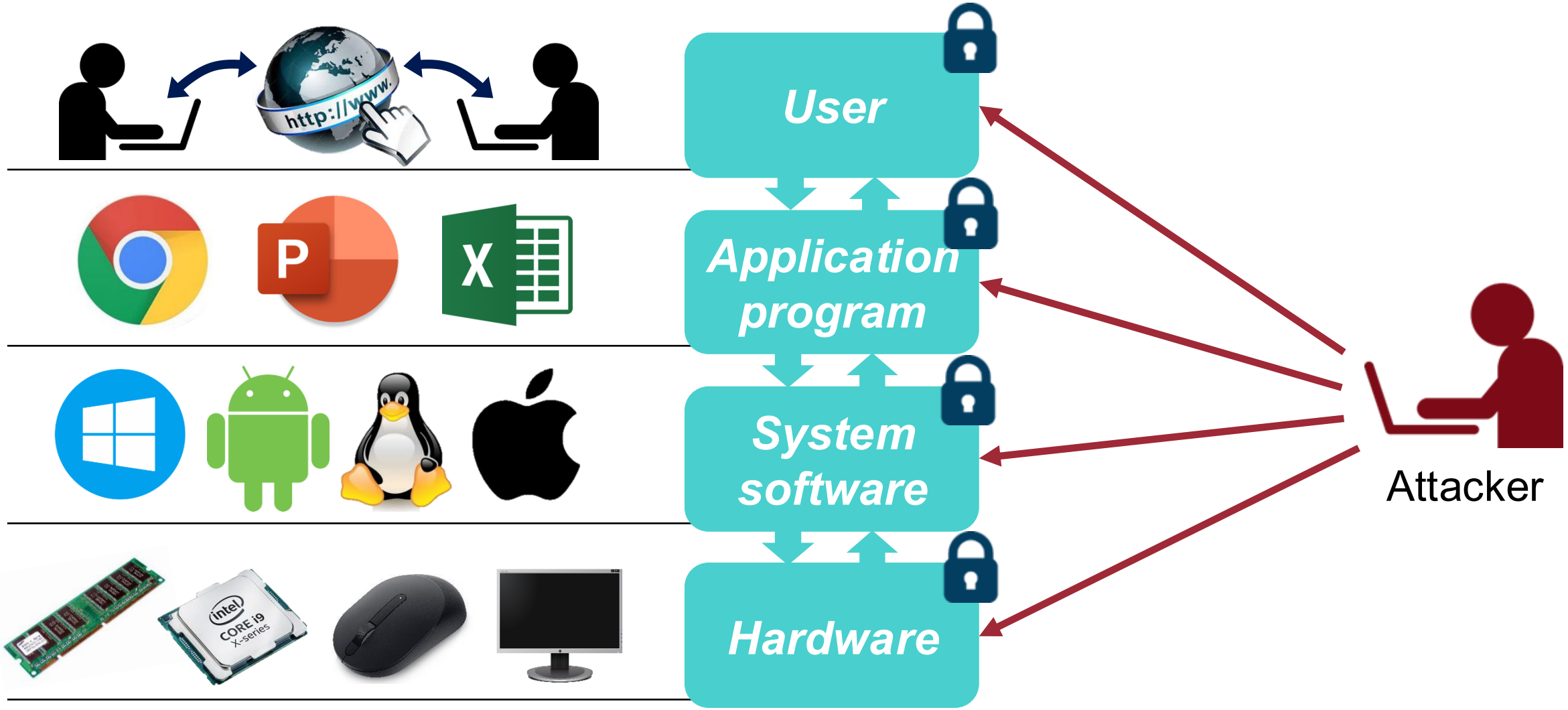
- There will be **two consecutive lab sessions** on SQL Injection on September 17 (Thu.) and September 22 (Tue.)

Sep 8 Tue	**Security & Web Programming Basic** Slides	NEXT
Sep 10 Thu	Lab #1 **HTML, JavaScript, and PHP** Lab Homepage	
Sep 15 Tue	**SQL Injection** Online Slides	No offline class A recorded lecture video will be provided
Sep 17 Thu	Lab #2 **SQL Injection** Lab Homepage	Assignment 1 out (Optional) Lab #3: SQL 101
Sep 22 Tue	Lab #4 **Advanced SQL Injection** Lab Homepage	

Security Basics

Computer Security

The protection of **computer systems** from unauthorized access



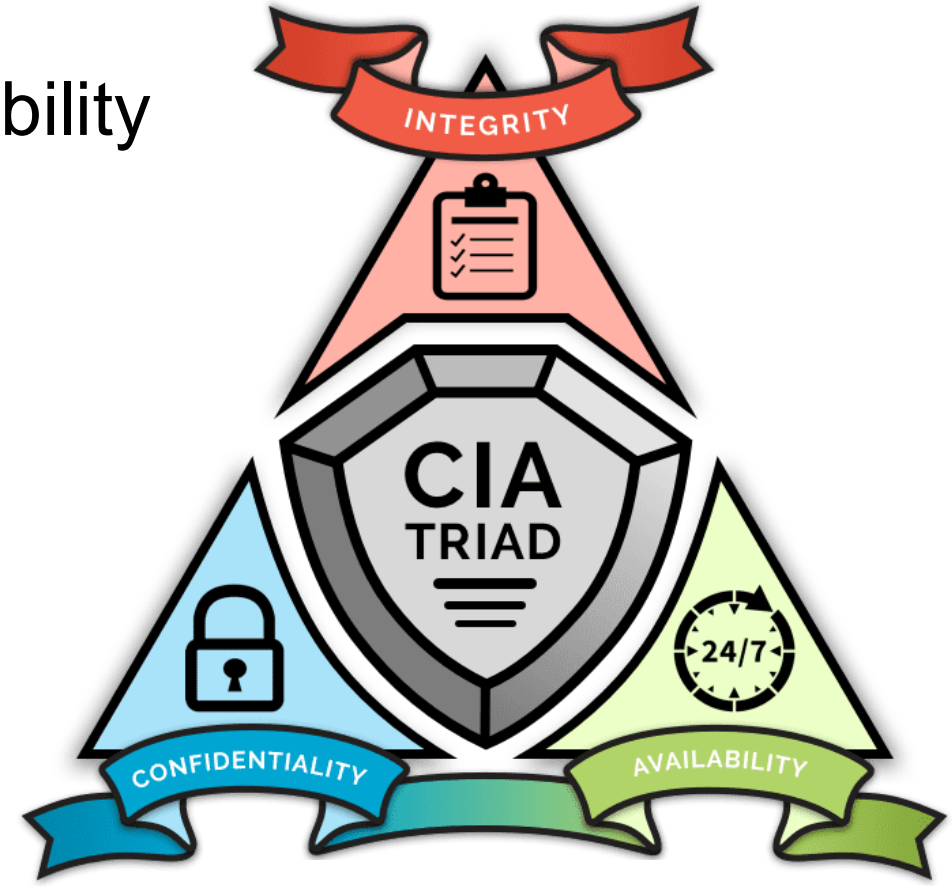
Q. Is Your Computer Secure?

Under what conditions can you say your computer is secure?

Secure Systems Satisfy the CIA Properties

- Three most important **properties** of computer security

- **CIA**: **C**onfidentiality, **I**ntegrity, and **A**vailability



CIA

- **C**onfidentiality
- **I**ntegrity
- **A**vailability

CIA (1): Confidentiality (기밀성)

- **C**onfidentiality: information is not made available to unauthorized parties

- **I**ntegrity

- **A**vailability

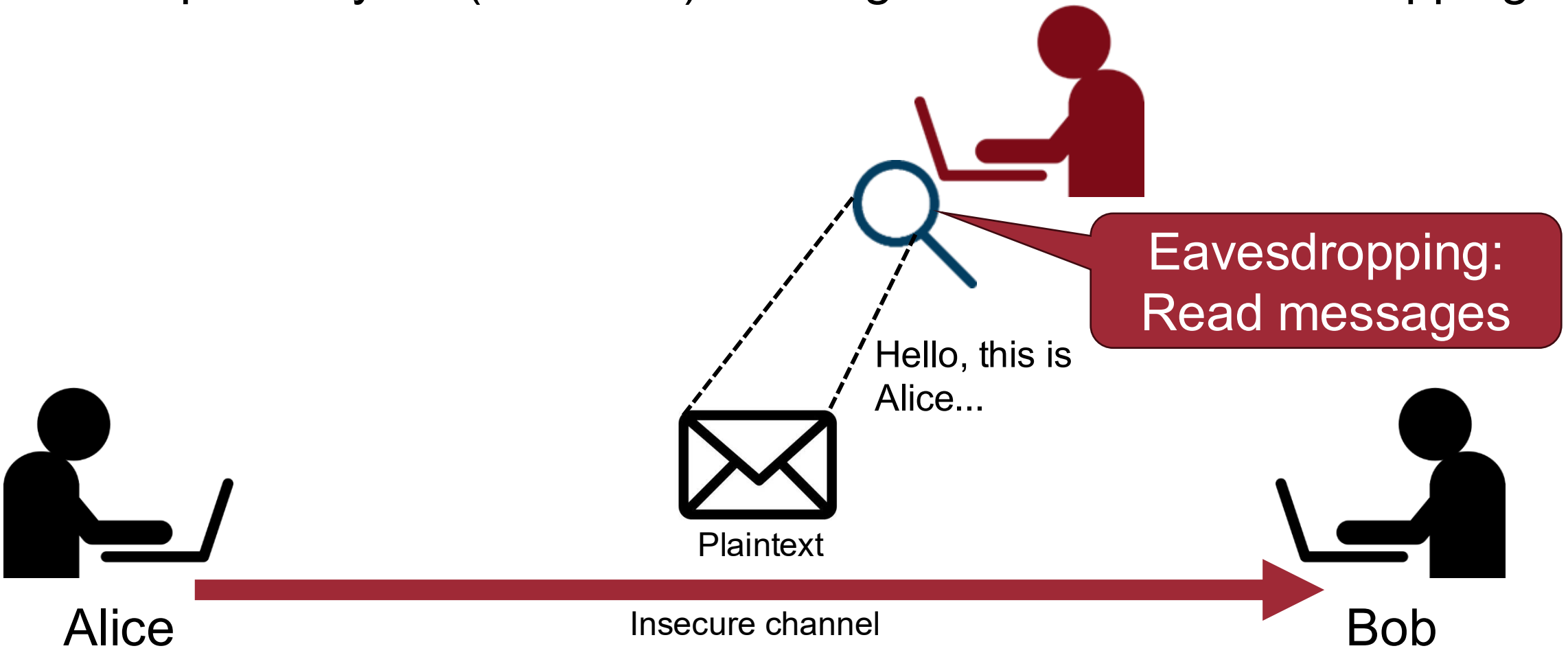
CIA (1): Confidentiality

- Information is not made available to unauthorized parties
- Avoidance of the unauthorized disclosure of information
  - Protection of data
  - Provide access for those who are allowed to see the data
  - Disallow others from learning anything about the data

- How to achieve confidentiality?
  - **Cryptography (암호화, 암호학)**: transformation of information
  - **Access control (접근제어):** gatekeeper

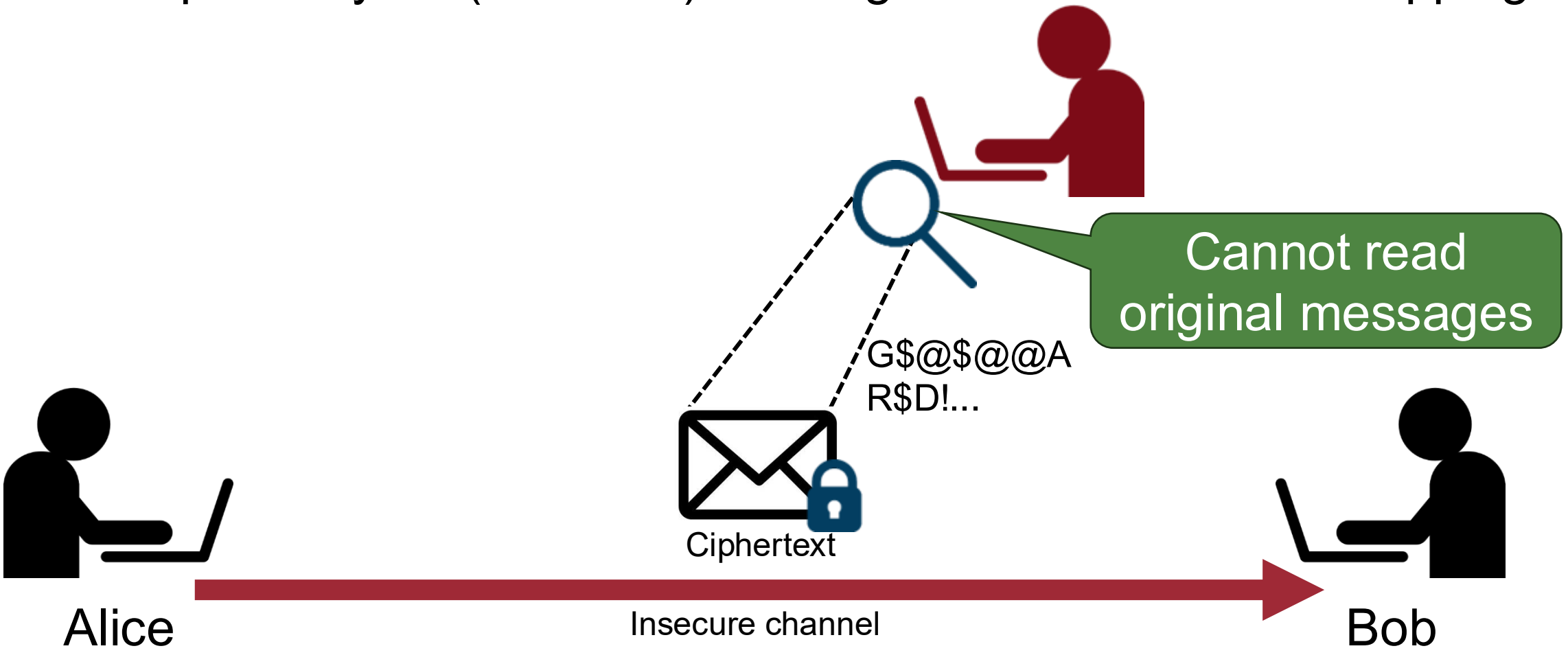
CIA (1): Confidentiality – Cryptography

- "Secret writing" in Greek
- Goal: protect your (sensitive) messages/data from eavesdropping

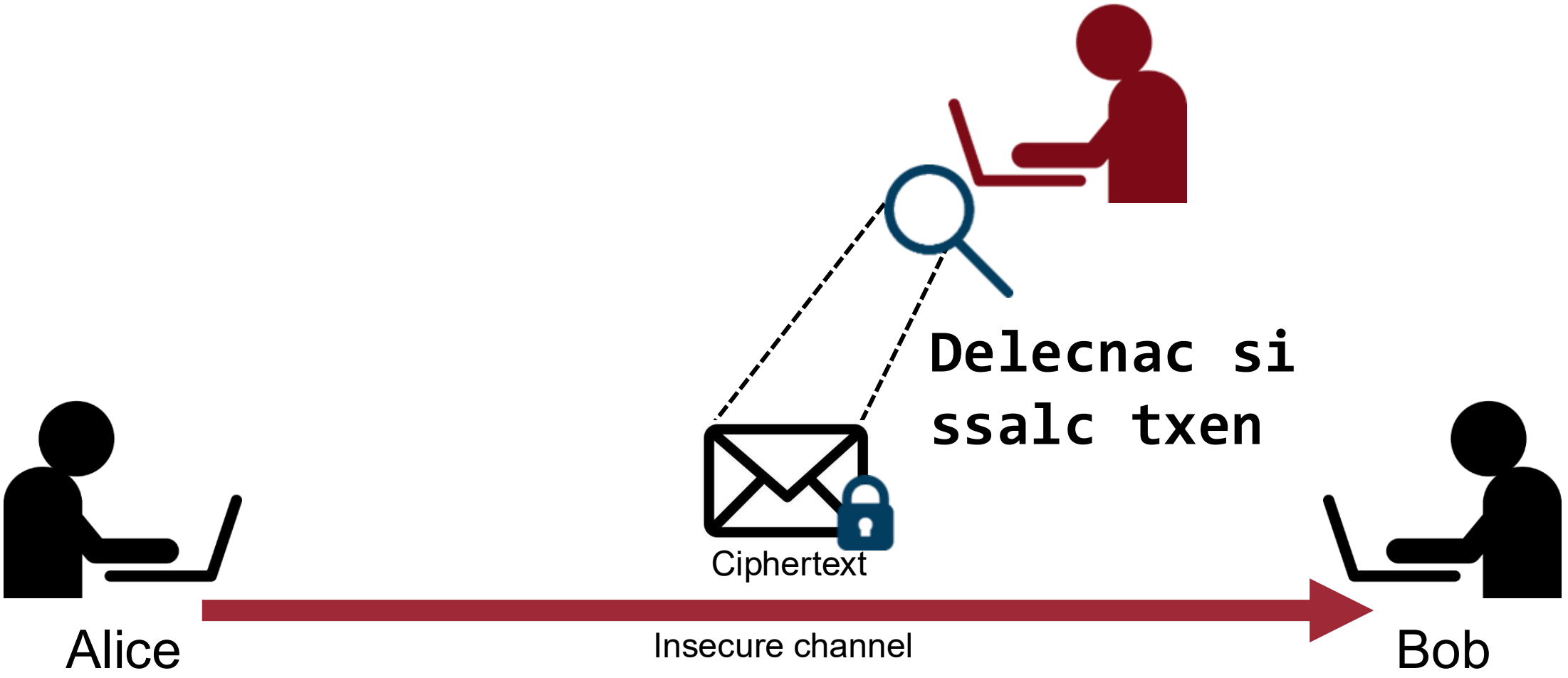


CIA (1): Confidentiality – Cryptography

- "Secret writing" in Greek
- Goal: protect your (sensitive) messages/data from eavesdropping

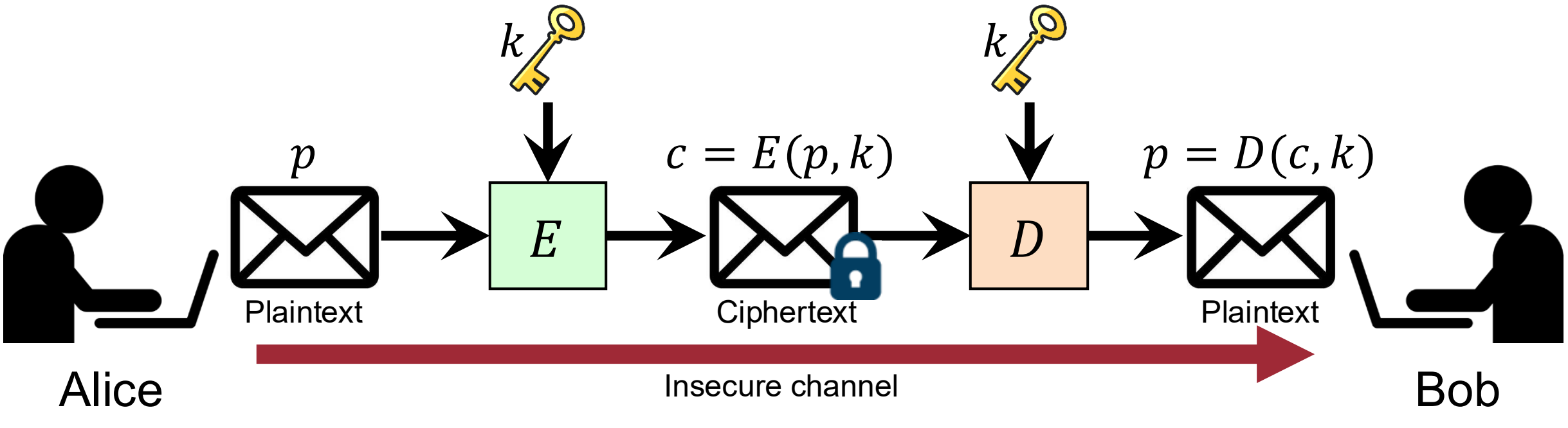


Quiz: Can you Decrypt this One?



CIA (1): Confidentiality – Cryptography

- To be secure: make it **extremely difficult** to decrypt the data without the decryption key
- Two functions: encryption (E) and decryption (D) parameterized by a plaintext (p), ciphertext (c), and cryptographic key (k)

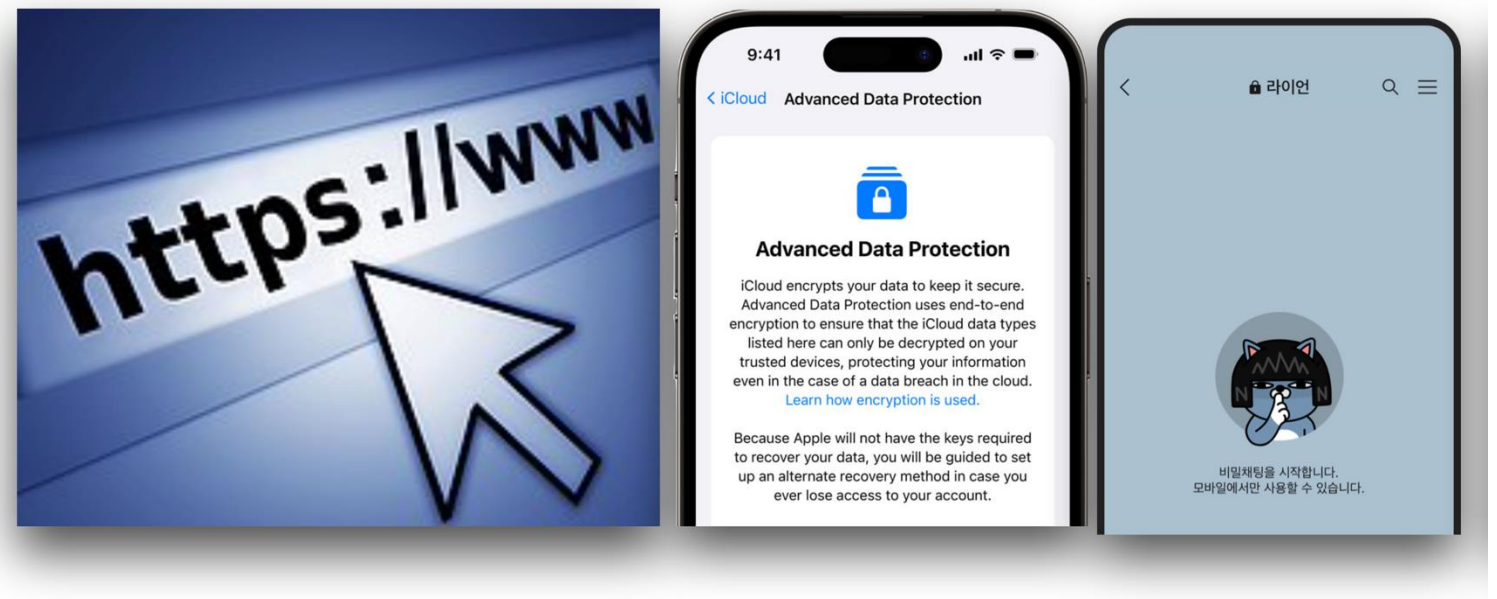


CIA (1): Confidentiality – Cryptography

- Transformation of information using an encryption key
- Only be read by another user who has the decryption key

- Schemes: symmetric-key encryption, public-key encryption, etc

- Example:



CIA (1): Confidentiality – Access Control

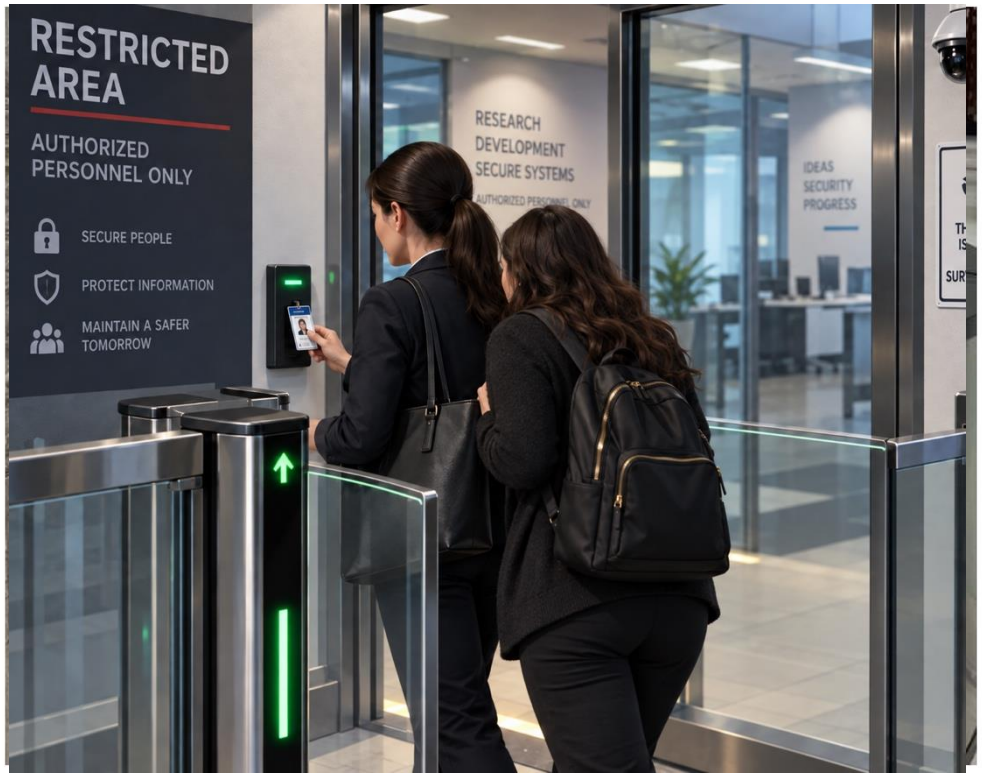
- ***Rules*** and ***policies*** that limit access to confidential information
- Determine what users have permission to do
- Permission is determined by identity (e.g., name, serial) or role (e.g., professor, TA, student)
- Example: Linux file system

	/etc/passwd	/usr/bin	/home/prof/exam_problem/
root	rw	rwx	rwx
professor	r	rx	rwx
ta	r	rx	r
student1	r	rx	-
student2	r	rx	-

Students 1 and 2 are unable to read the exam problem!

CIA (1): Confidentiality – Access Control

Access Control Failure



[헤럴드경제=박로명 기자] "입주민이 카드를 찍으면 뒤에 외부인이 붙어서 커뮤니티 시설에 입장하는 장면을 자주 목격한다. 가구 수가 많은 대단지다보니 외부인이 입주민 카드를 남용해 커뮤니티 시설을 사용하는 일이 빈번한데, 커뮤니티 시설에선 주민만 만나고 싶다"(강남구 개포동 아파트 주민 A씨)

An unauthorized person enters a restricted facility by tailgating an authorized person

CIA (1): Confidentiality

Exercise: Internet Banking

- What mechanism is used to achieve confidentiality?
  - Visit the bank website and login
  - ID and PW are sent to the server by your web browser using HTTPS
  - The server allows you to access only your account

CIA (2): Integrity (무결성)

- **C**onfidentiality: information is not made available to unauthorized parties

- **I**ntegrity

- **A**vailability

CIA (2): Integrity (무결성)

- **C**onfidentiality: information is not made available to unauthorized parties
- **I**ntegrity: information is not modified in an unauthorized manner
- **A**vailability

CIA (2): Integrity

Information has not been altered in an unauthorized way

- **Benign compromise**: information altered by accident
  - E.g., bit flips in memory due to cosmic ray

CIA (2): Integrity – Benign Compromise

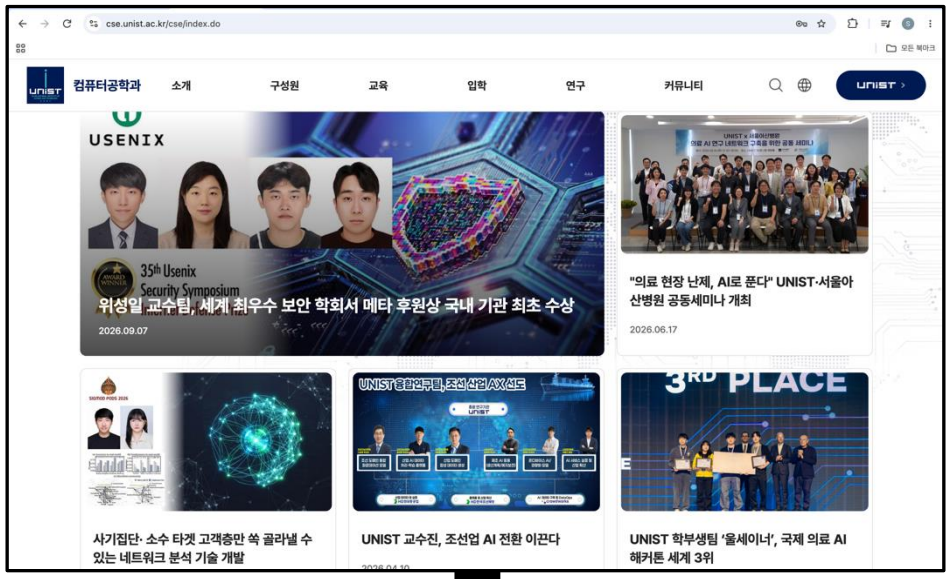
CIA (2): Integrity

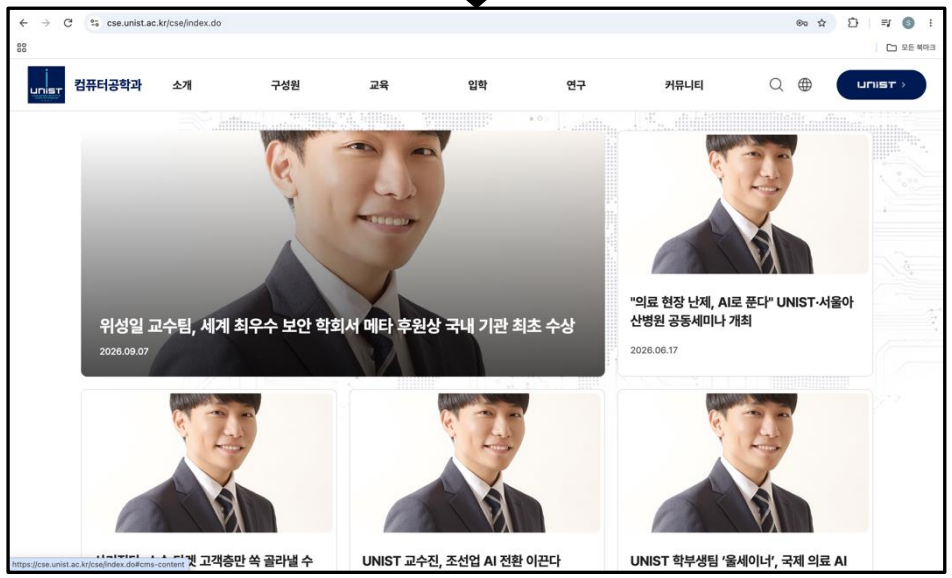
Information has not been altered in an unauthorized way

- **Benign compromise**: information altered by accident
  - E.g., bit flips in memory due to cosmic ray

- **Malicious compromise**: information altered by attackers
  - E.g., malicious code that changes some files in a system

CIA (2): Integrity – Malicious Compromise





수면위로 떠오른 UNIST(유니스트) '해킹 사건'

입력 2017-04-29 09:28

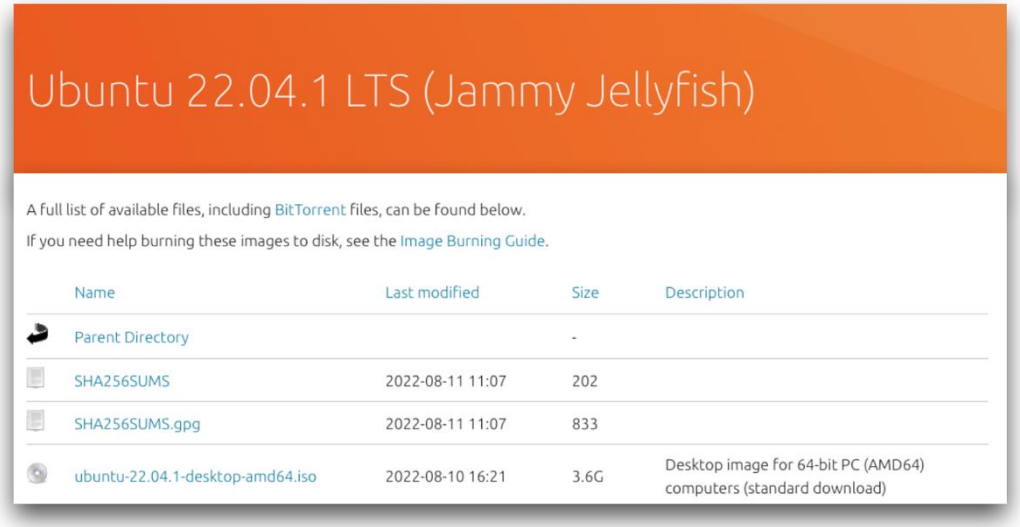
> **'룸메이트 바꾸려고' 전산망 뚫어…2013년부터 총 4건**
> **유니스트 보안 정보 둔감 '지적'**

지난 2014년 3월 유니스트 해킹동아리가 학교전산망에 침입, 자신들의 해커 모집광고를 종합전산망 홈피에 게재해 놓은 배너 모습.

CIA (2): Integrity

Ensuring Integrity

- How to ensure the integrity of computer systems?
- **Backups**: periodic archiving of data
- **Checksums**: computation of a function that maps the data to a numerical value



CIA (3): Availability (가용성)

- **C**onfidentiality: information is not made available to unauthorized parties
- **I**ntegrity: information is not modified in an unauthorized manner
- **A**vailability

CIA (3): Availability (가용성)

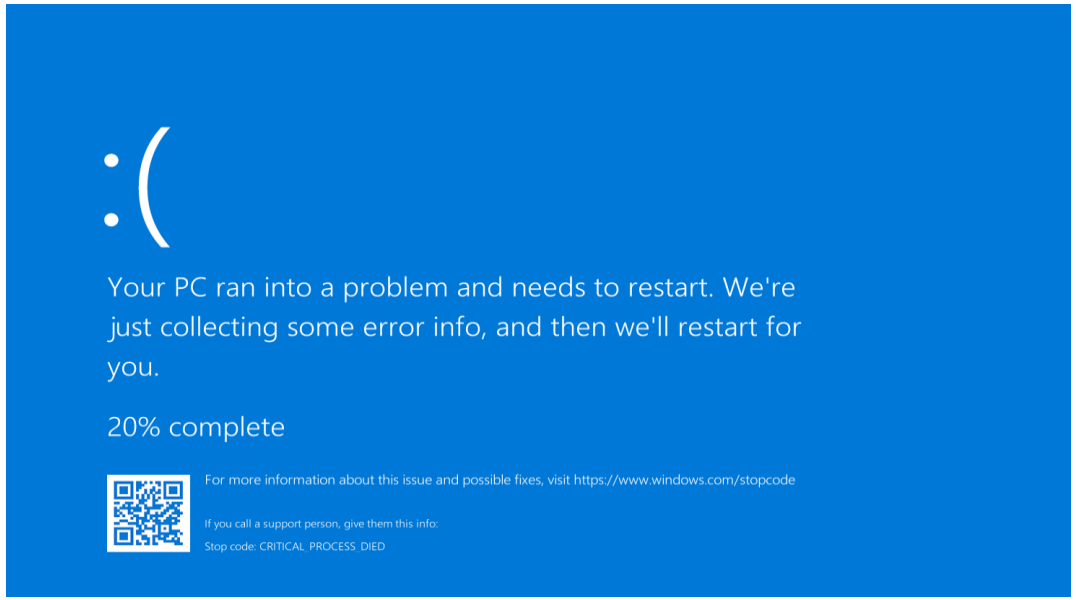
- **C**onfidentiality: information is not made available to unauthorized parties
- **I**ntegrity: information is not modified in an unauthorized manner
- **A**vailability: information should be readily available when it is needed

CIA (3): Availability

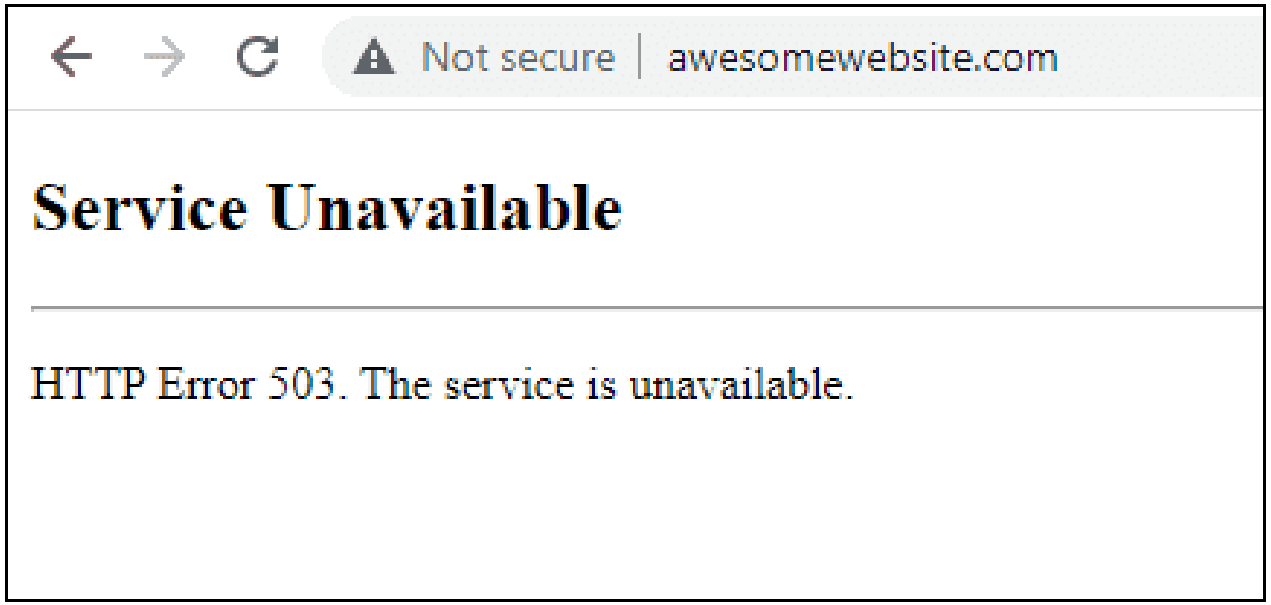
- Information should be ***accessible*** and ***modifiable*** in a timely fashion
- Imagine a unbreakable and unopenable vault. Is it useful?

CIA (3): Availability

- Information should be ***accessible*** and ***modifiable*** in a timely fashion
- Imagine a unbreakable and unopenable vault. Is it useful?



Blue Screen of Death



503 Error

CIA (3): Availability

- Information should be ***accessible*** and ***modifiable*** in a timely fashion
- Imagine a unbreakable and unopenable vault. Is it useful?

Kakao's meltdown raises big questions about its management

"President office said KaKao's network disturbance could even be **a threat to national security**"

CIA (3): Availability

- Information should be ***accessible*** and ***modifiable*** in a timely fashion
- Imagine a unbreakable and unopenable vault. Is it useful?

- How to achieve availability?
  - **Physical protections**: keep information available even in physical challenges (e.g., storms, earthquakes, or power outages)
  - **Computational redundancies**: computers that serve as fallbacks in the case of failure

Other properties?

- **C**onfidentiality
- **I**ntegrity
- **A**vailability

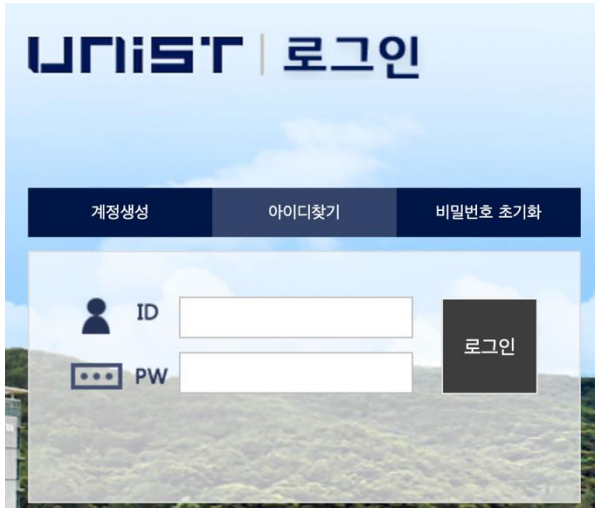
Other properties?

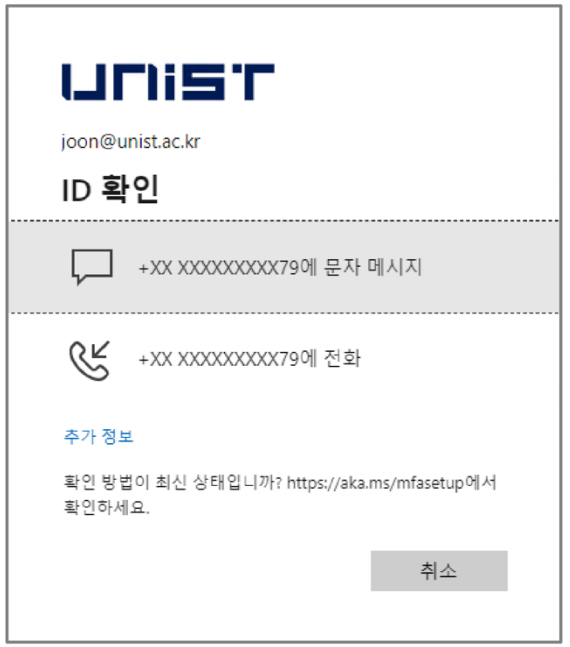
- **C**onfidentiality
- **I**ntegrity
- **A**vailability

+ **Authentication**: the ability of a computer system to *confirm the sender's identity*

Authentication (인증)

- Determination of the **identity** or **role**
- Typical method
  - Something you are (Fingerprint, iris pattern, …)
  - Something you know (Password, PIN, …)
  - Something you have (Smart card, key, …)





Other properties?

- **C**onfidentiality
- **I**ntegrity
- **A**vailability

+ **Authentication**: the ability of a computer system to *confirm the sender's identity*

+ **Non-repudiation**: the ability of a computer system to *confirm that the sender can not deny about something sent*

Non-repudiation (부인방지)

- What happens if the suspected attacker says that…
  - "I did not make commitment. Maybe someone pretended to be me! (오리발 내밀기, repudiation)"

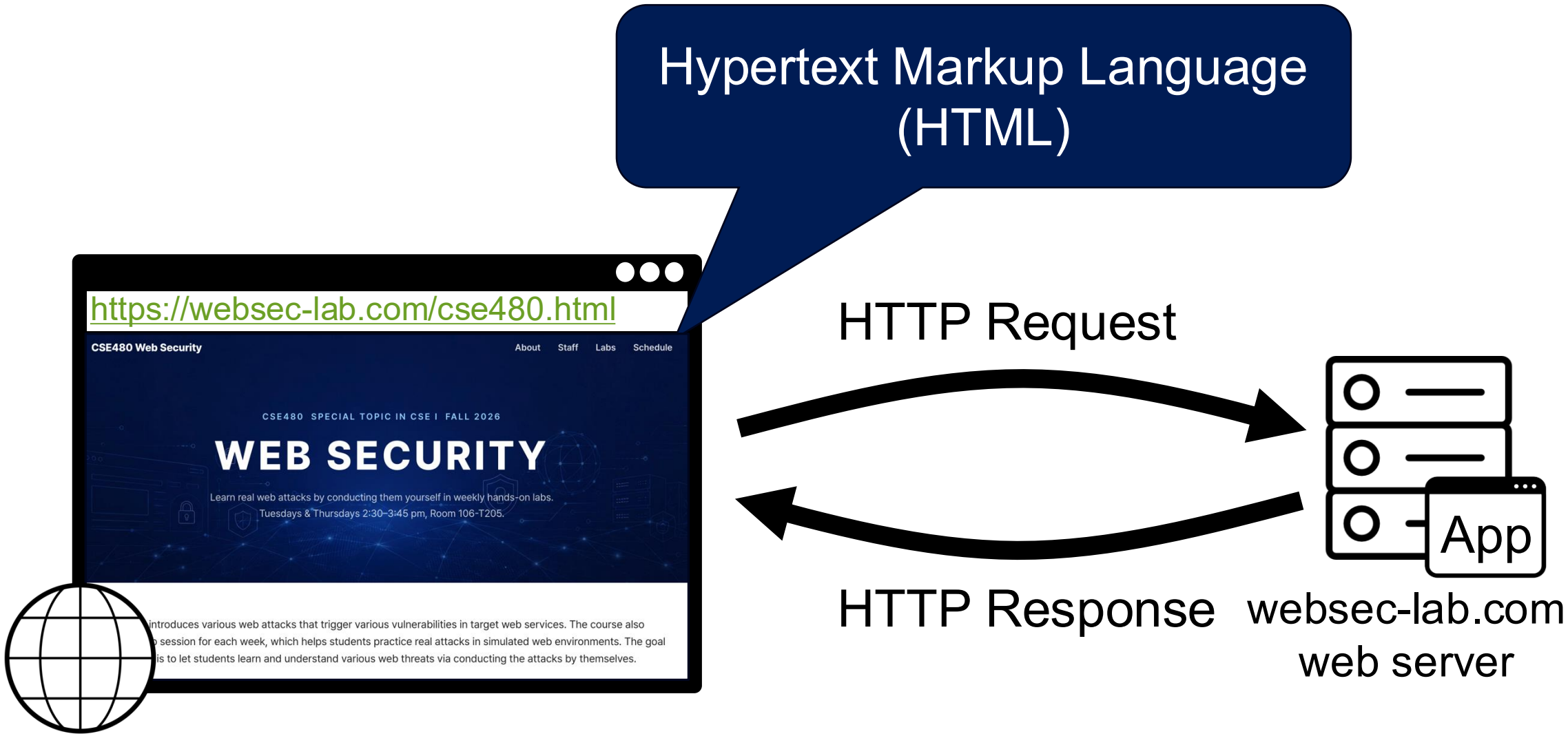
AI 개요

'오리발을 내민다'는 잘못을 저지르고도 엉뚱한 변명을 하며 모른 척 딱 잡아떼는 태도를 뜻하는 속담입니다. 영남일보 +1

- How to determine that statements, policies, and permissions are genuine?
- **Non-repudiation** by secure authentication: authentic statement cannot be denied
  - E.g., digital signature

Web Programming Basic

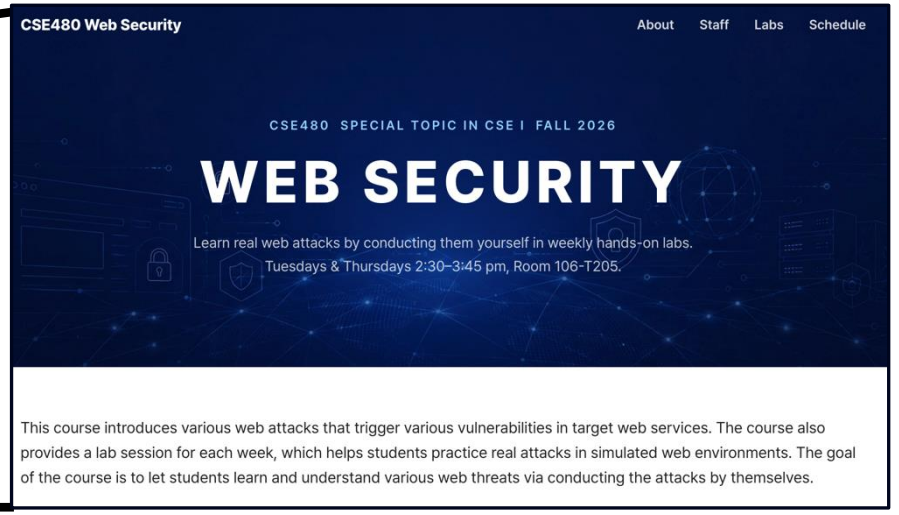
Web Infrastructure



Hypertext Markup Language (HTML)

- Markup language for web page layout
  - NOT programming language (i.e., for computation)!
- A web page (document) is written in HTML using markup tags
  - E.g., `<p>`, `<img>`

```
<html>
  <body>
    ...
  </body>
</html>
```

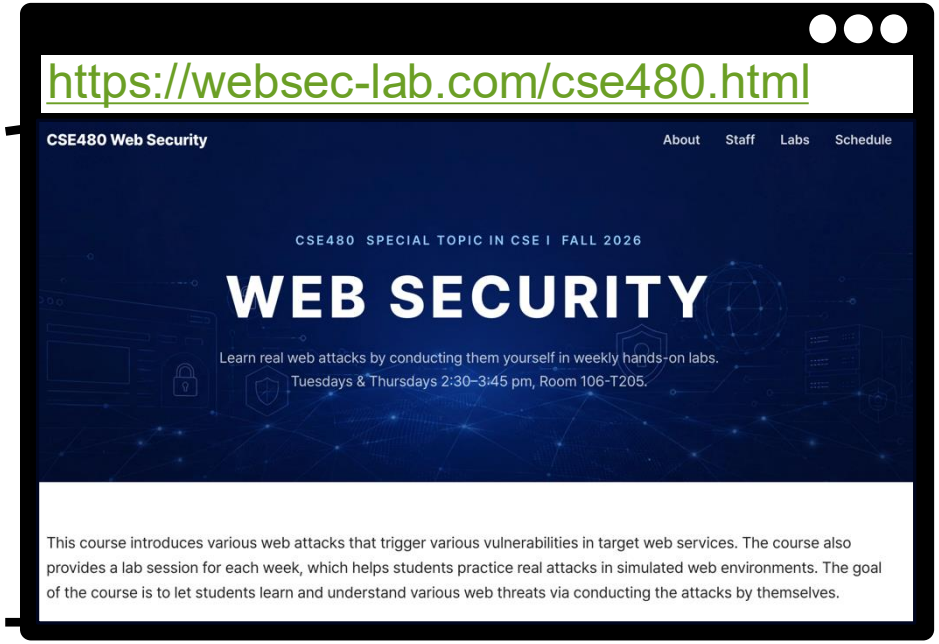


Hypertext Markup Language (HTML)

- Markup language for web page layout
  - NOT programming language (i.e., for computation)!
- A web page (document) is written in HTML using markup tags
  - E.g., `<p>`, `<img>`
- A **browser** interprets a web page when rendering the page
- Describes a hyper-text document
  - E.g., image, audio, video

What if we need computation?
⇒ JavaScript!

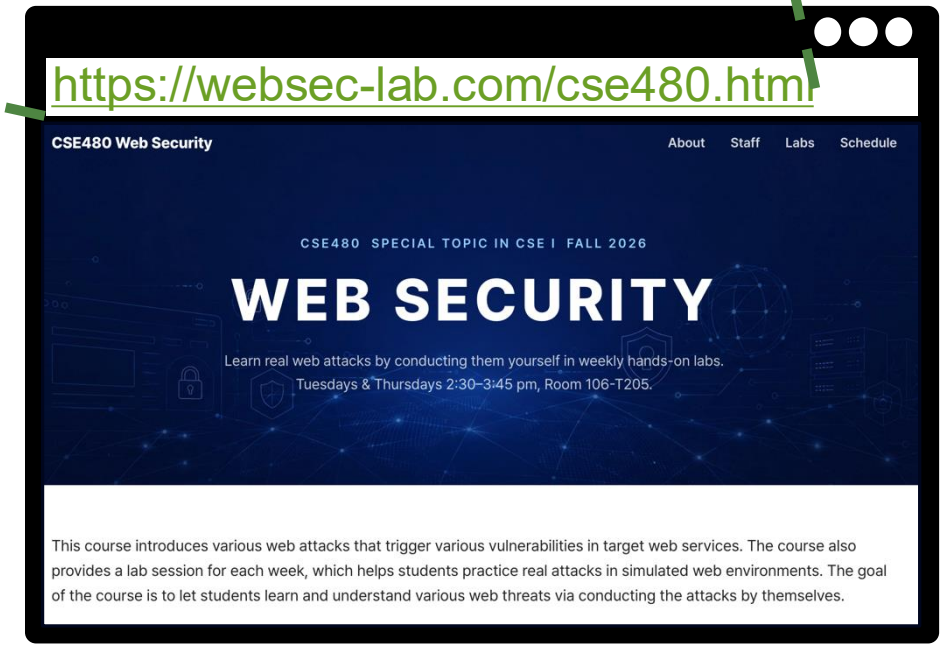
```
<html>
  <body>
    ...
  </body>
</html>
```



Uniform Resource Locators (URLs)

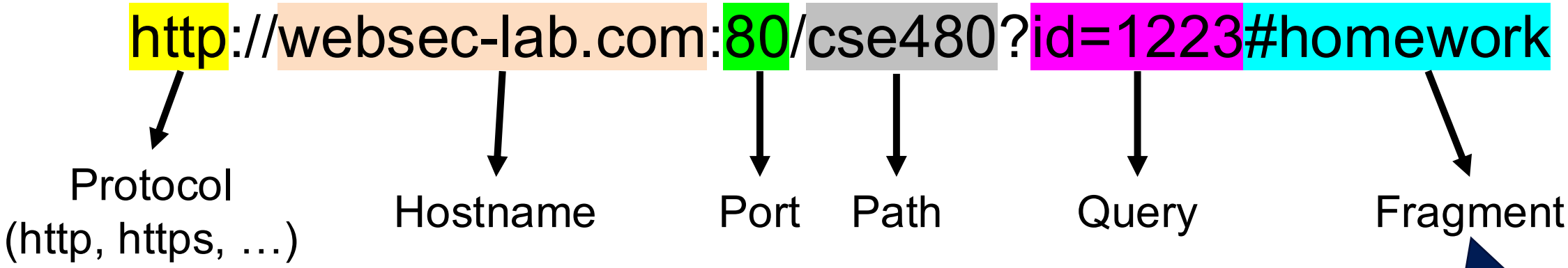
- Global identifiers of network-retrievable documents
- Example

http://websec-lab.com:80/cse480?id=1223#homework



Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

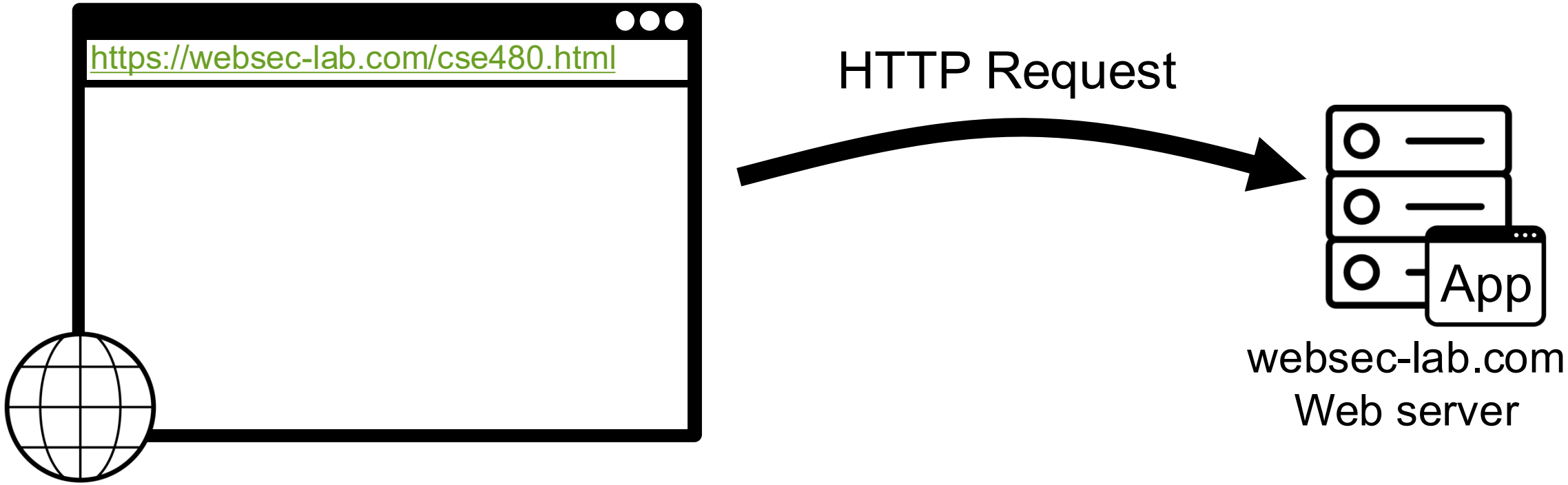


- Special characters are encoded as hex:
  - New line → %0A
  - Space → %20
  - + → %2B

Fragments are not sent to the server

Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers

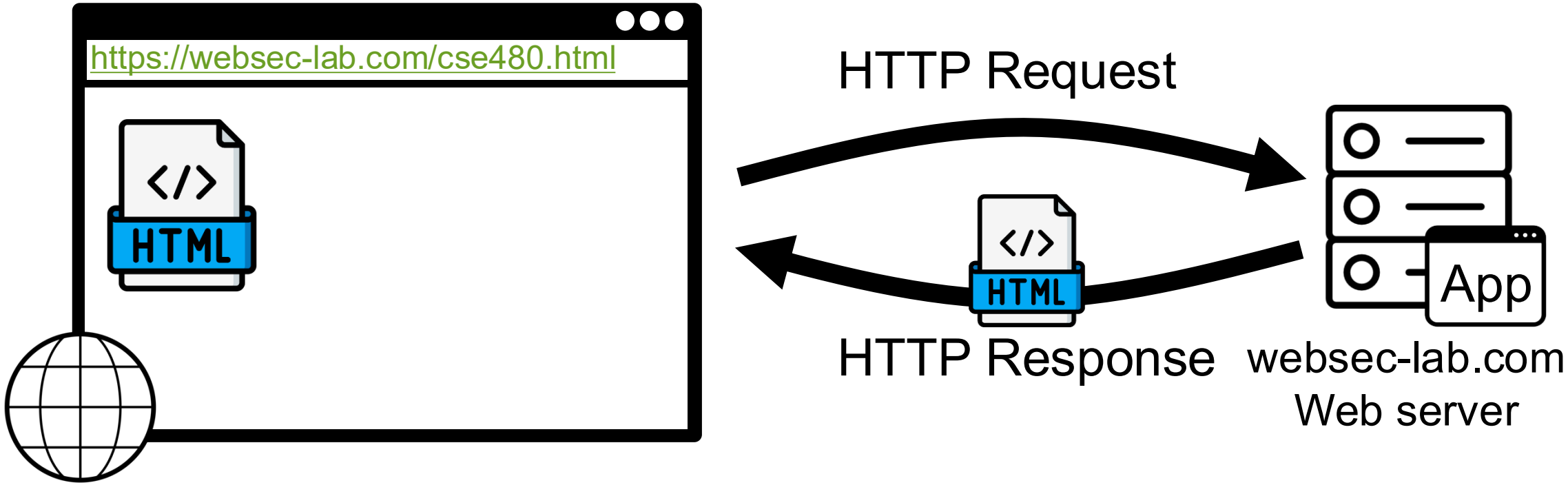


Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers

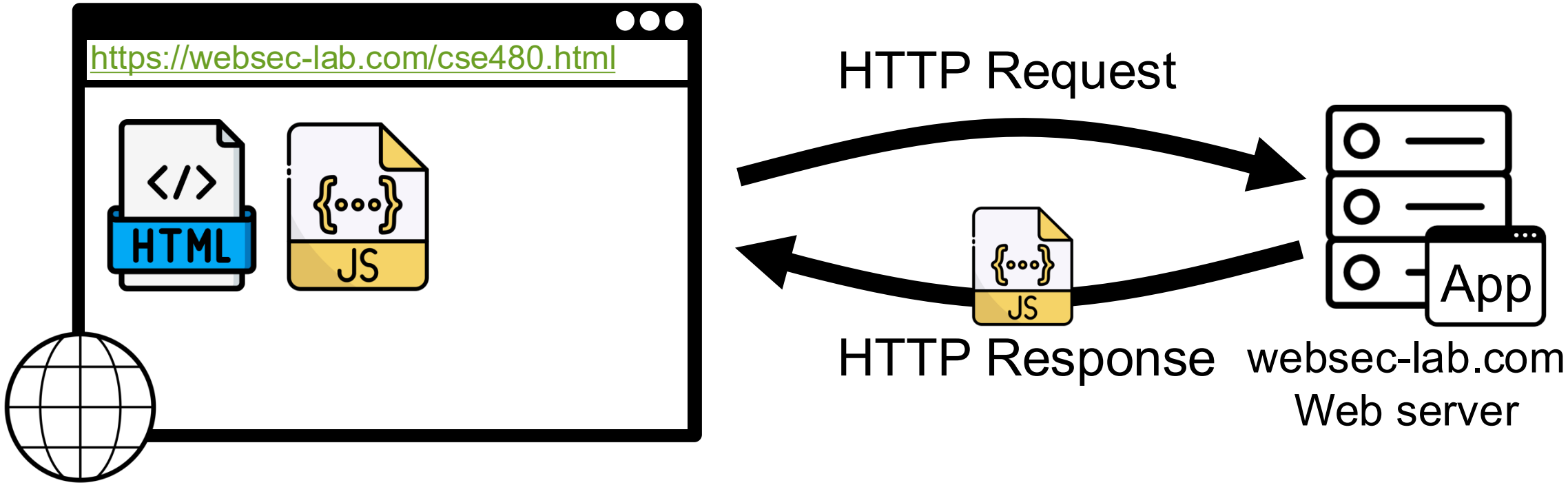
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



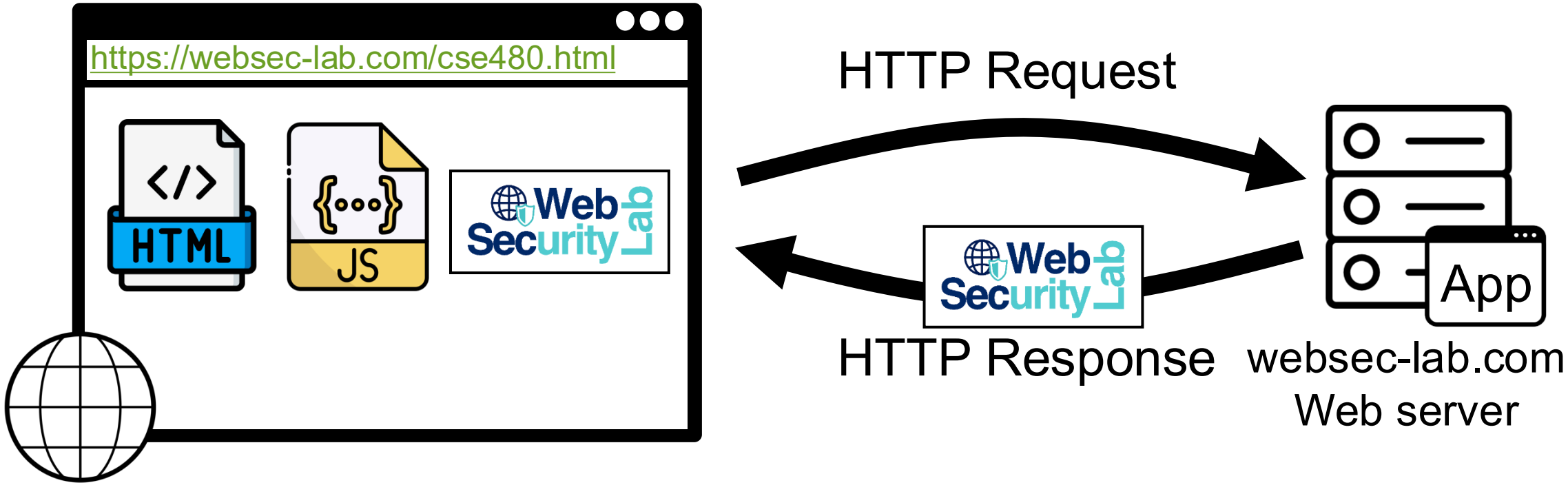
Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers



Hyper Text Transfer Protocol (HTTP)

- The primary protocol for data transfer between web browsers and servers

- Application layer protocol
  - A request is sent over a TCP connection on port:80

- Stateless request/response protocol
  - Each request is independent to previous requests

HTTP Request


```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

HTTP Request

Method | File path | Protocol

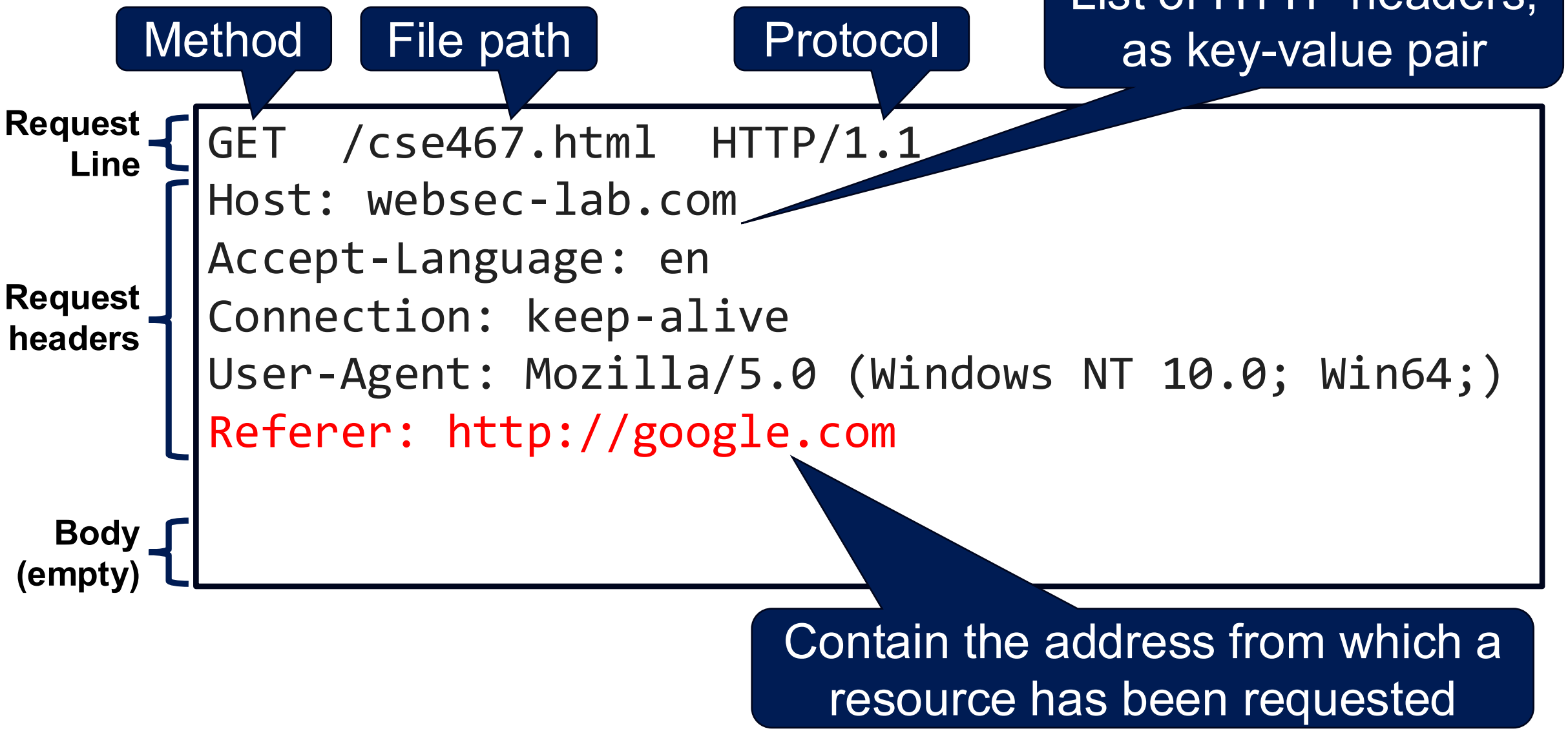
Request Line

```
GET  /cse467.html  HTTP/1.1
Host: websec-lab.com
Accept-Language: en
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;)
Referer: http://google.com
```

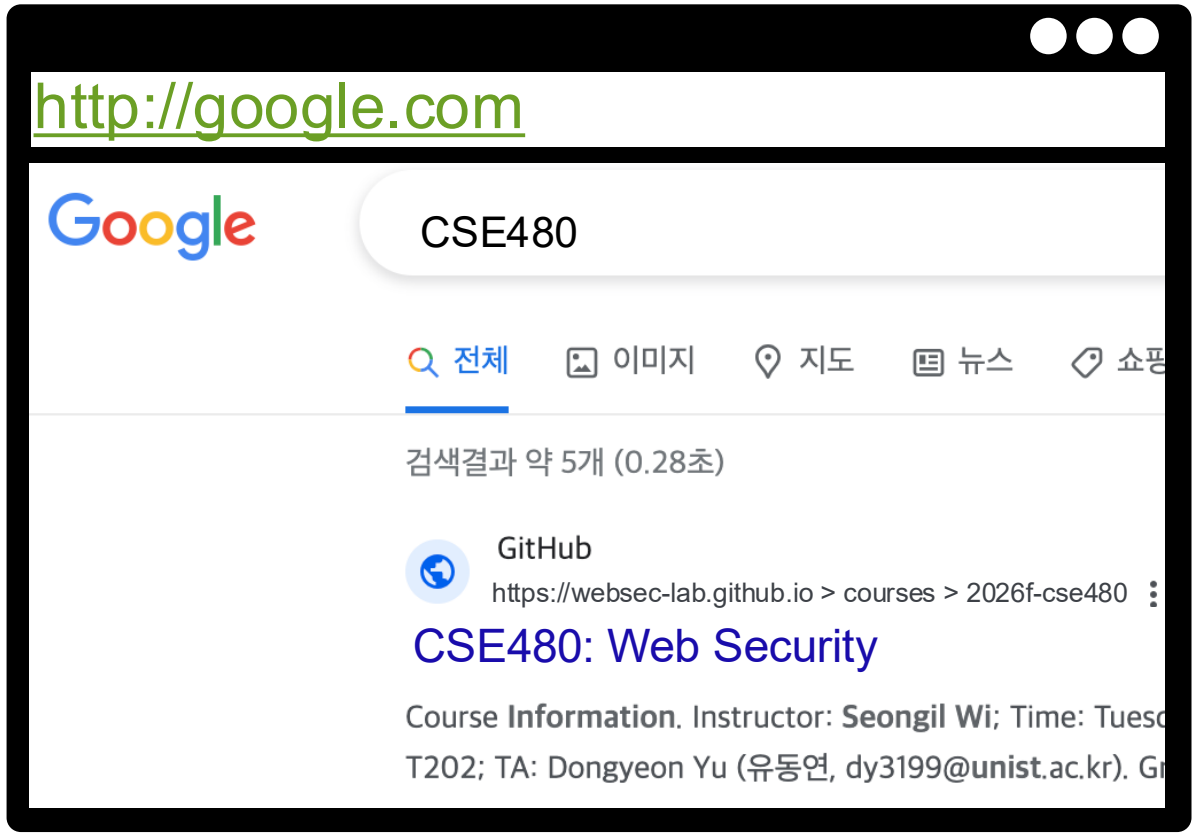
Many HTTP Methods

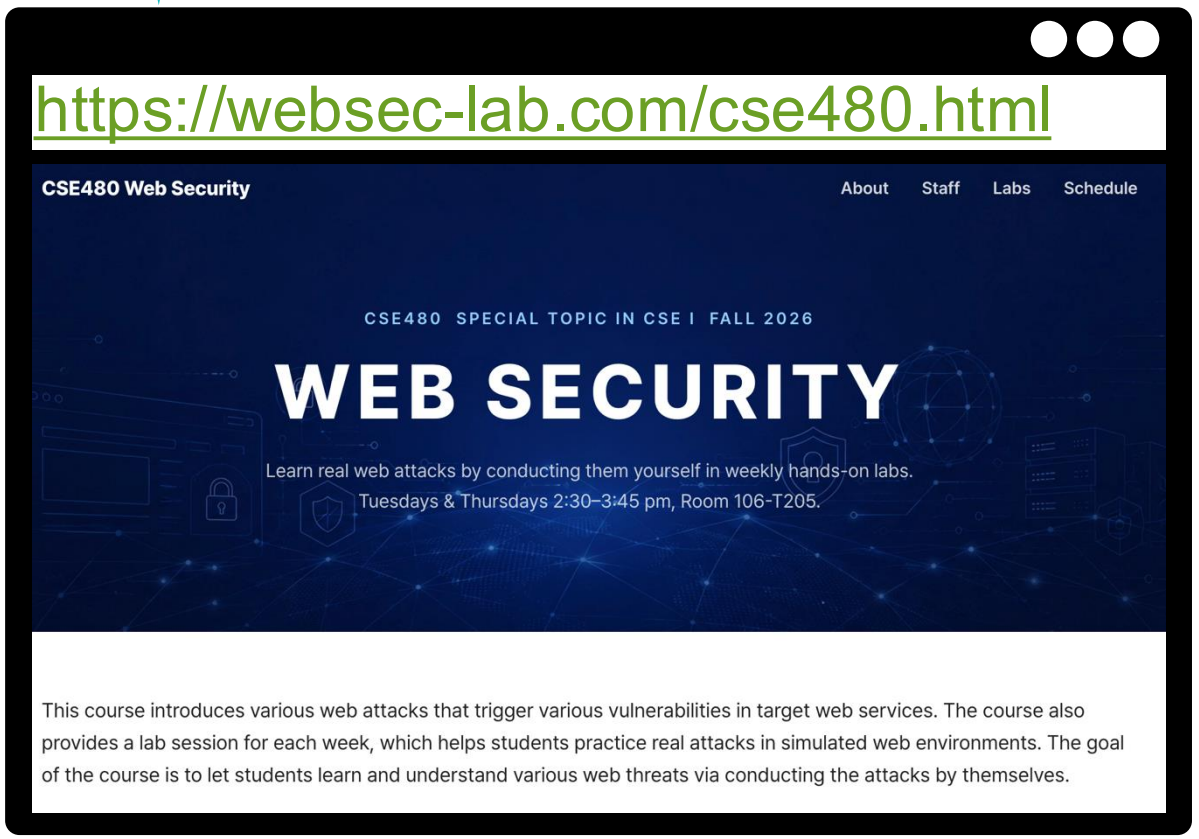
- **GET**: Get the resource at the specified URL
- **POST**: Create new resource at URL with payload
- **PUT**: Replace current representation of the target resource with request payload
- **PATCH**: Update part of the resource
- **DELETE**: Delete the specified URL

HTTP Request



Referrer Header



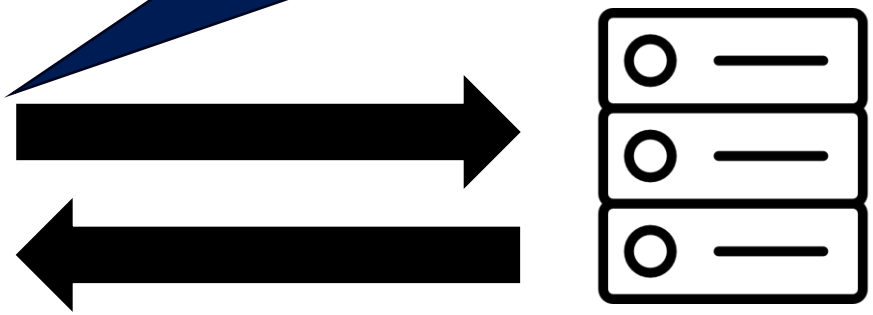


Referrer Header

Referrer Header in Detail

Referrer Header in Detail

(2) Send HTTP request (with referrer header)



Web server
websec-lab.github.io

The server can analyze **where the request originated**

Question

Are there any security issues with the referrer header?

HTTP Response

Status Line

Response headers

Response body

```
HTTP/1.1  200 OK
Date: Sat, 21 Oct 2023 07:58:24 GMT
Connection: Keep-alive
Content-Type: text/html
Content-Length: 2543

<html>
  <body>
    some data...
  </body>
</html>
```

HTTP Response

HTTP version | Status code | Status text

Status Line

Response headers

Response body

```
HTTP/1.1  200 OK
Date: Sat, 21 Oct 2023 07:58:24 G
Connection: Keep-alive
Content-Type: text/html
Content-Length: 2543

<html>
  <body>
    some data...
  </body>
</html>
```

HTTP STATUS CODES

2xx Success

200	Success / OK

3xx Redirection

301	Permanent Redirect
302	Temporary Redirect
304	Not Modified

4xx Client Error

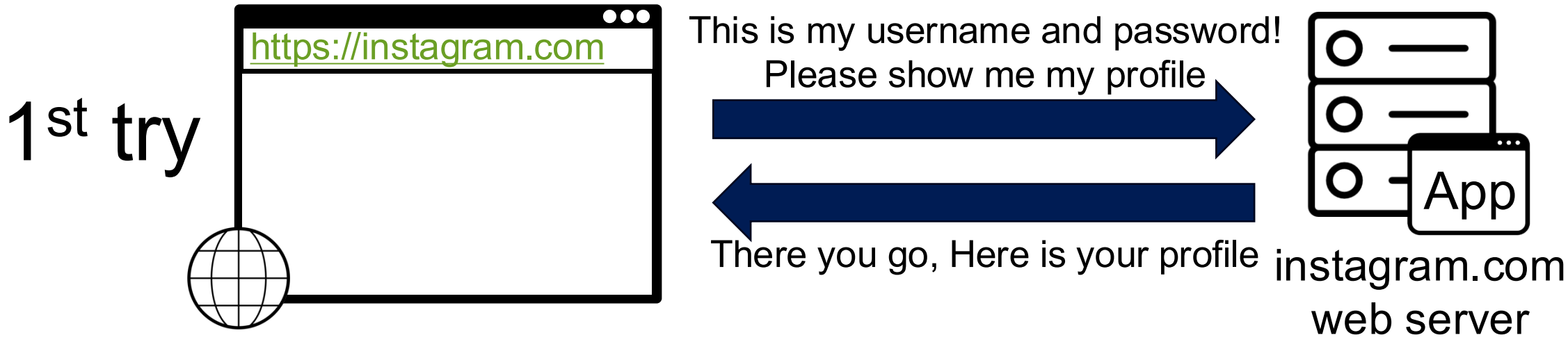
401	Unauthorized Error
403	Forbidden
404	Not Found
405	Method Not Allowed

5xx Server Error

501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

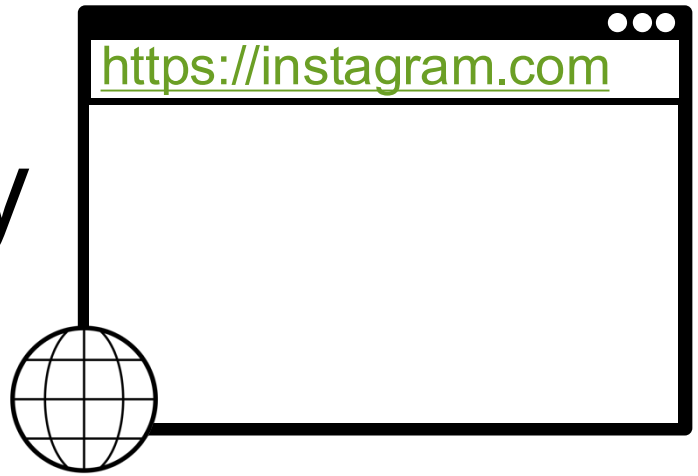
INFIDIGIT

HTTP is a Stateless Protocol



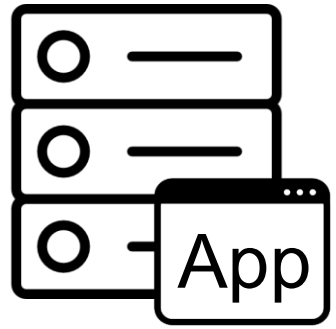
HTTP is a Stateless Protocol

1st try



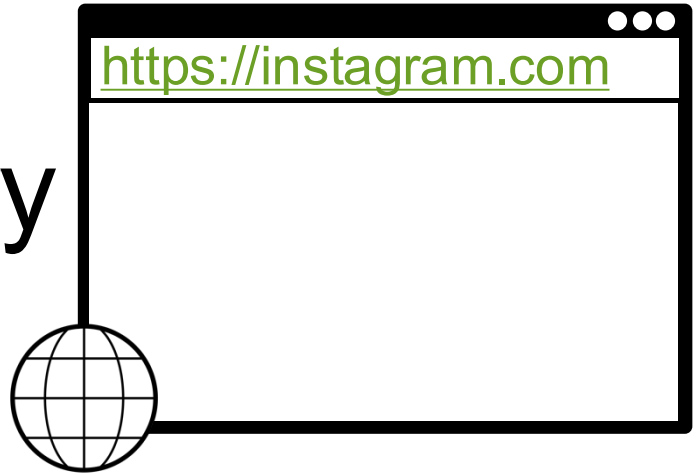
This is my username and password!
Please show me my profile

There you go, Here is your profile



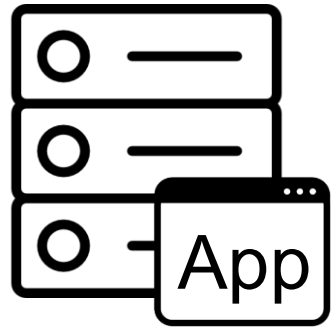
instagram.com
web server

2nd try



Hey Instagram,
show me my profile

Who are you?
Please login and verify who you are…



instagram.com
web server

HTTP is a Stateless Protocol

Stateless protocol:
Each request is independent to previous request

2nd try

is my username and password!
Please show me my profile

There you go, Here is your profile

App

instagram.com
web server

Hey Instagram,
show me my profile

Who are you?
Please login and verify who you are…

App

instagram.com
web server

Question

How to make HTTP “act” stateful?

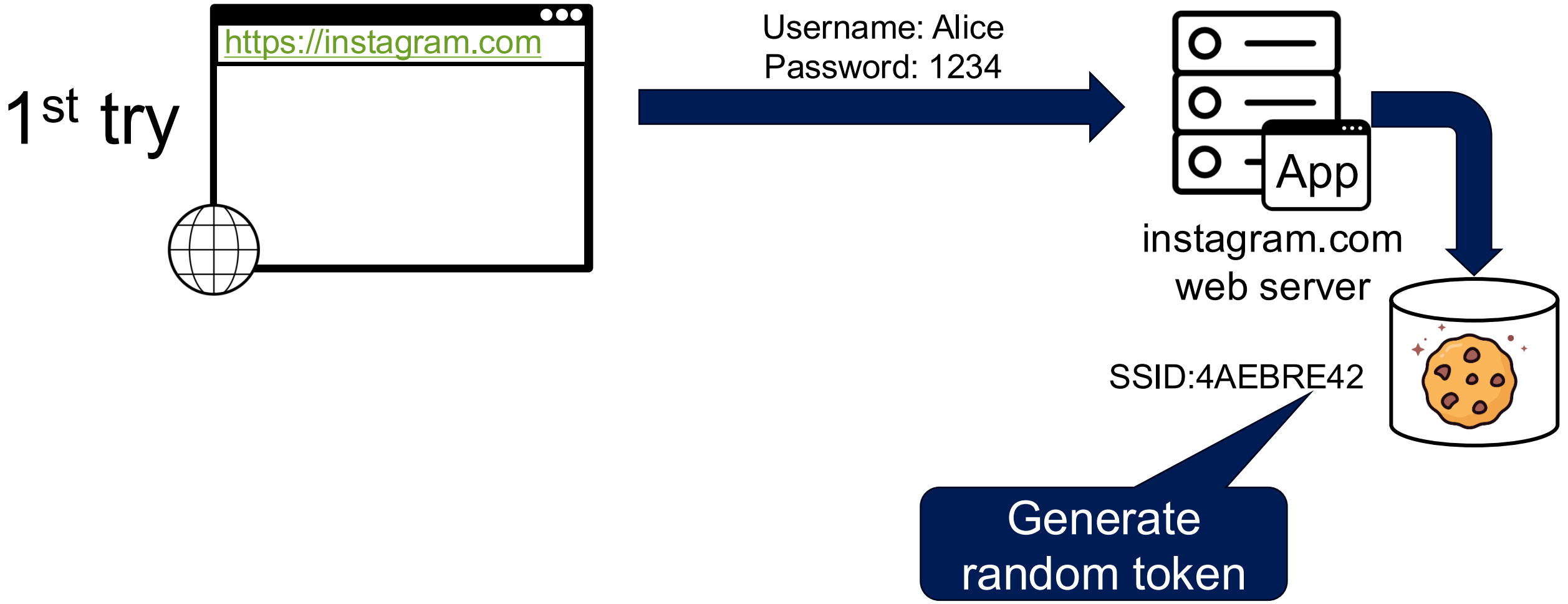
Adding State to HTTP

- Recall: no inherent state in HTTP
  - Server does not keep any state after the connection is closed
- For static content sites, no problem
  - Developing “applications” is impossible though
  - E.g., shopping cart on Amazon
- Need to introduce state in HTTP
  - in the form of “cookies”

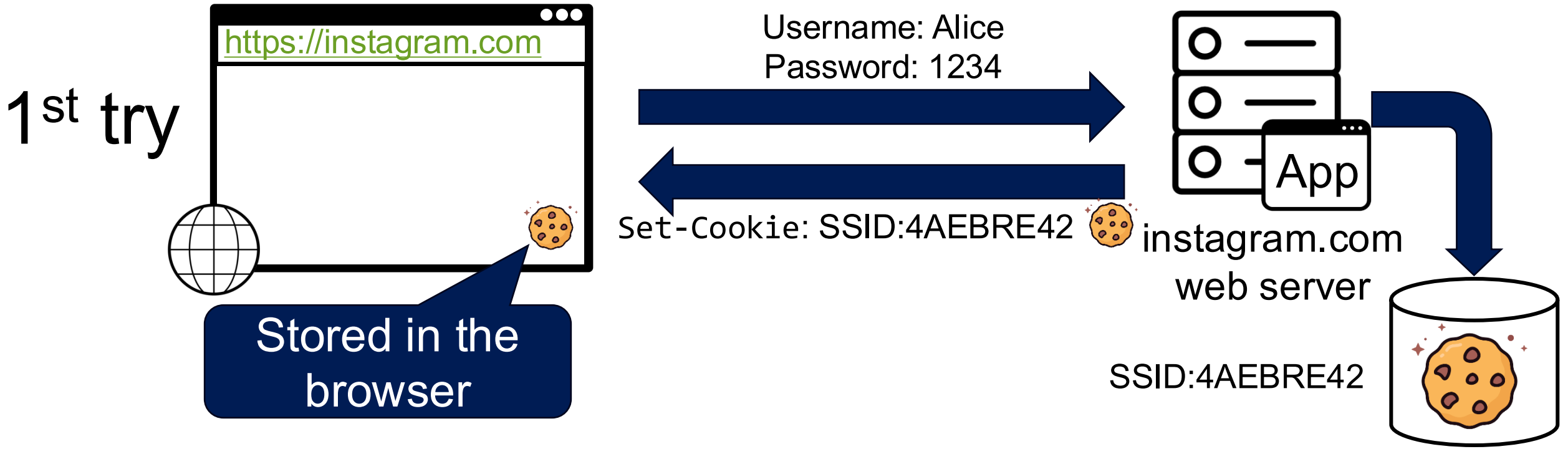
Cookie: Making HTTP Stateful

- HTTP cookie: small piece of data that a server sends to the browser, who stores it and sends it back with subsequent requests

Cookie: Making HTTP Stateful

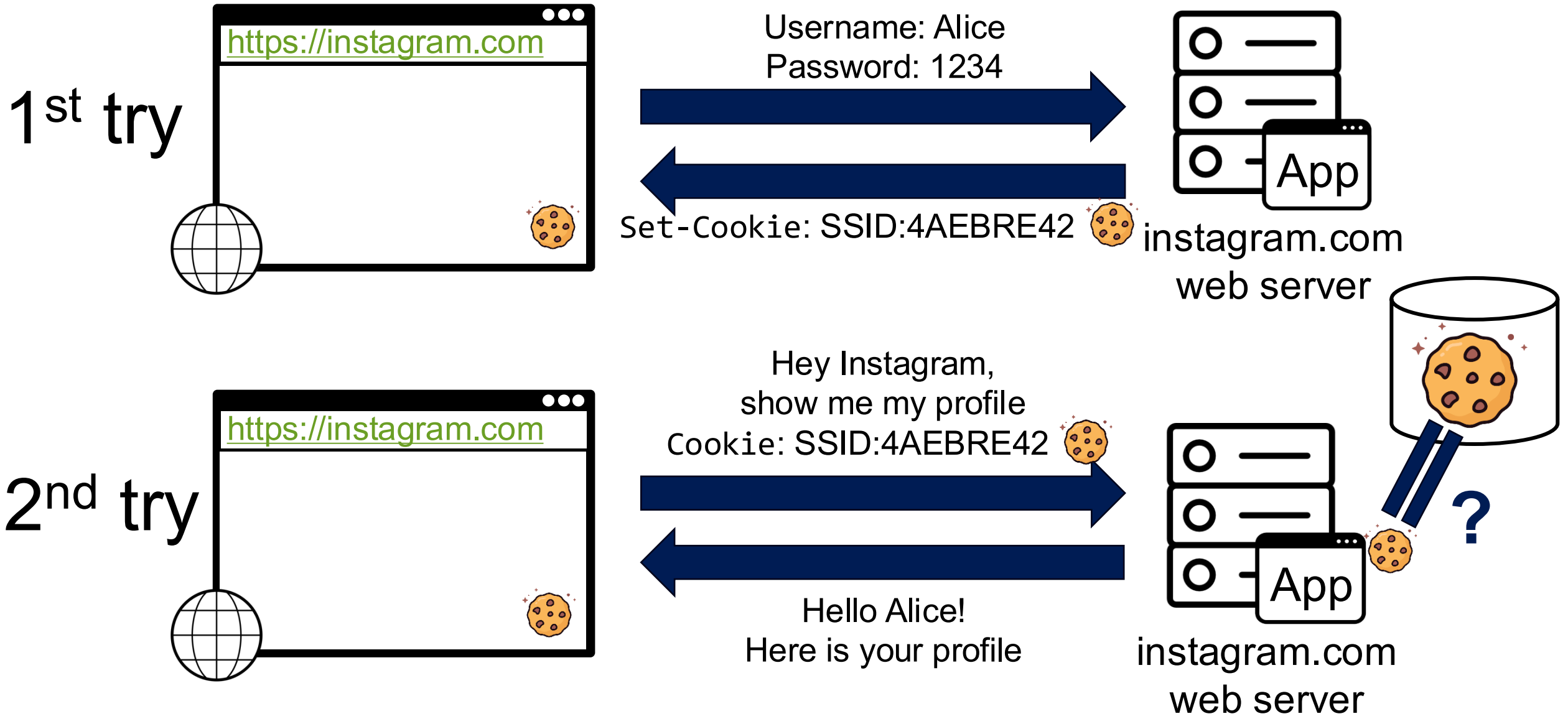


Cookie: Making HTTP Stateful



Cookie: Making HTTP Stateful

Cookie: Making HTTP Stateful



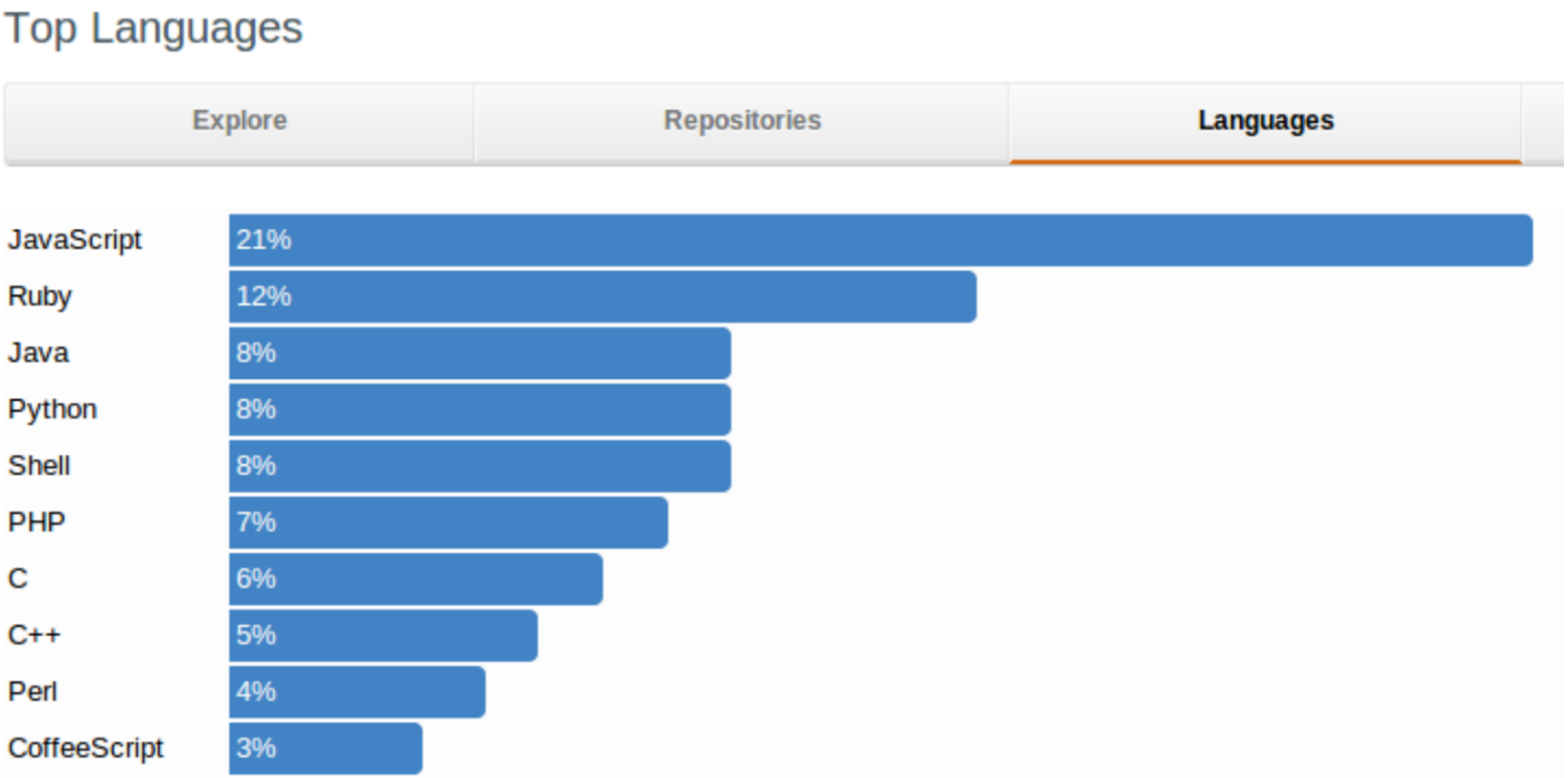
Cookie: Making HTTP Stateful

- Generate random token on first page visit
- Sent to client via `Set-Cookie` header
- Client always sends along cookies in every request to the server

- Cookies are persisted in the browser
  - Controllable by `Expires` option in cookie

JavaScript (JS)

- Most popular language in the world!



JavaScript (JS)

- Developed by Brendan Eich at Netscape
- Later standardized for browser compatibility
  - ECMAScript Edition 3 (a.k.a., JavaScript 1.5)

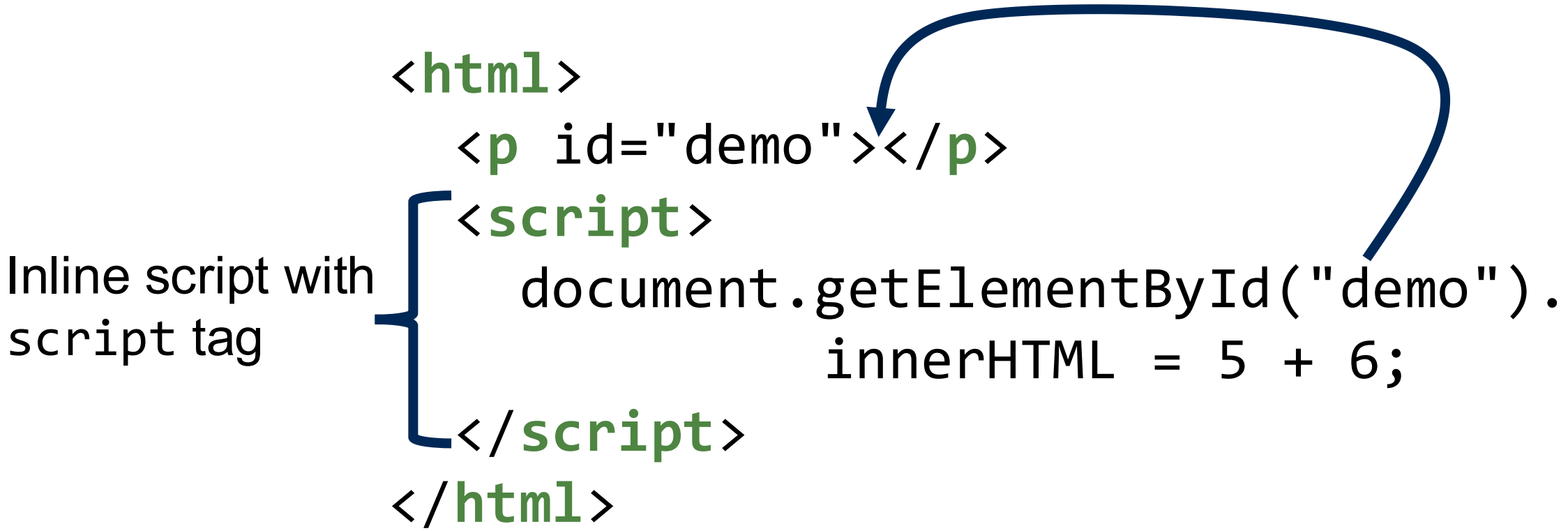
- HTML may contain JS program code to make web pages more dynamic

JS Example (1)


```
<html>
  <p id="demo"></p>
  <script>
    document.getElementById("demo").
                innerHTML = 5 + 6;
  </script>
</html>
```

https://websec-lab.github.io/courses/2026f-cse480/demo/demo1.html

JS Example (1)



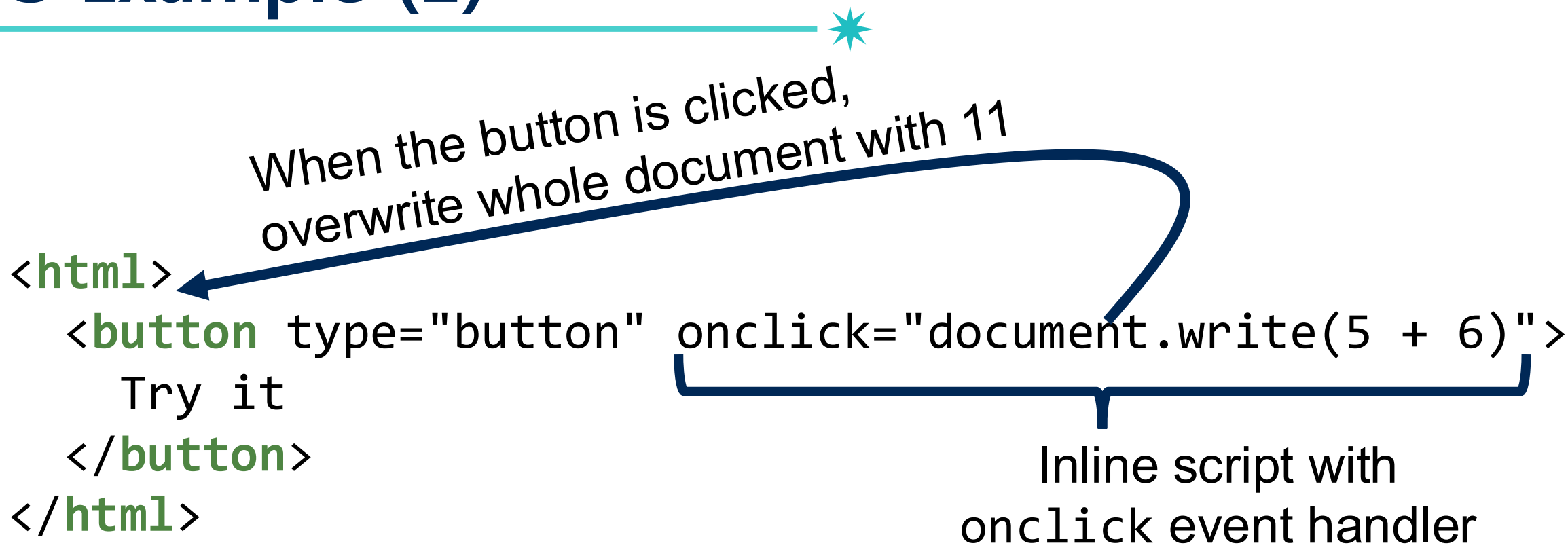
https://websec-lab.github.io/courses/2026f-cse480/demo/demo1.html

JS Example (2)


```
<html>
  <button type="button" onclick="document.write(5 + 6)">
    Try it
  </button>
</html>
```

https://websec-lab.github.io/courses/2026f-cse480/demo/demo2.html

JS Example (2)



https://websec-lab.github.io/courses/2026f-cse480/demo/demo2.html

JS Example (3)

index.html

```
<html>
  <script src="write.js">
  </script>
</html>
```

write.js

```
document.write(5 + 6)
```

https://websec-lab.github.io/courses/2026f-cse480/demo/demo3.html

JS Example (3)

Overwrite whole document with 11

index.html

```
<html>
  <script src="write.js">
  </script>
</html>
```

write.js

```
document.write(5 + 6)
```

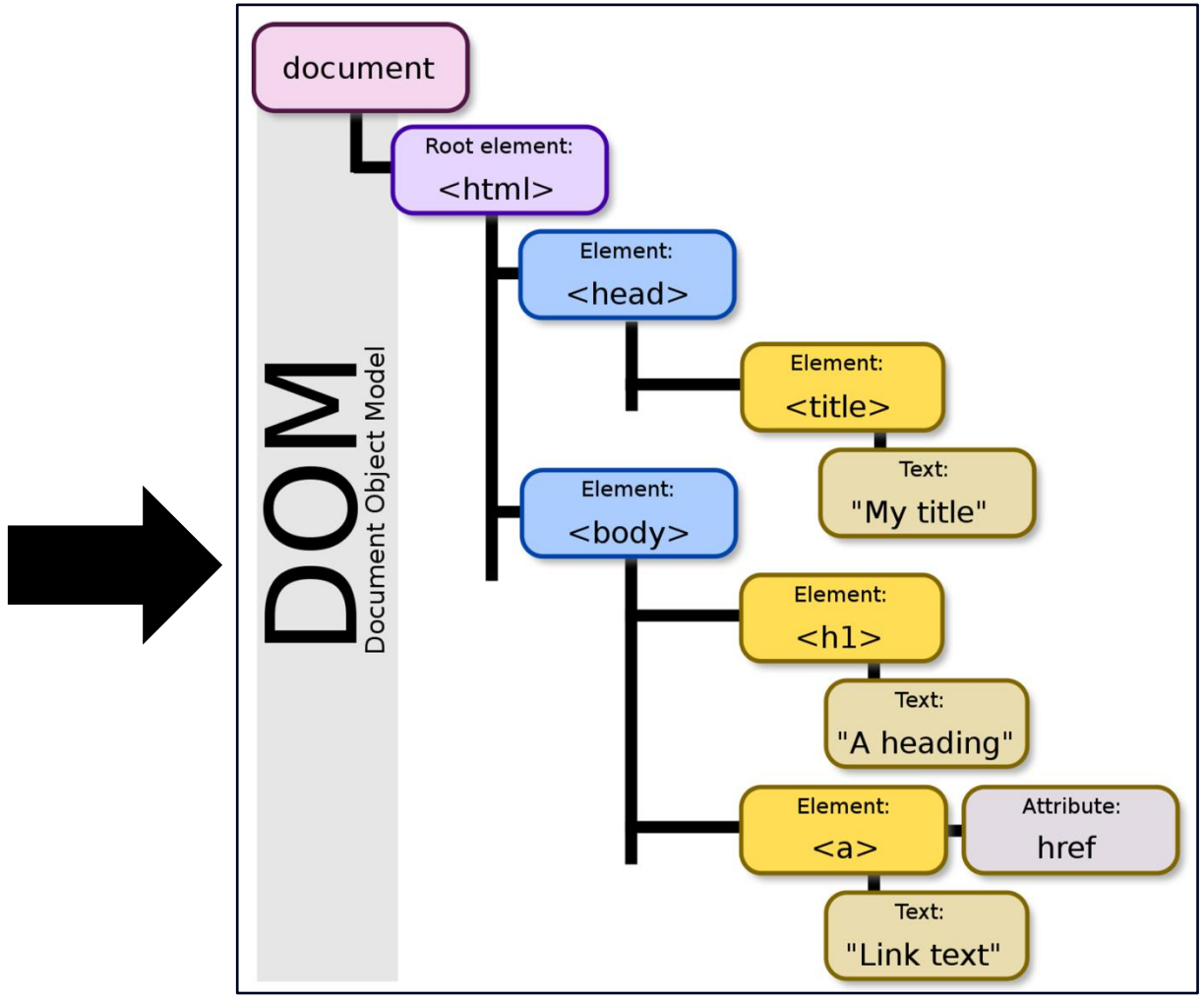
External script with `src` attribute

https://websec-lab.github.io/courses/2026f-cse480/demo/demo3.html

Document Object Model (DOM)

- An HTML document: structured data

```
<html>
  <head>
    <title>
      My title
    </title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href=“cse480.com”>Link text</a>
  </body>
<body>
```



DOM and JS APIs

- Exposed to JavaScript through global objects
  - `document`: Access to the document (e.g., cookies, head/body)
  - `navigator`: Information about the browser (e.g., UA, plugins)
  - `screen`: Information about the screen (e.g., dimension, color depth)
  - `location`: Access to the URL (read and modify)
  - `history`: Navigation

Changing HTML DOM using JS

- JavaScript can change all the HTML DOM components in the page!

- using several APIs
  - `createElement(elementName)`
  - `createTextNode(text)`
  - `appendChild(newChild)`
  - `removeChild(node)`

Changing HTML DOM using JS (Example)


```
<html>
  <body>
    <ul id=“t1”>
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1

Changing HTML DOM using JS (Example)


```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

- Item 1

```
<script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode('Item 2')
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
```

Changing HTML DOM using JS (Example)

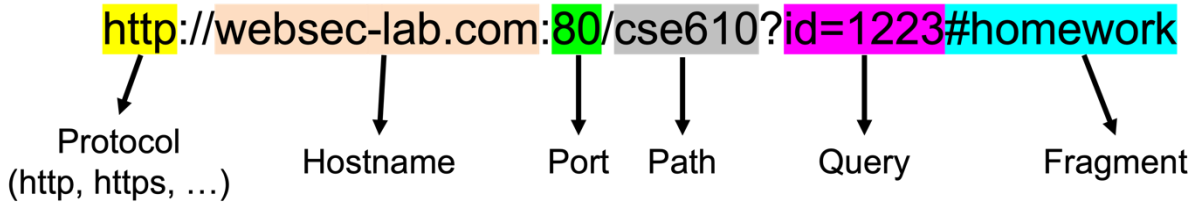

```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

```
<script>
  var list = document.getElementById('t1')
  var newitem = document.createElement('li')
  var newtext = document.createTextNode('Item 2')
  list.appendChild(newitem)
  newitem.appendChild(newtext)
</script>
```

- Item 1
- Item 2

Accessing HTML DOM using JS (Example)

- `location.protocol`: protocol
- `location.hostname`: only HTTP host
- `location.port`: only the port
- `location.pathname`: path



- We can display all cookies for current document by `alert(document.cookie)`

Server-side Web Application

- Runs on a web server (application server)
- Can be implemented in many existing programming languages
  - PHP (Most popular!), Java, Python, Ruby on Rail, JavaScript (Node.js)

Server-side
web application

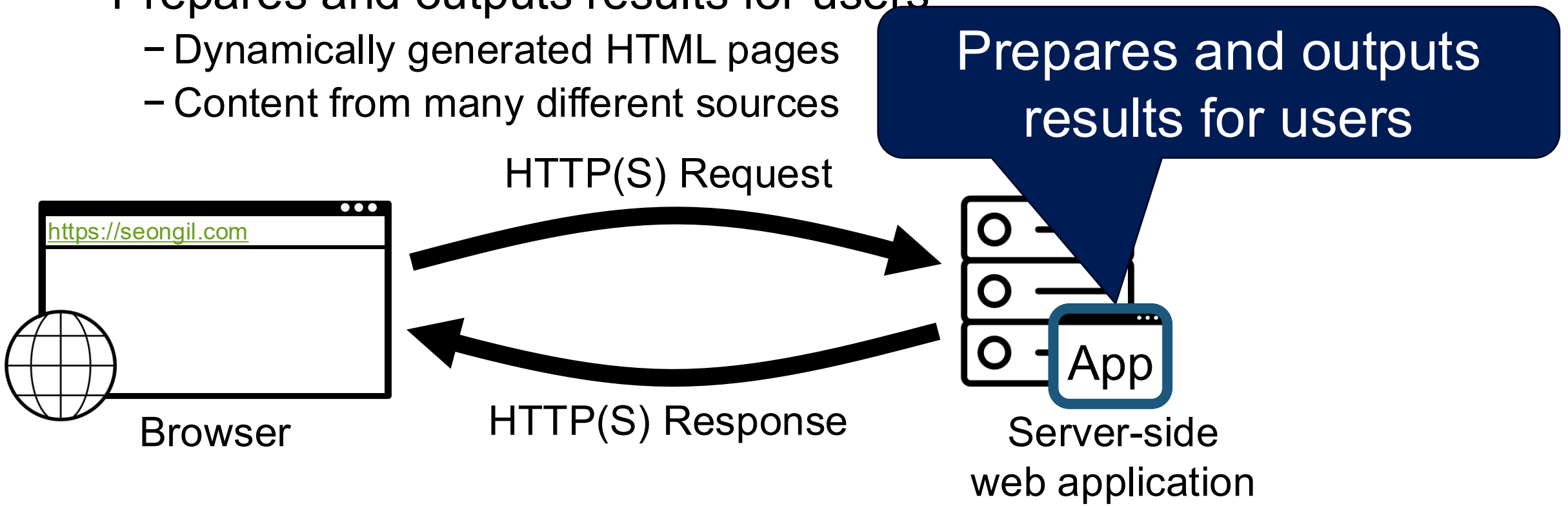
Server-side Web Application

- Runs on a web server (application server)
- Can be implemented in many existing programming languages
  - PHP (Most popular!), Java, Python, Ruby on Rail, JavaScript (Node.js)
- Prepares and outputs results for users
  - Dynamically generated HTML pages
  - Content from many different sources

Server-side
web application

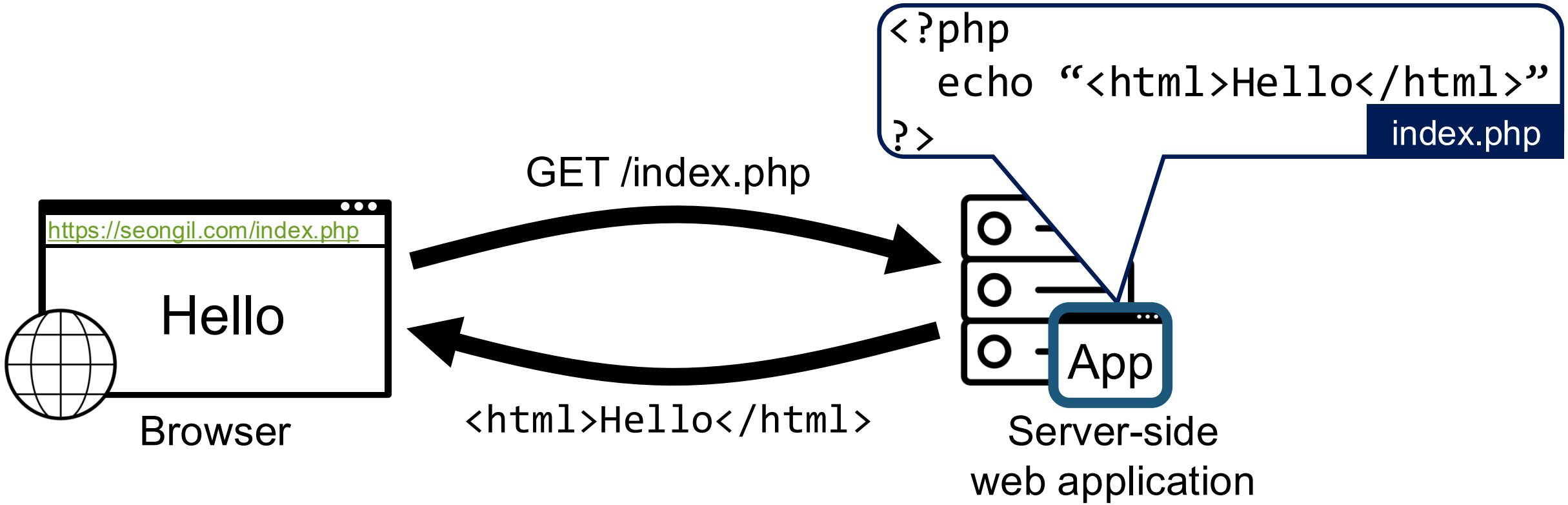
Server-side Web Application

- Runs on a web server (application server)
- Can be implemented in many existing programming languages
  - PHP (Most popular!), Java, Python, Ruby on Rail, JavaScript (Node.js)
- Prepares and outputs results for users
  - Dynamically generated HTML pages
  - Content from many different sources



Hypertext Preprocessor (PHP)

- A general-purpose scripting language that is especially suited to server-side web application development



PHP Variables and APIs: Example

Variable	Description
`$_GET`	Handles query-string parameters passed through the URL
`$_POST`	Handles data submitted via POST requests
`$_REQUEST`	Handles request data from GET, POST, and COOKIE
`$_SERVER`	Provides server environment and request information
`$_SESSION`	Stores and accesses session data across multiple requests

Do not memorize, just understand!

PHP Variables and APIs: Example

API	Description
`echo / print`	Outputs strings or values to the HTTP response
`include / require`	Executes another PHP file
`mysqli_query()`	Executes an SQL query on a MySQL database
`system()`	Executes an OS command and outputs the result

Do not memorize, just understand!

Question?