CSE610: Web Programming & Security

2. Web Programming

Seongil Wi

Department of Computer Science and Engineering

Notice: Paper Presentation

- Considering the correction period (정정기간), I will provide a list of papers **after the next class**
- Paper list distribution: 3/5, 6 PM
- Selection of papers for presentation will be on a first-come, first-served basis (선착순)

Notice: Term Project

- 1~2 persons for one team
- The topics must be related to the web security/web-related security
- Submit your proposal by **3/15, 11:59 PM**

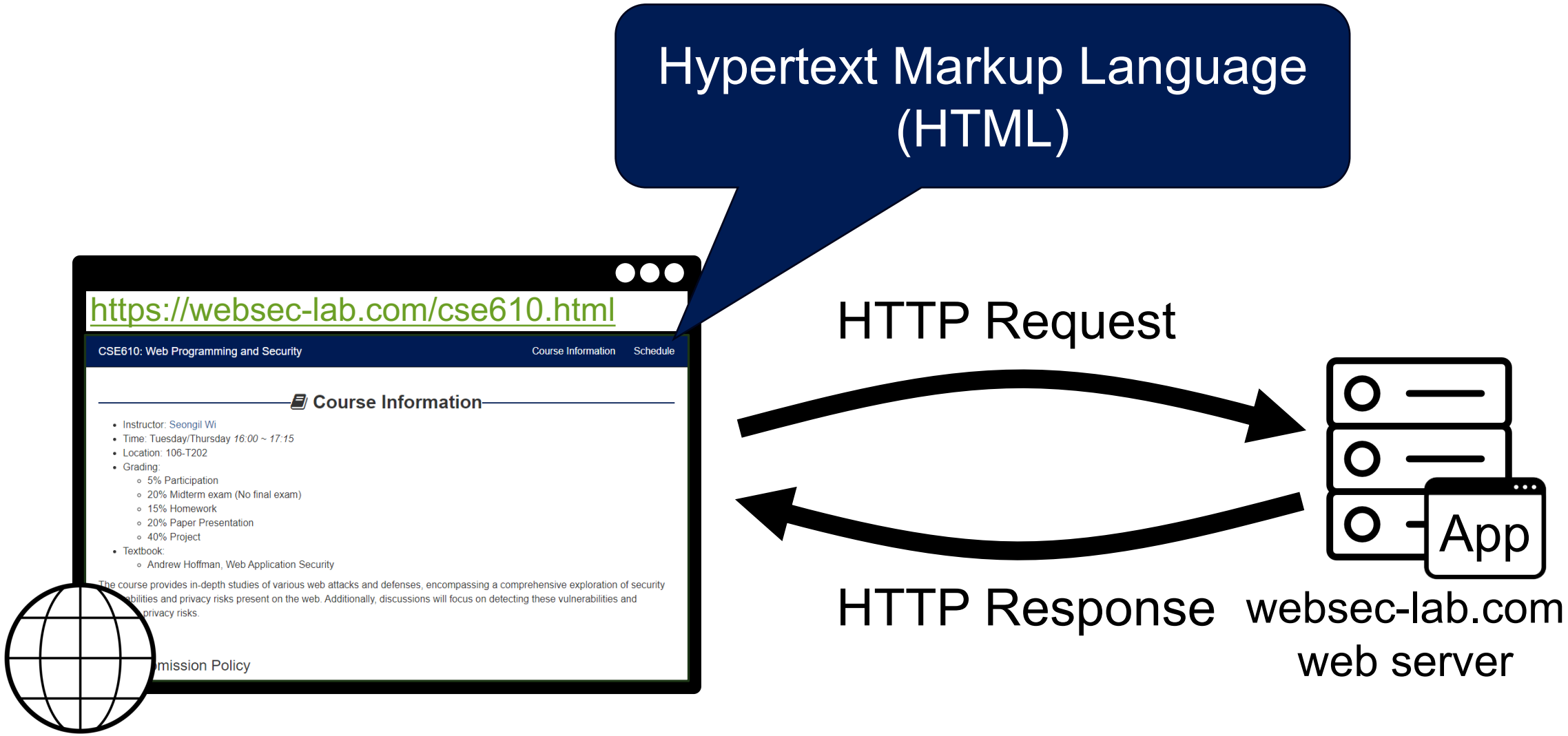
Proposal Submission Guidelines

- You should upload a single PDF file on BlackBored.
- The name of the PDF file should have the following format: `[your ID-last name.pdf]`
  - If your name is Gil-dong Hong, and your ID is 20231234, then you should submit a file named “20231234-Hong.pdf”
  - If your team consists of two people, each member must submit a PDF file

- **Your proposal must follow the following format:**
  - Template: Double-Column ACM format (Sigconf style) – provided on BlackBored
  - 2 pages maximum (reference is excluded)
  - Format: Background, Motivation, Proposed Idea, Expected Results, Research Timeline, Reference

Web Programming Basic

Web Infrastructure



Hypertext Markup Language (HTML)

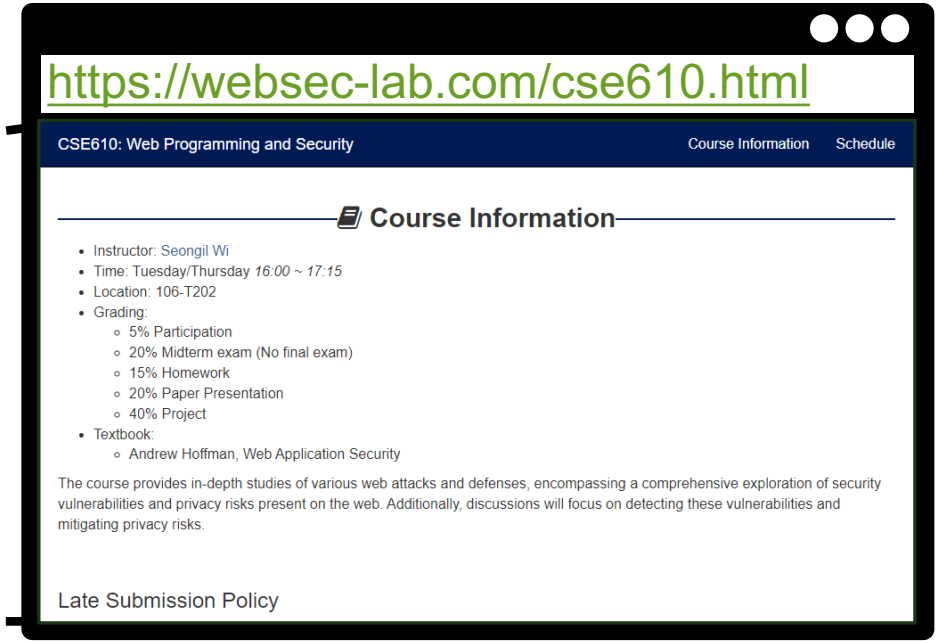
- Markup language for web page layout
  - NOT programming language (i.e., for computation)!
- A web page (document) is written in HTML using markup tags
  - E.g., `<p>`, `<img>`

```
<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

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

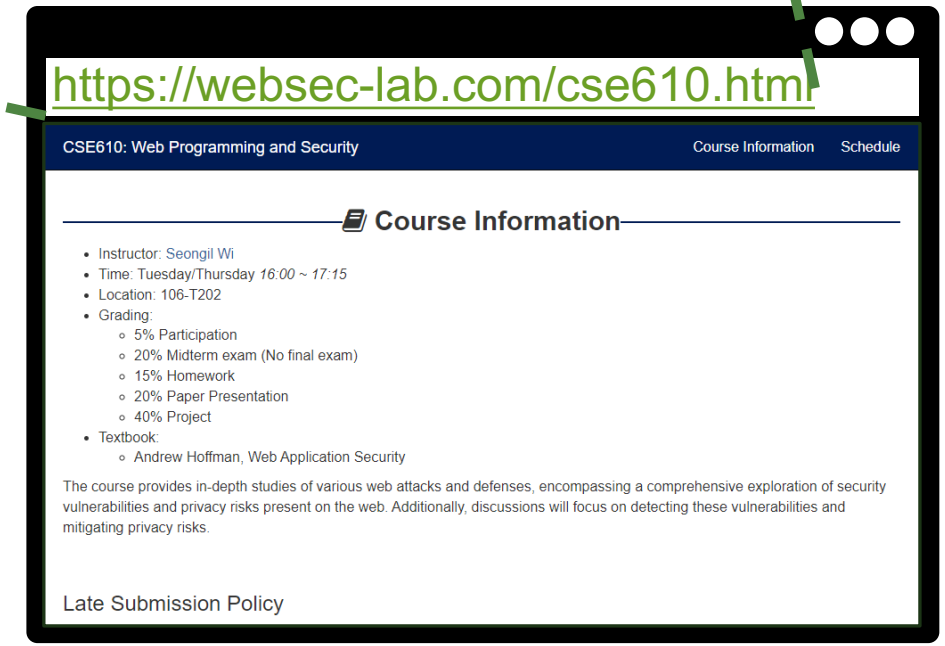


What if we need computation?
⇒ JavaScript!

Uniform Resource Locators (URLs)

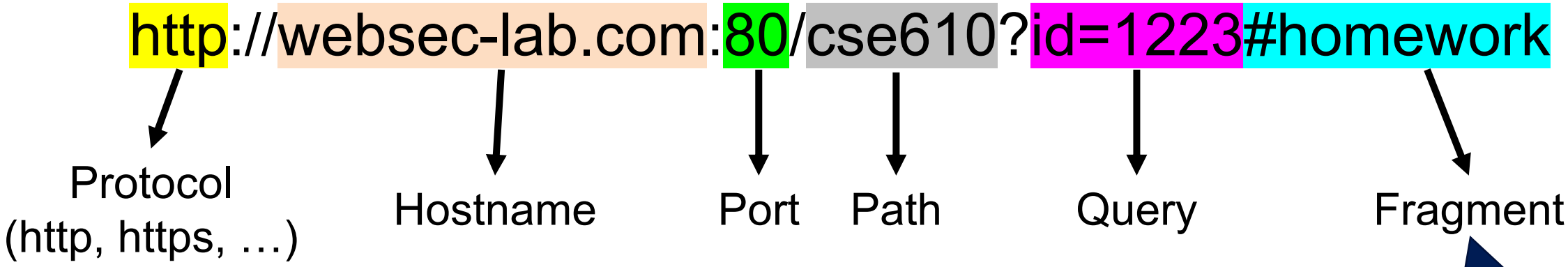
- Global identifiers of network-retrievable documents
- Example

http://websec-lab.com:80/cse610?id=1223#homework



Uniform Resource Locators (URLs)

- Global identifiers of network-retrievable documents
- Example

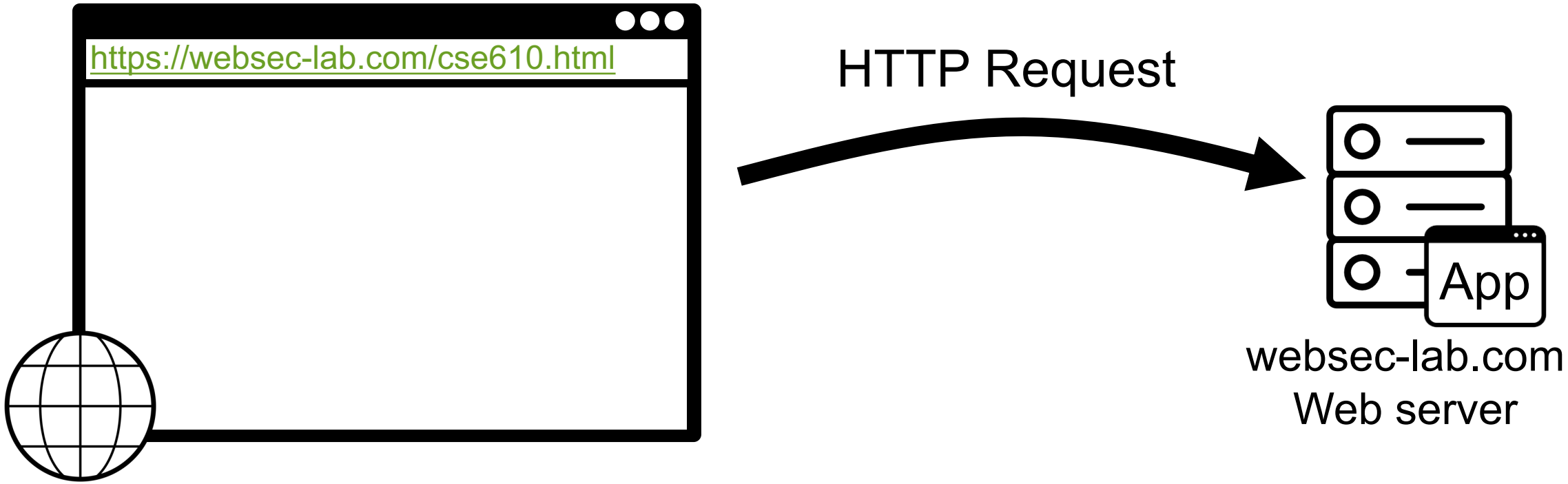


- Special characters are encoded as hex:
  - New line → %0A
  - Space → %20
  - + → %2B

Fragments are not sent to the server

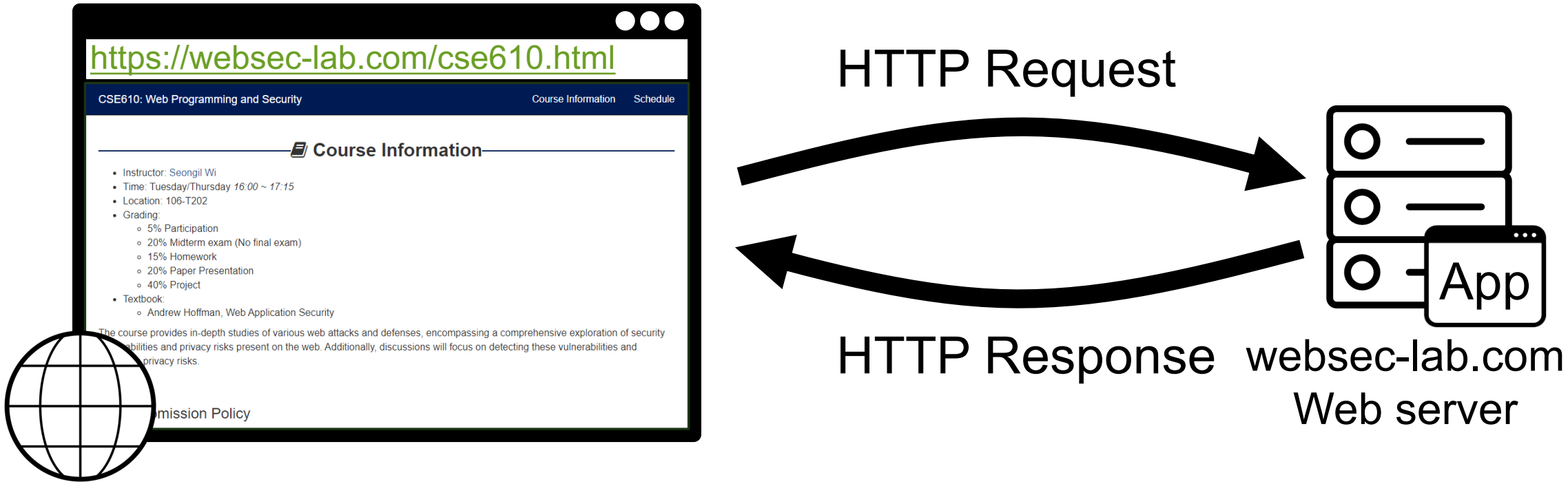
Hyper Text Transfer Protocol (HTTP)

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



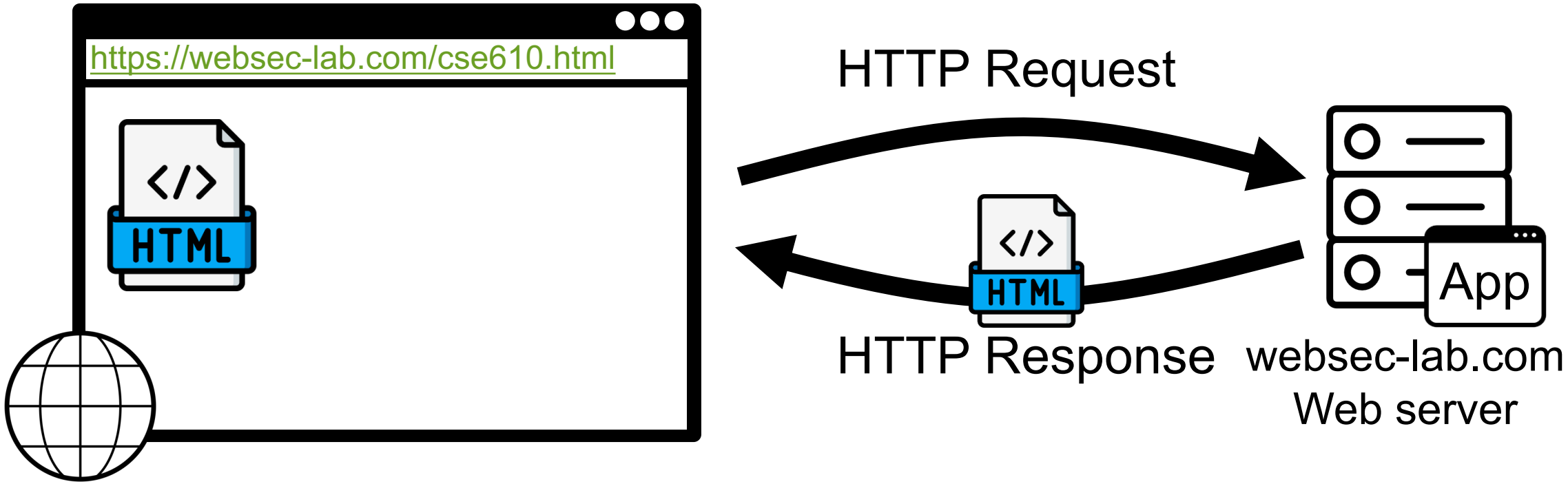
Hyper Text Transfer Protocol (HTTP)

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



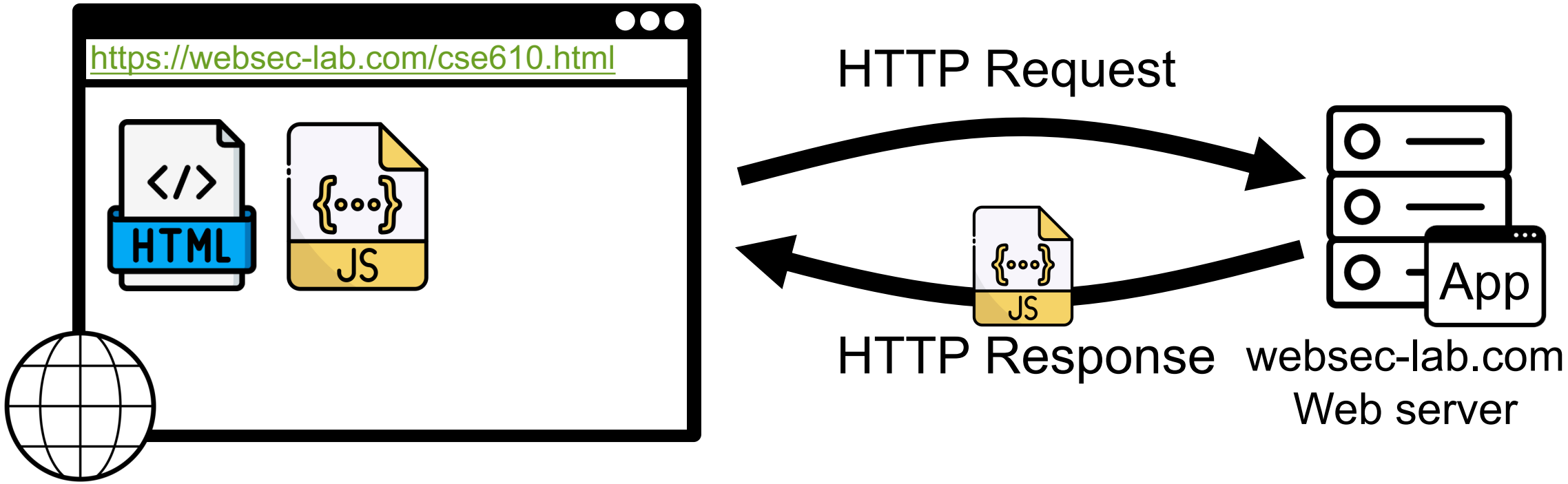
Hyper Text Transfer Protocol (HTTP)

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



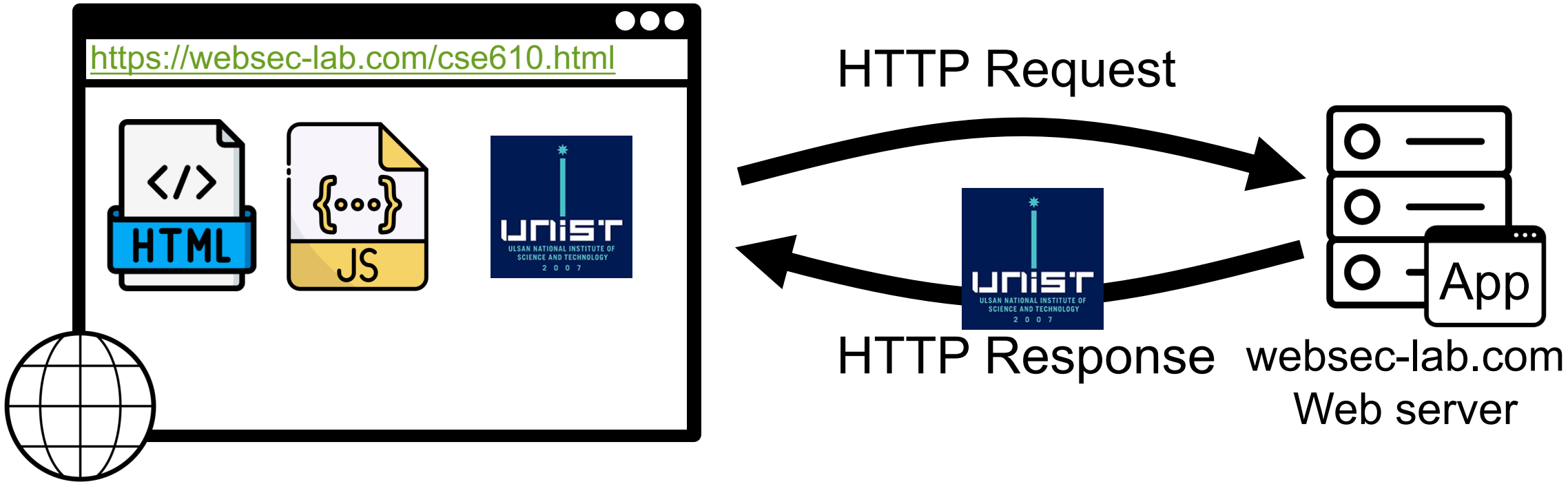
Hyper Text Transfer Protocol (HTTP)

- 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

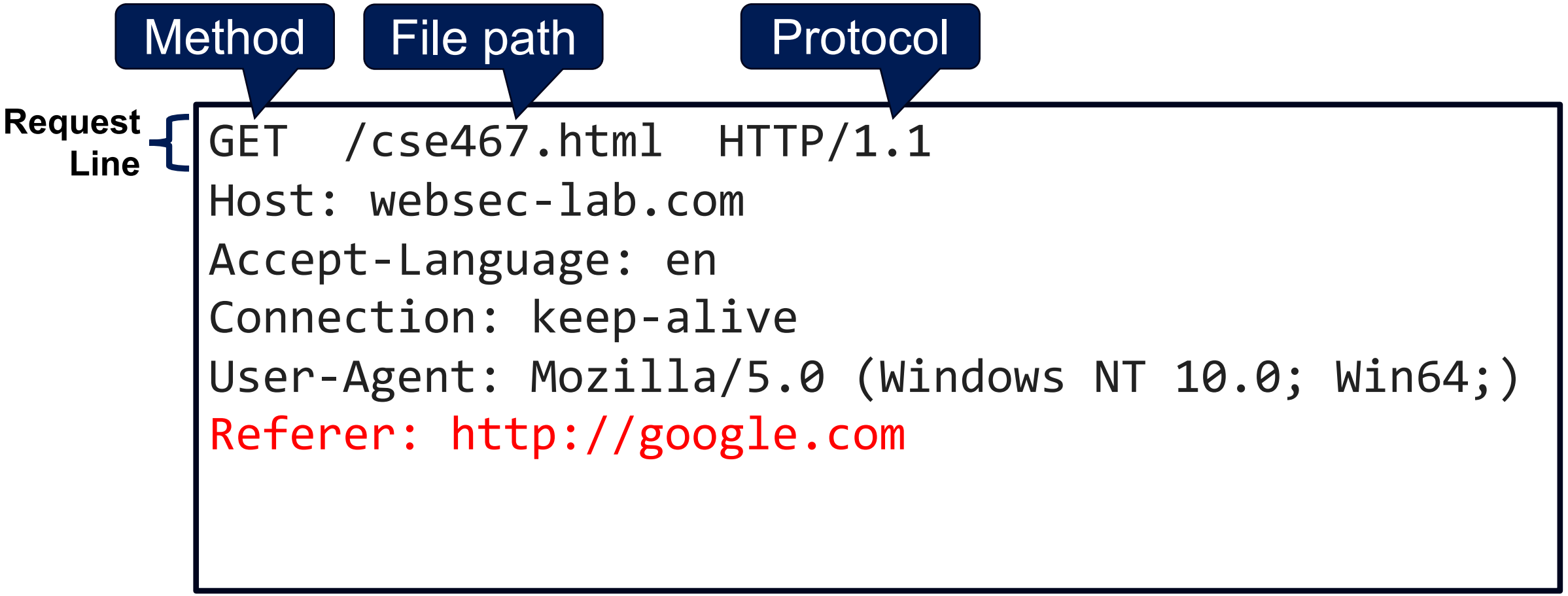
- Application layer protocol
  - A request is sent over a TCP connection on port:80

- 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
```

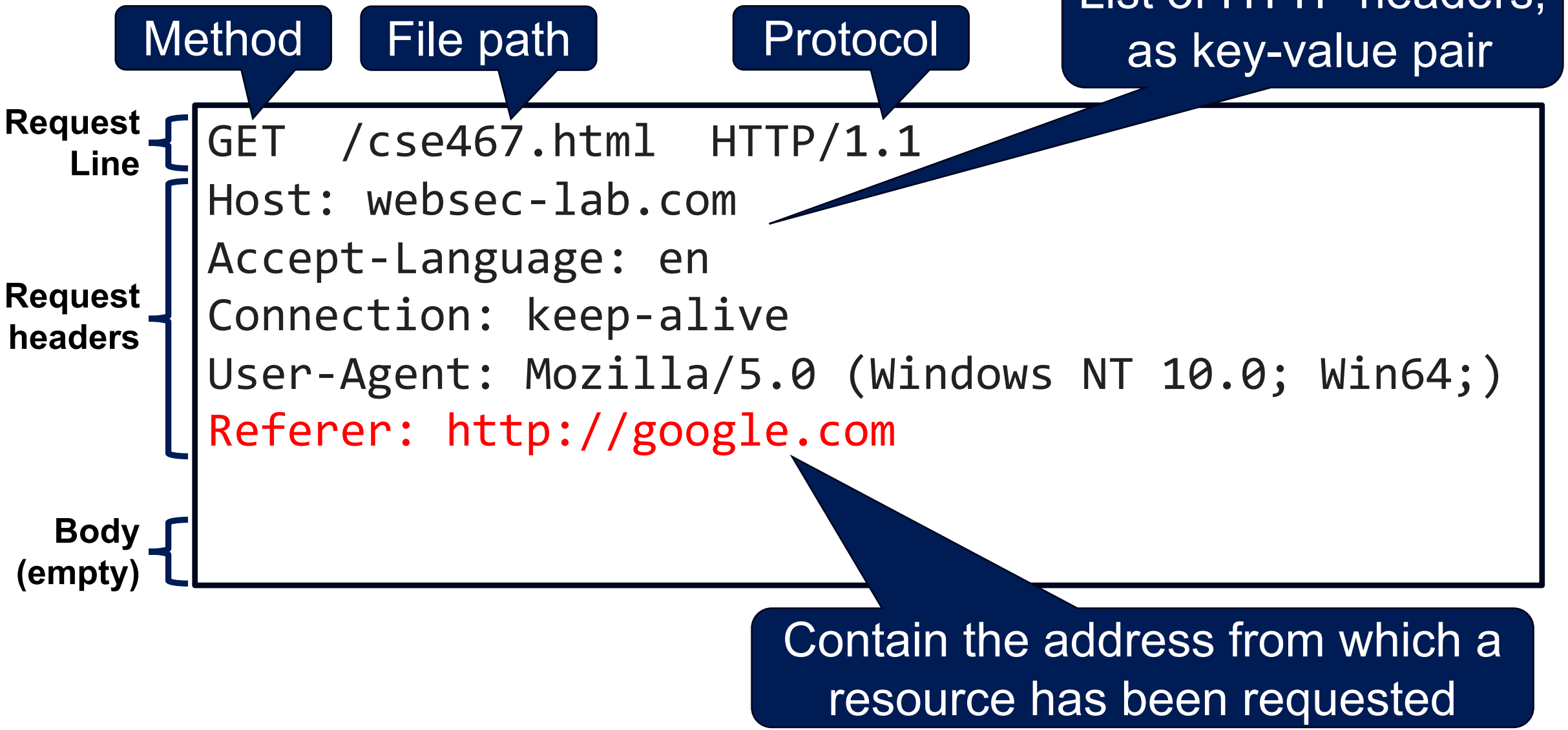
HTTP Request



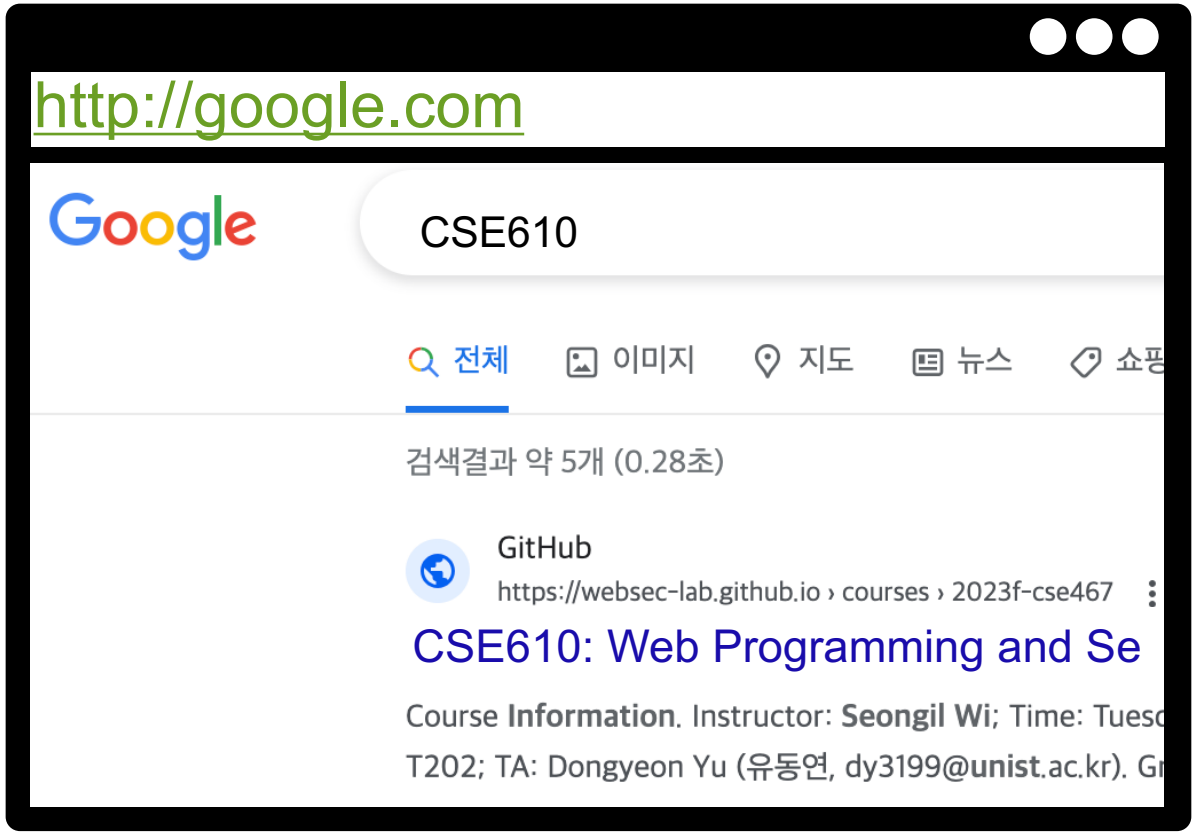
Many HTTP Methods

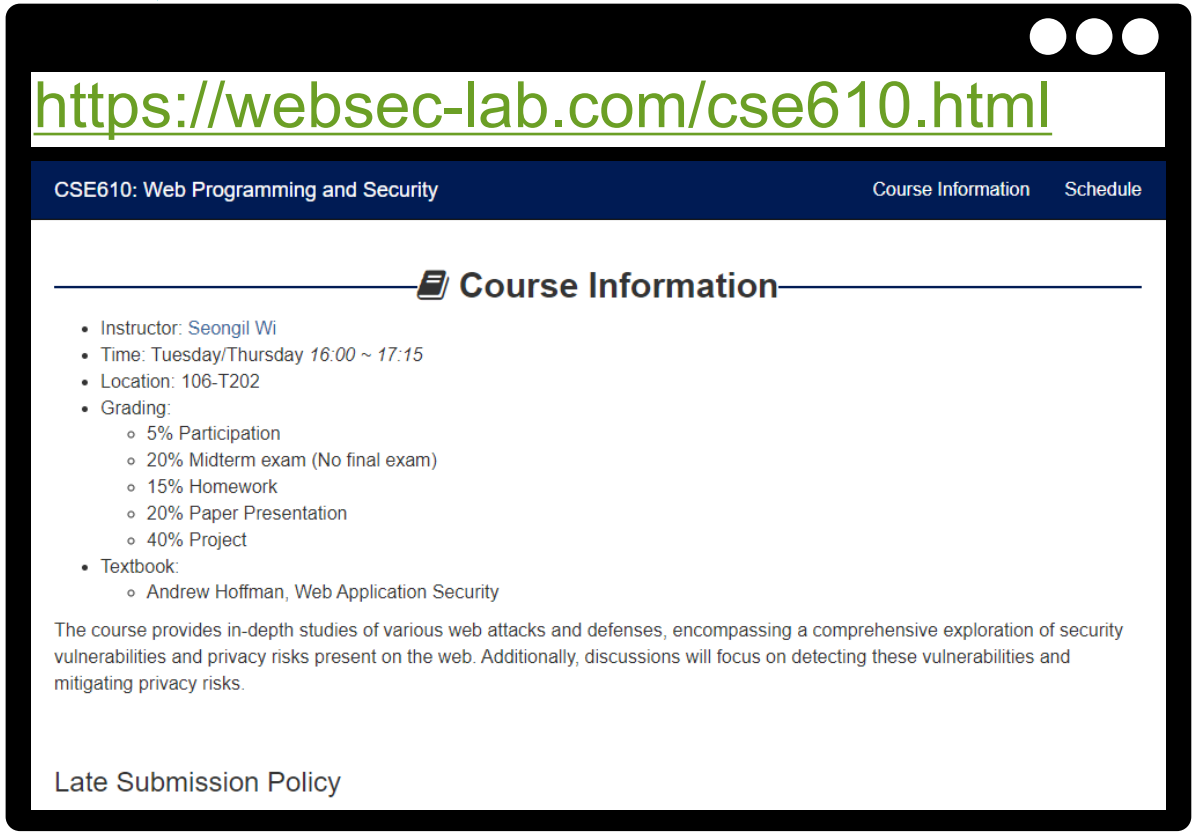
- **GET**: Get the resource at the specified URL
- **POST**: Create new resource at URL with payload
- **PUT**: Replace current representation of the target resource with request payload
- **PATCH**: Update part of the resource
- **DELETE**: Delete the specified URL

HTTP Request

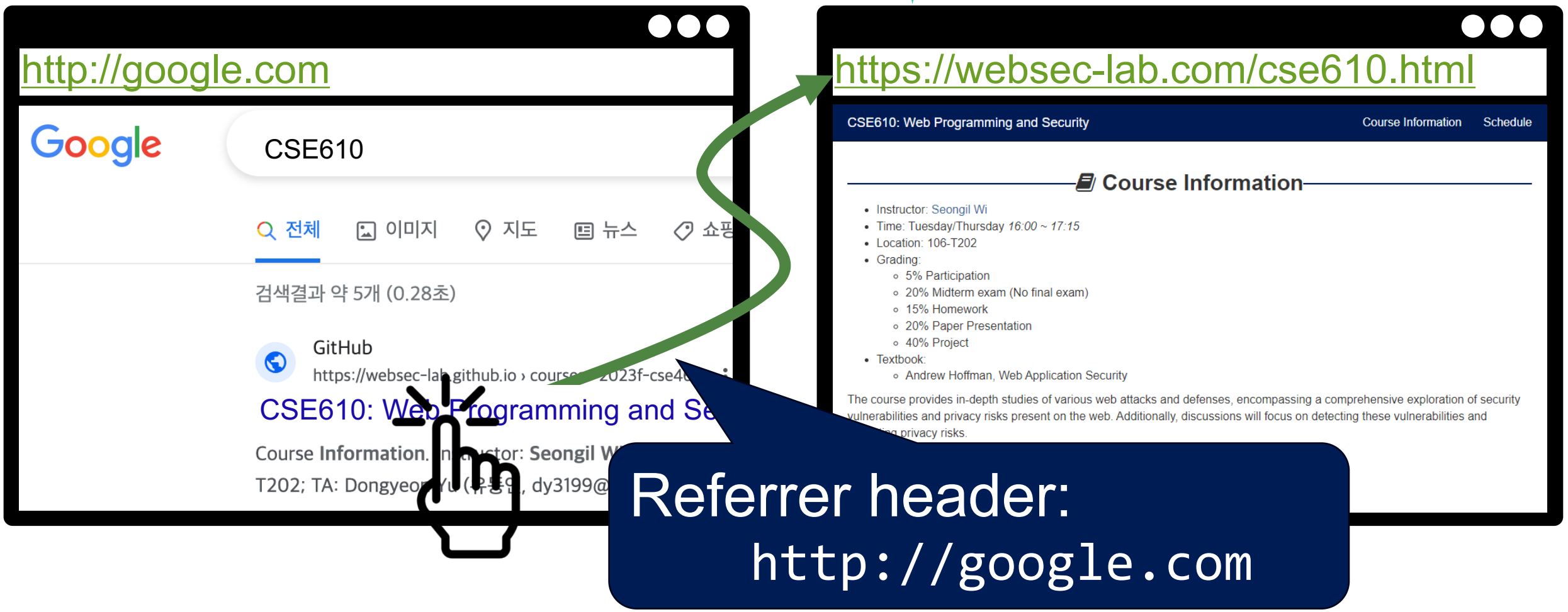


Referrer Header

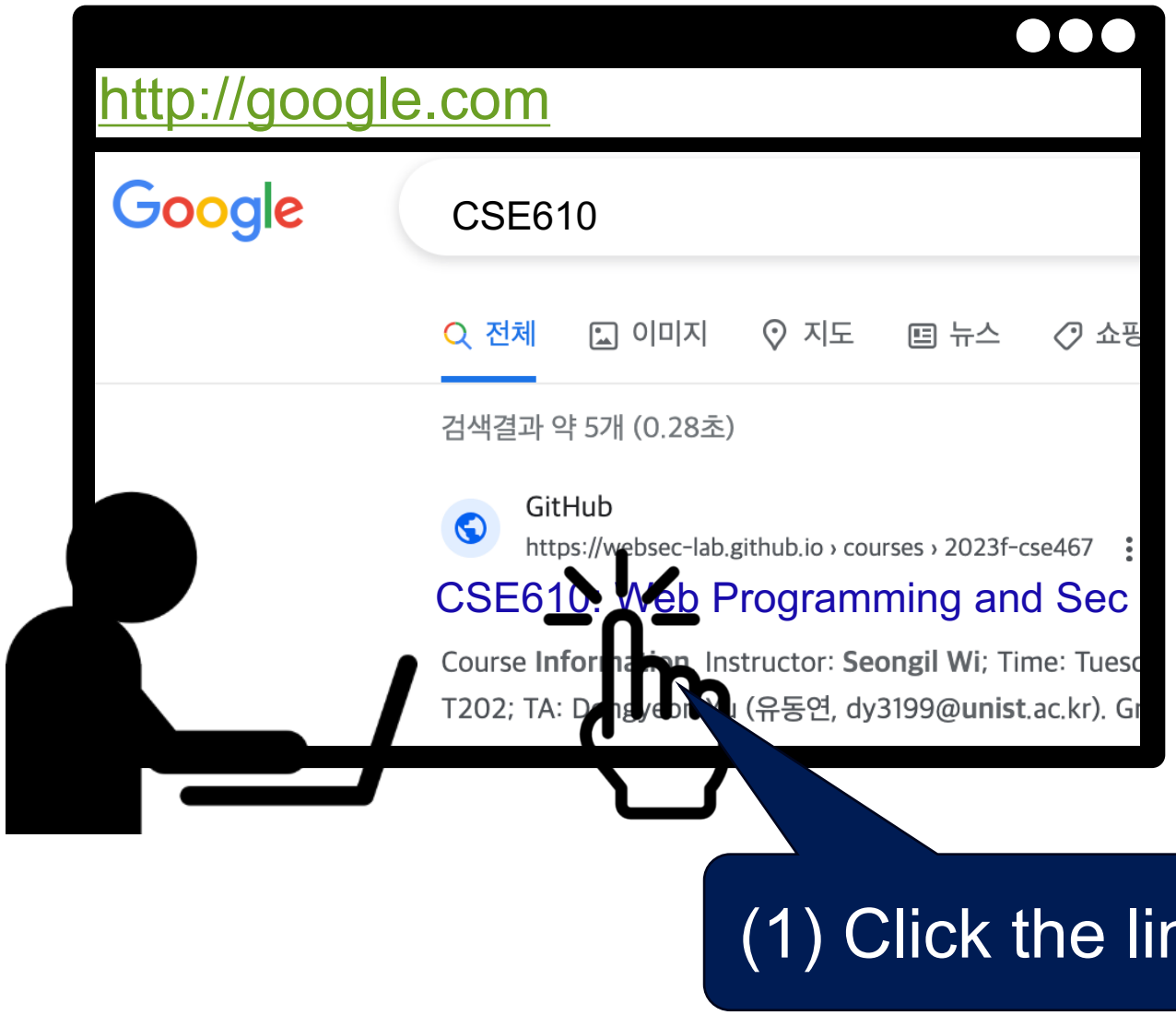




Referrer Header

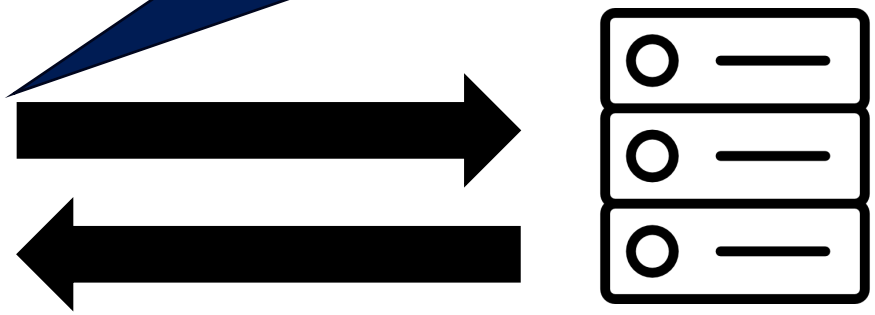


Referrer Header in Detail



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

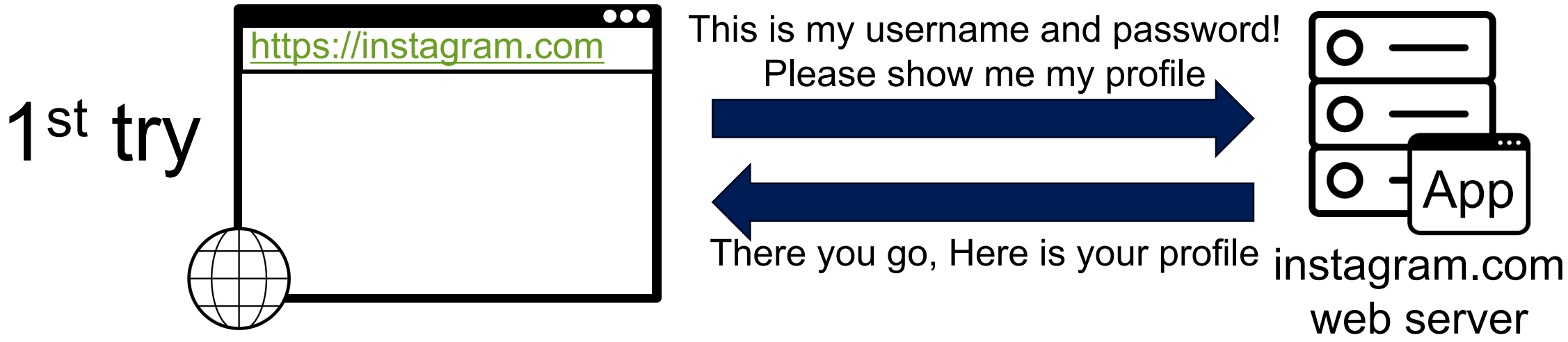
4xx Client Error

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

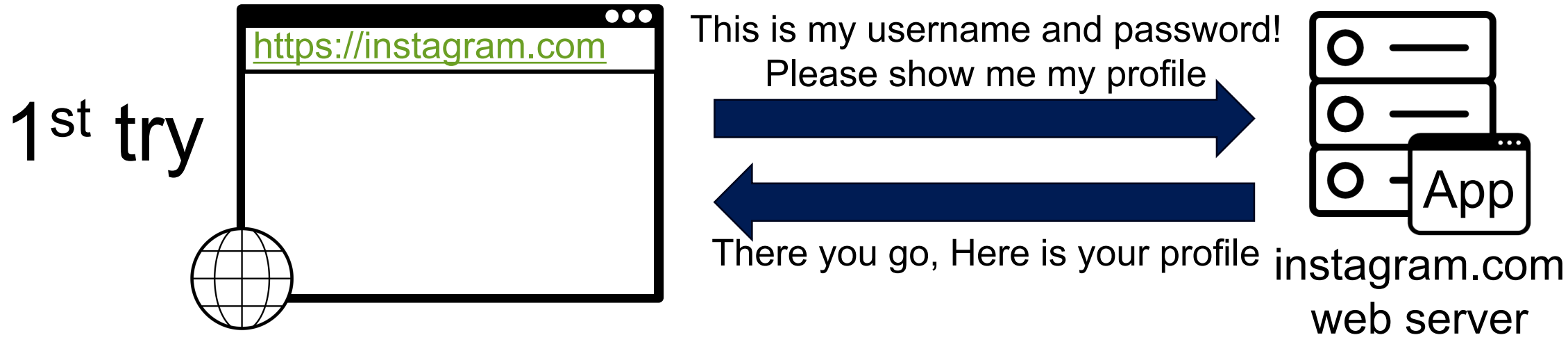
5xx Server Error

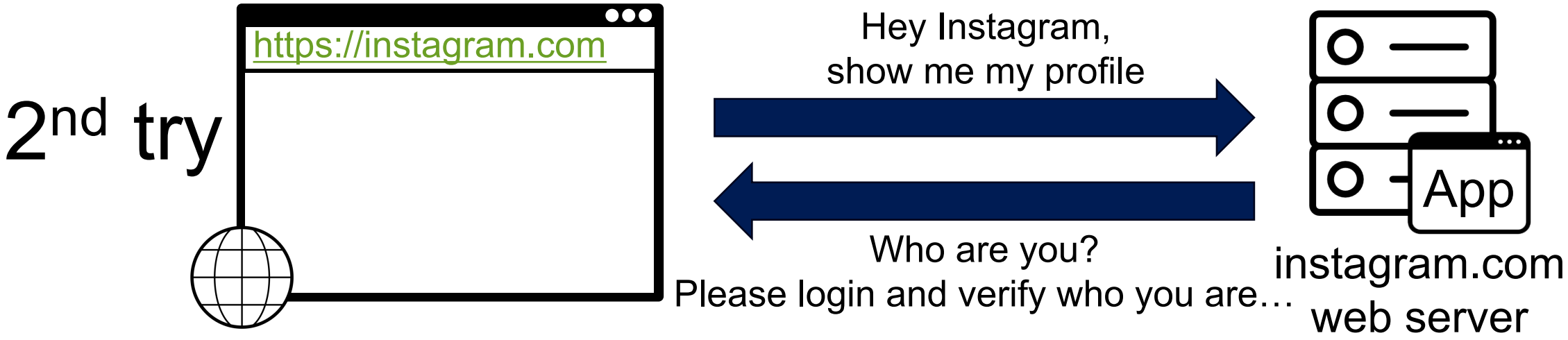
501	Not Implemented
502	Bad Gateway
503	Service Unavailable
504	Gateway Timeout

HTTP is a Stateless Protocol



HTTP is a Stateless Protocol





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?

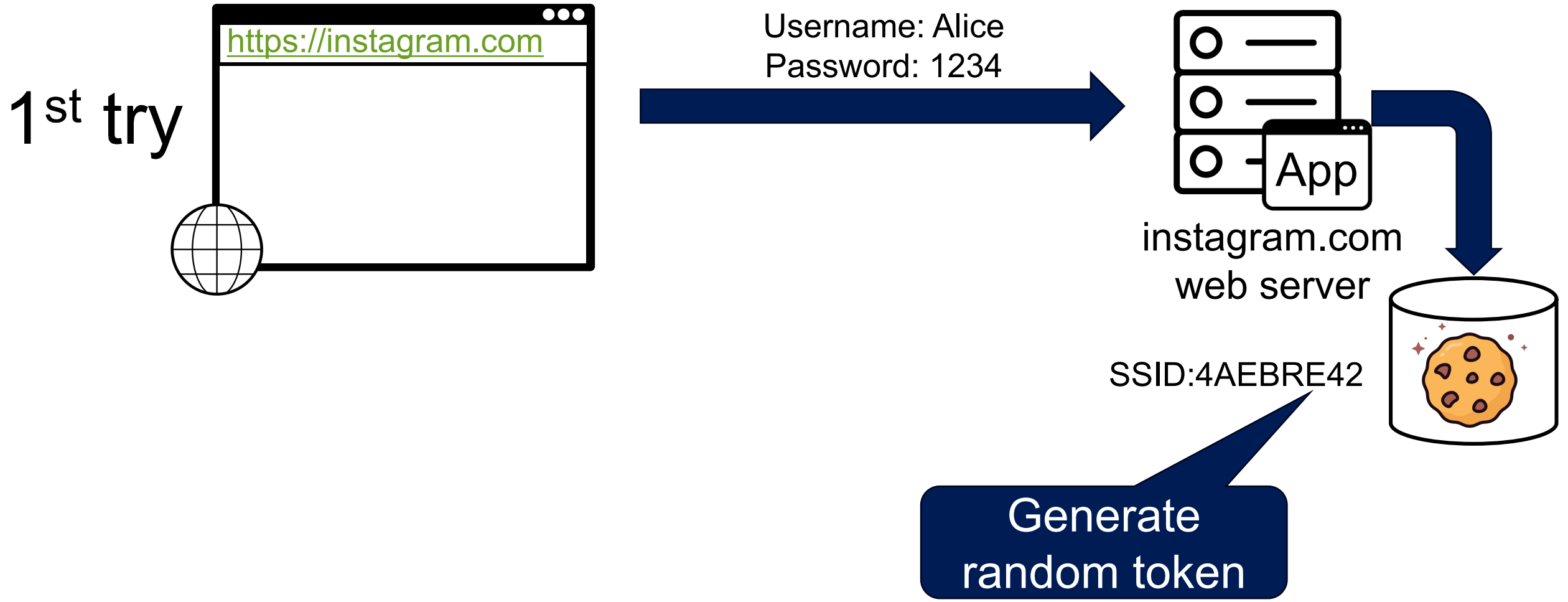
Adding State to HTTP

- 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”

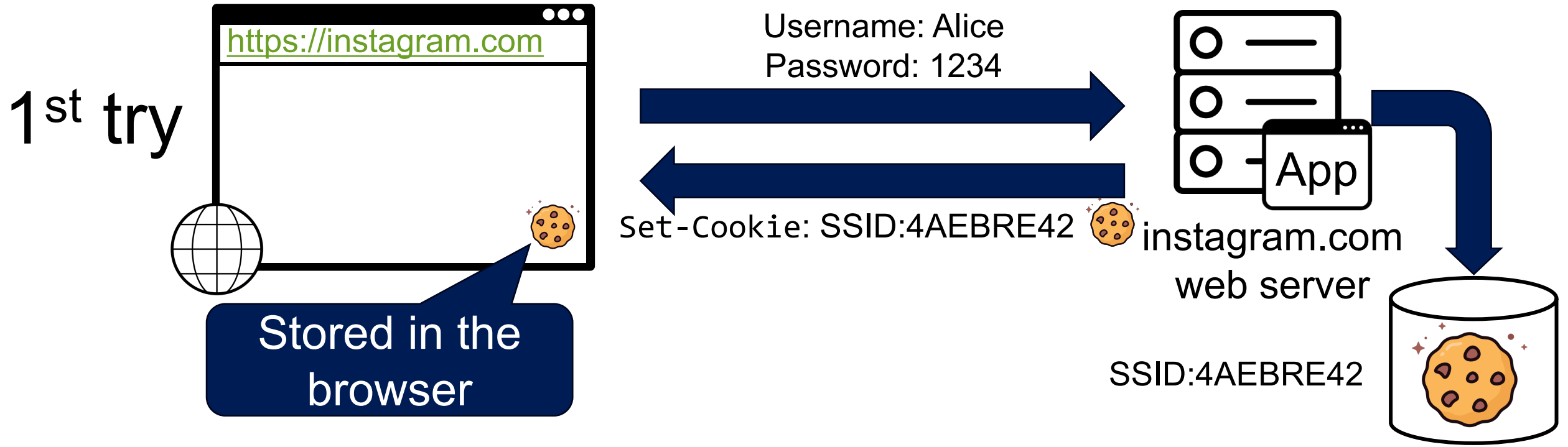
Cookie: Making HTTP Stateful

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

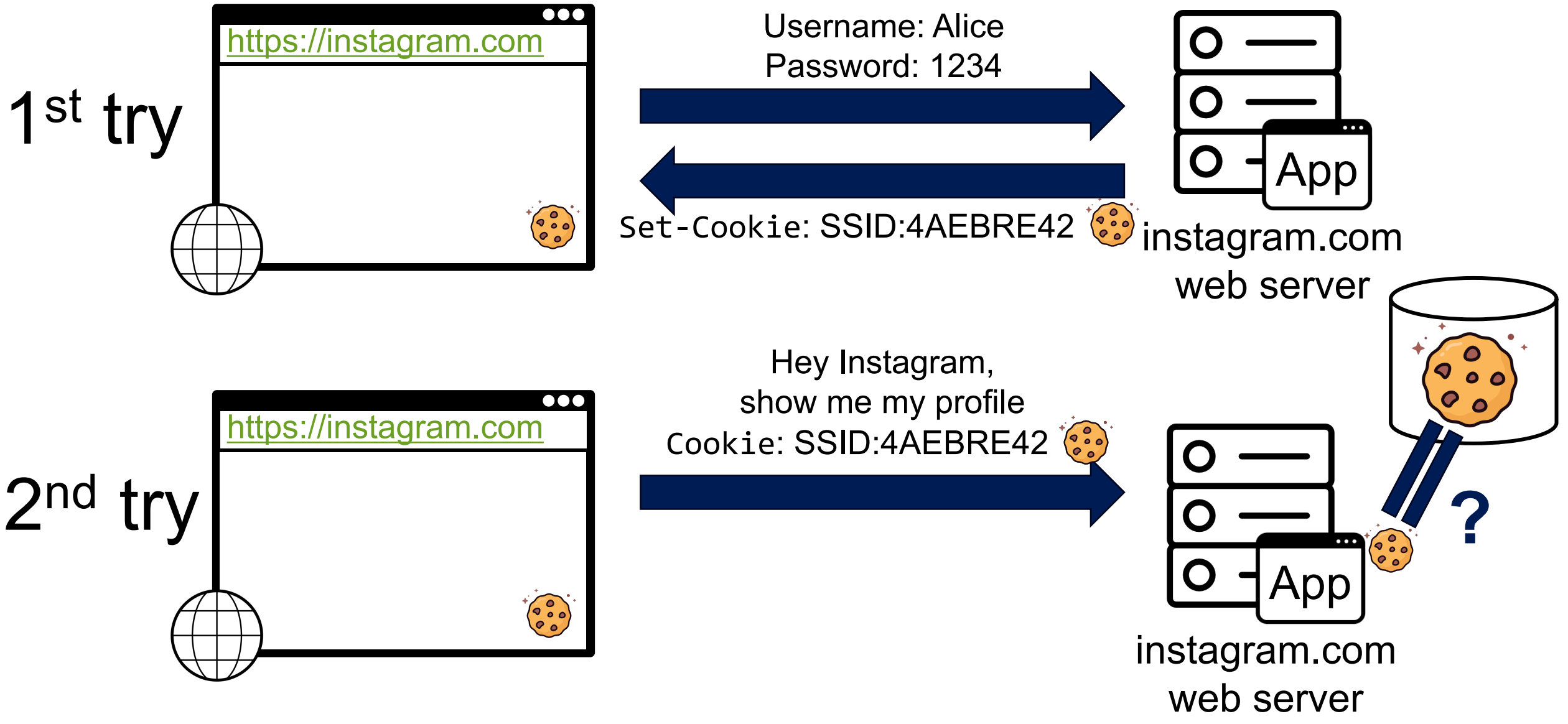
Cookie: Making HTTP Stateful



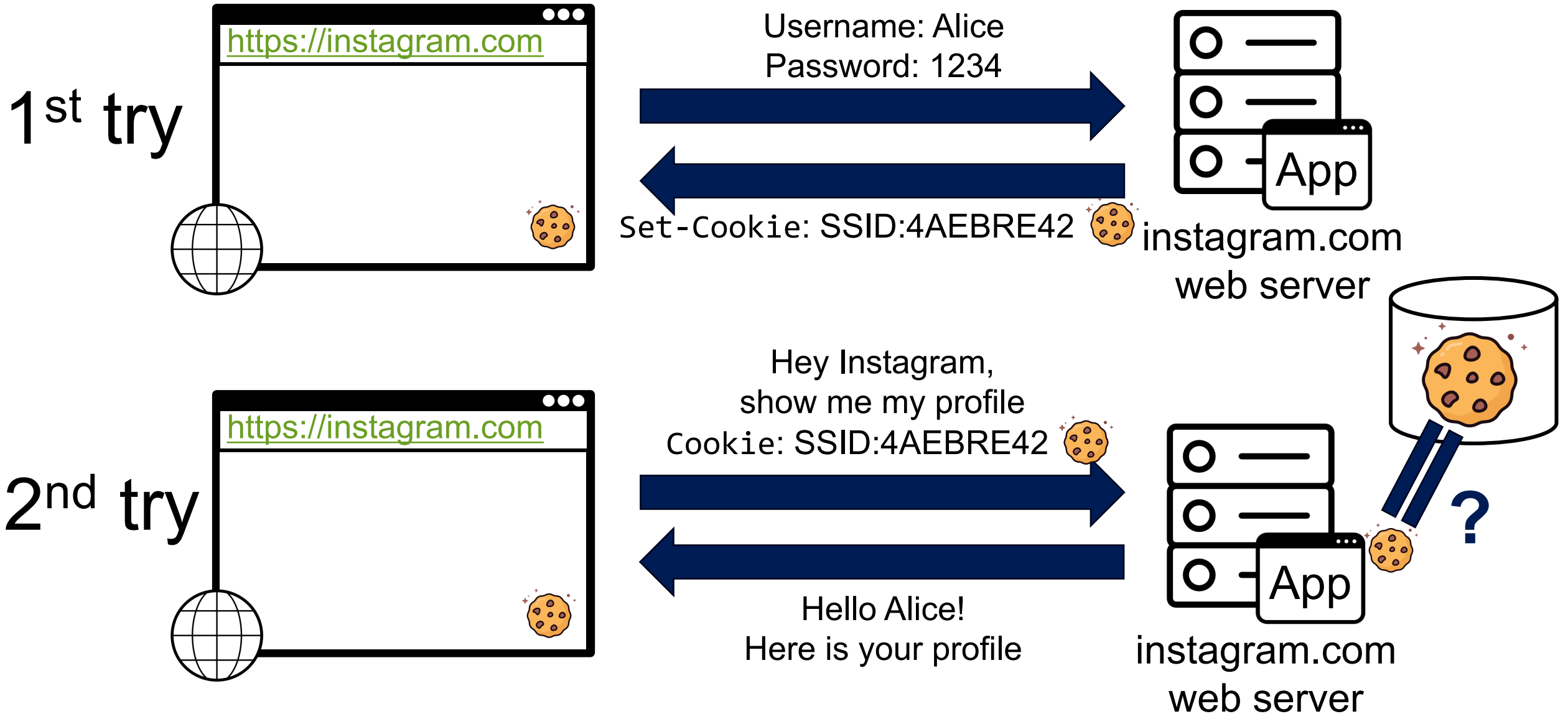
Cookie: Making HTTP Stateful



Cookie: Making HTTP Stateful



Cookie: Making HTTP Stateful



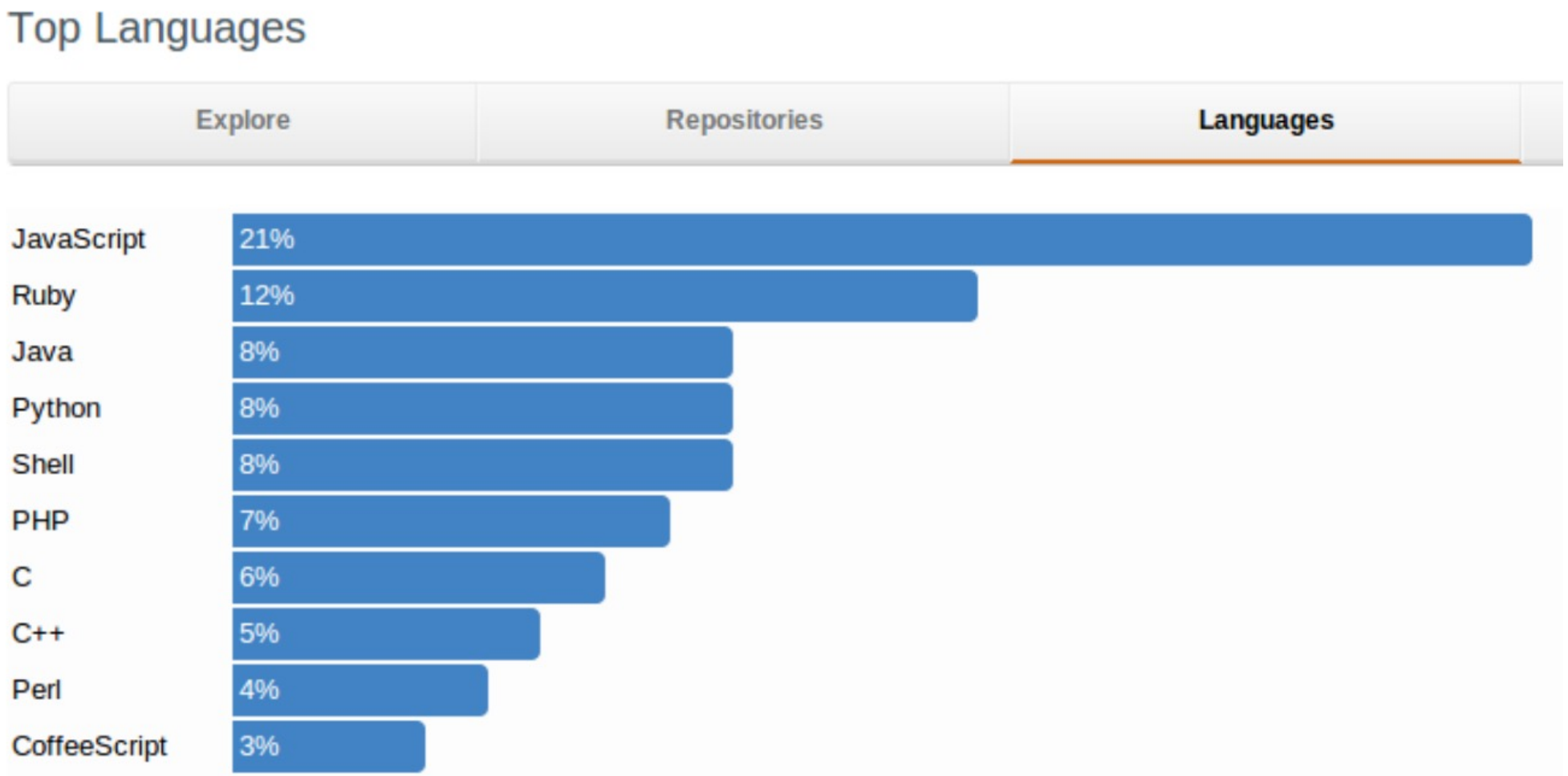
Cookie: Making HTTP Stateful

- 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)

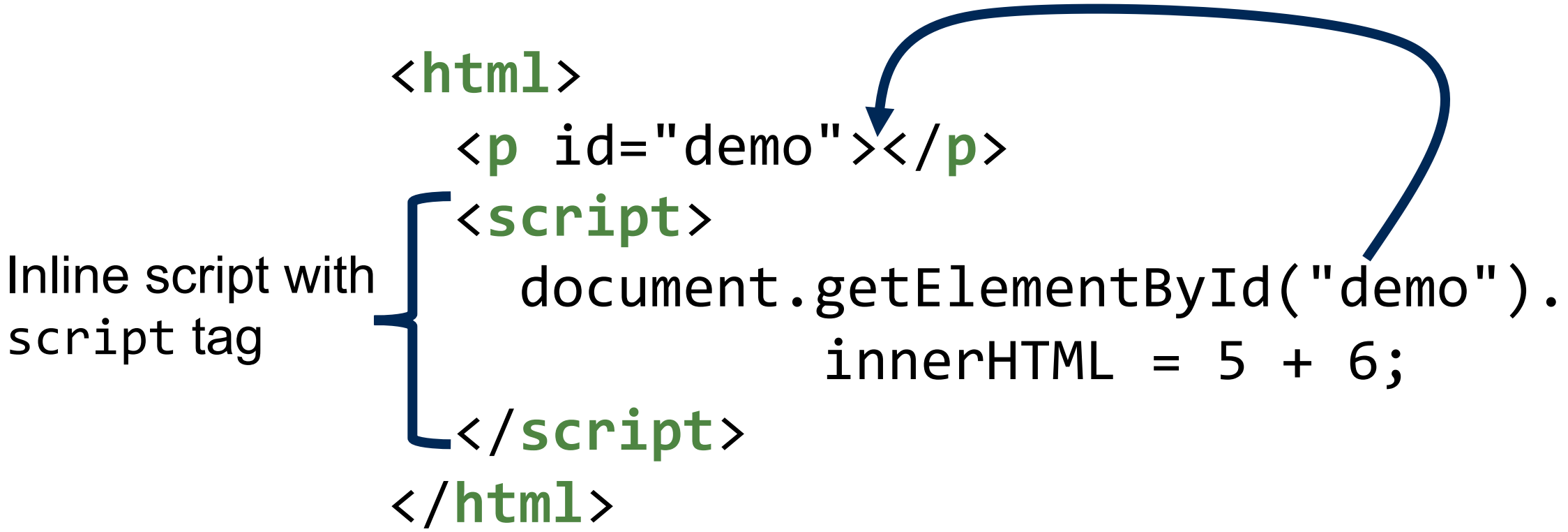
- HTML may contain JS program code to make web pages more dynamic

JS Example (1)


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

https://websec-lab.github.io/courses/2024s-cse610/demo/demo1.html

JS Example (1)



https://websec-lab.github.io/courses/2024s-cse610/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/2024s-cse610/demo/demo2.html

JS Example (2)

When the button is clicked, overwrite whole document with 11

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

Inline script with `onclick` event handler

https://websec-lab.github.io/courses/2024s-cse610/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/2024s-cse610/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)
```

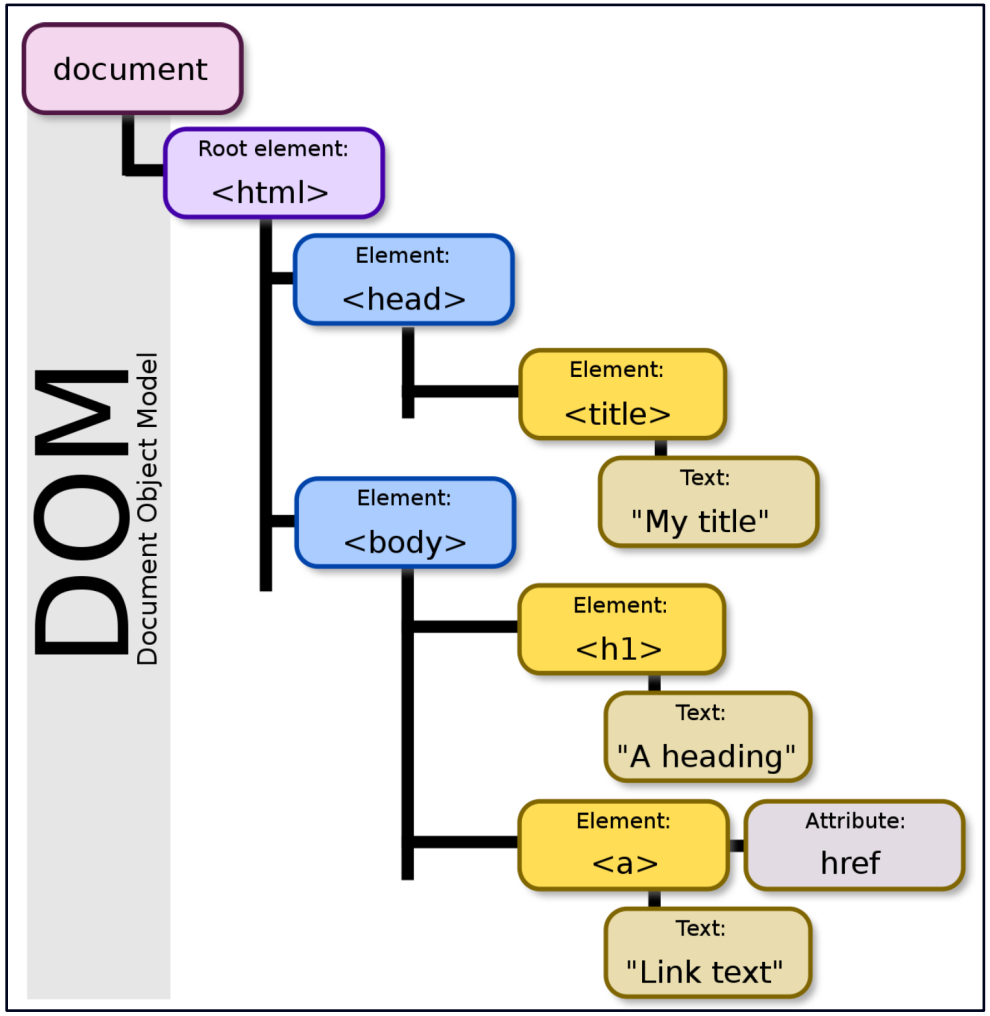
External script with `src` attribute

https://websec-lab.github.io/courses/2024s-cse610/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="cse610.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>
```

```
<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

Changing HTML DOM using JS (Example)

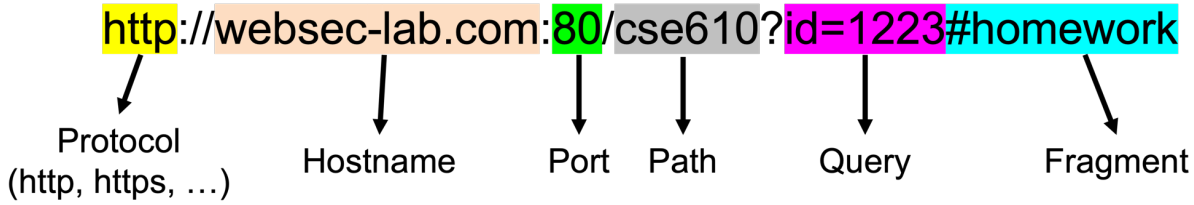

```
<html>
  <body>
    <ul id="t1">
      <li>Item 1</li>
    </ul>
…
  </body>
</html>
```

```
<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



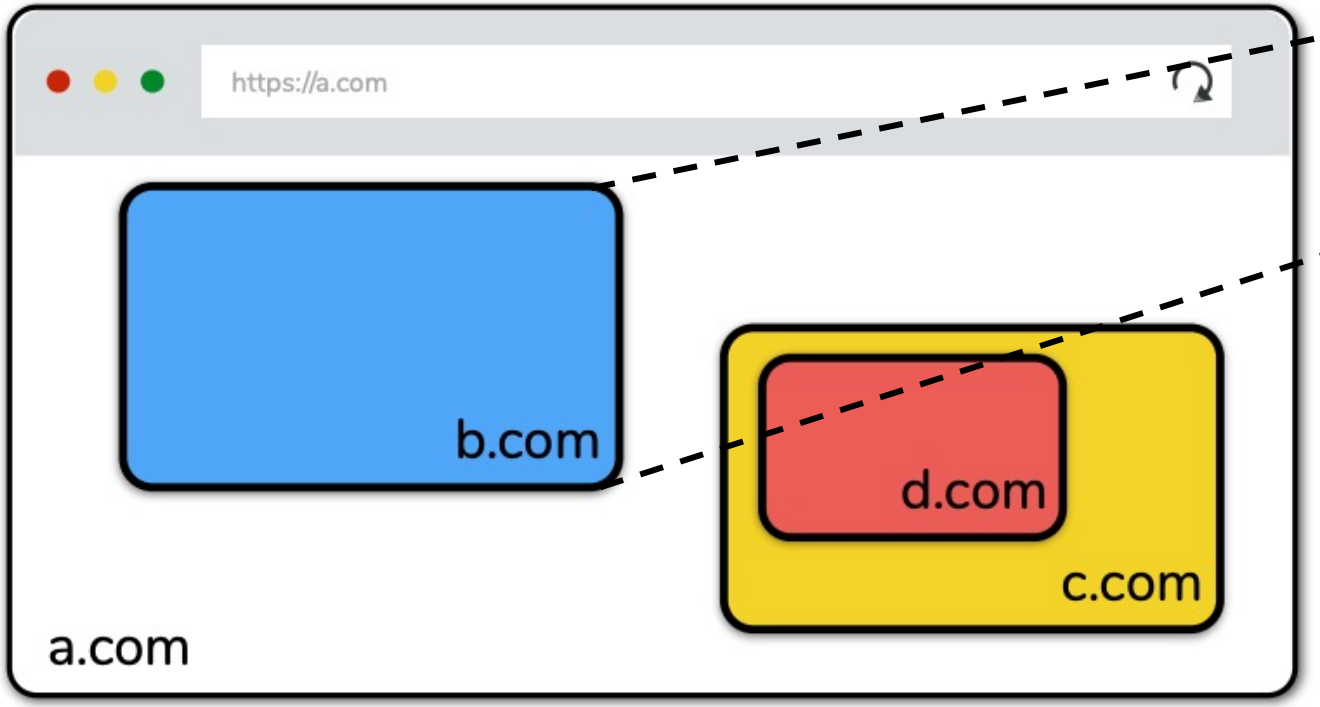
- We can display all cookies for current document by `alert(document.cookie)`

Basic Browser Execution Model

- Each browser window…
  - Loads content
  - Parses HTML and runs JavaScript
  - Fetches sub-resources (e.g., images, CSS, Javascript)
  - Respond to events like `onClick`, `onMouseover`, `onLoad`, `setTimeout`

Nested Execution Model

- Windows may contain frames from different sources
  - **Frame**: rigid visible division
  - **iFrame**: floating inline frame




```
<iframe src="b.com">
</iframe>
```

Nested Execution Model

- Windows may contain frames from different sources
  - **Frame**: rigid visible division
  - **iFrame**: floating inline frame

- Why use frames?
  - Delegate screen area to content from another source
  - Browser provides isolation based on frames
  - Parent may work even if frame is broken

Web Threat Models

- **Network attacker**
- **Remote attacker**
- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker**

- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker:** can connect to remote system via the network
  - Mostly targets the server

- **Web attacker**

Web Threat Models

- **Network attacker**: resides somewhere in the communication link between client and server
  - Passive: evasdropping
  - Active: modification of messages, replay…

- **Remote attacker:** can connect to remote system via the network
  - Mostly targets the server

- **Web attacker**: controls attacker.com
  - Can obtain SSL/TLS certificates for attacker.com
  - Users can visit attacker.com

Web Attacker

Victims can visit attacker's webpage

Web attacker can control of his webpage

Victim

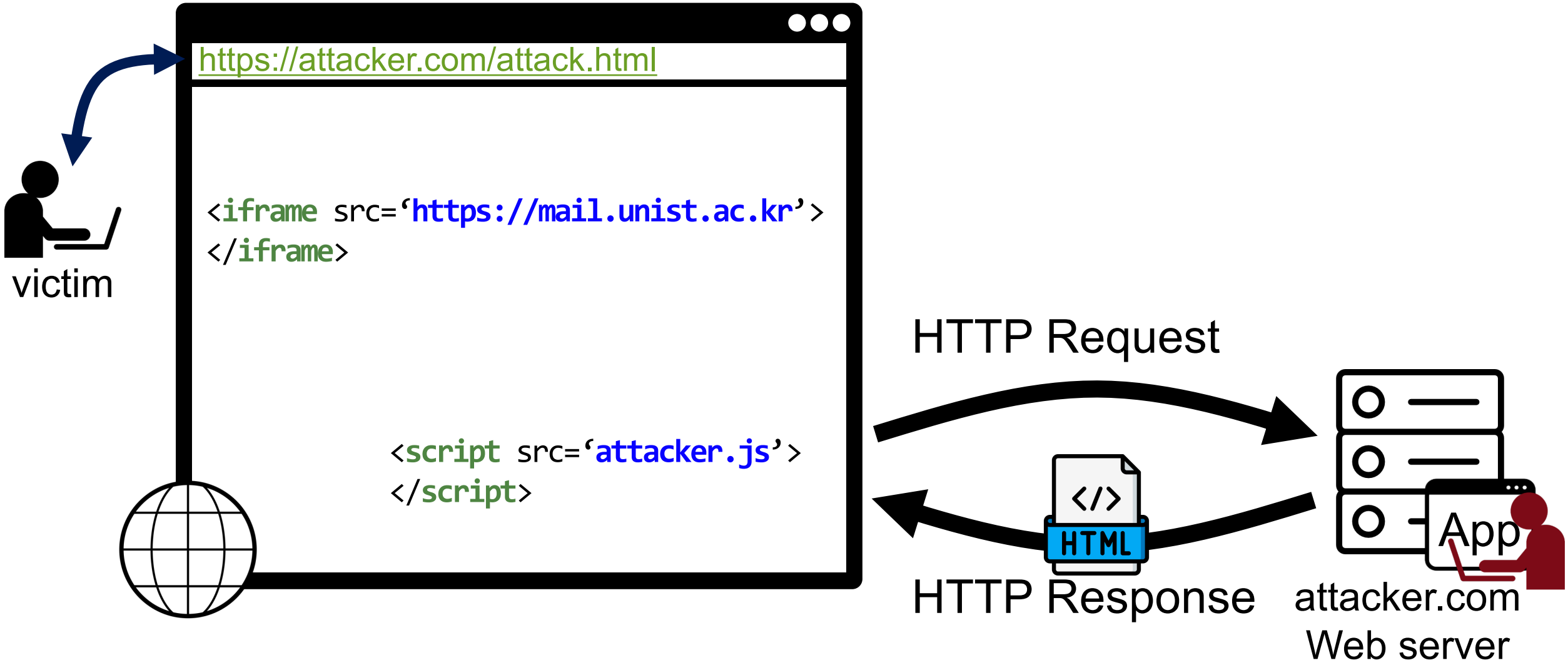
Web attacker

Question

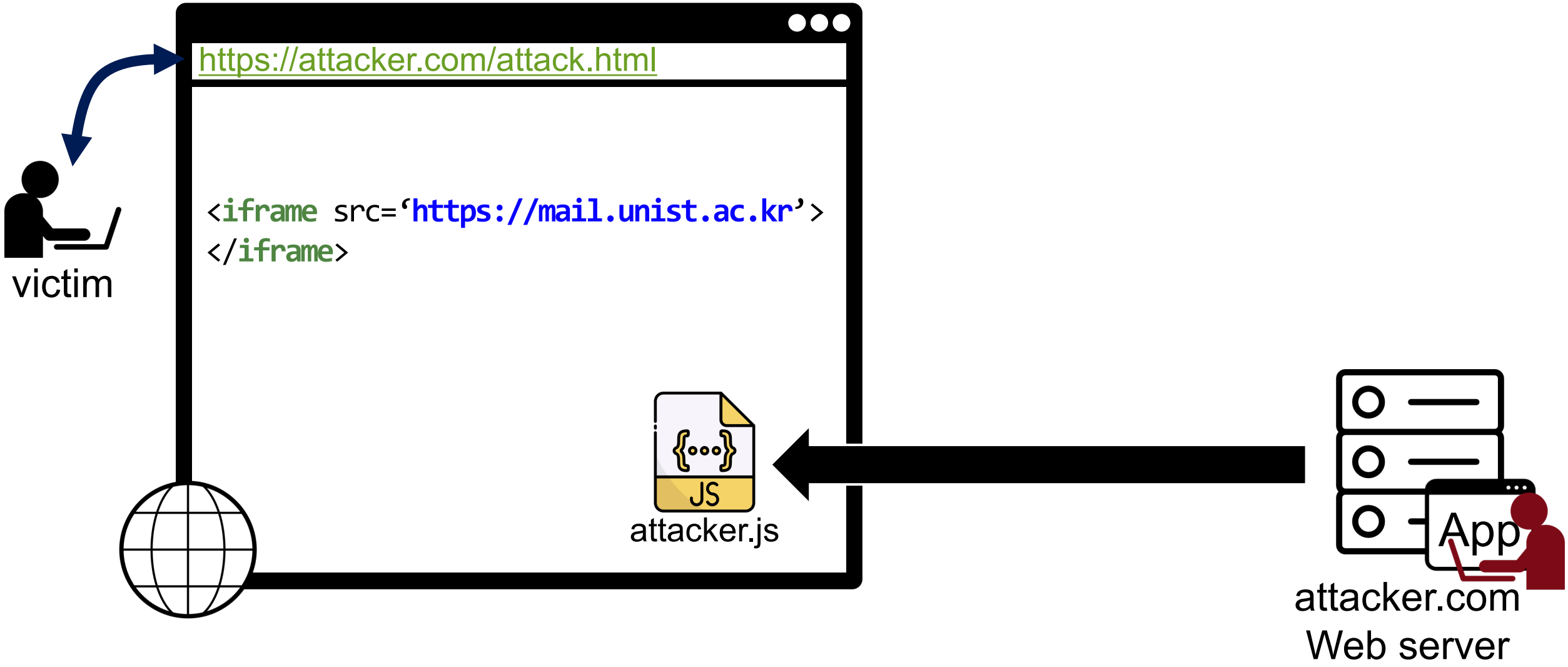
- Is the ***web attacker*** has a control on the victim's referrer header?

Client-side Security Basics

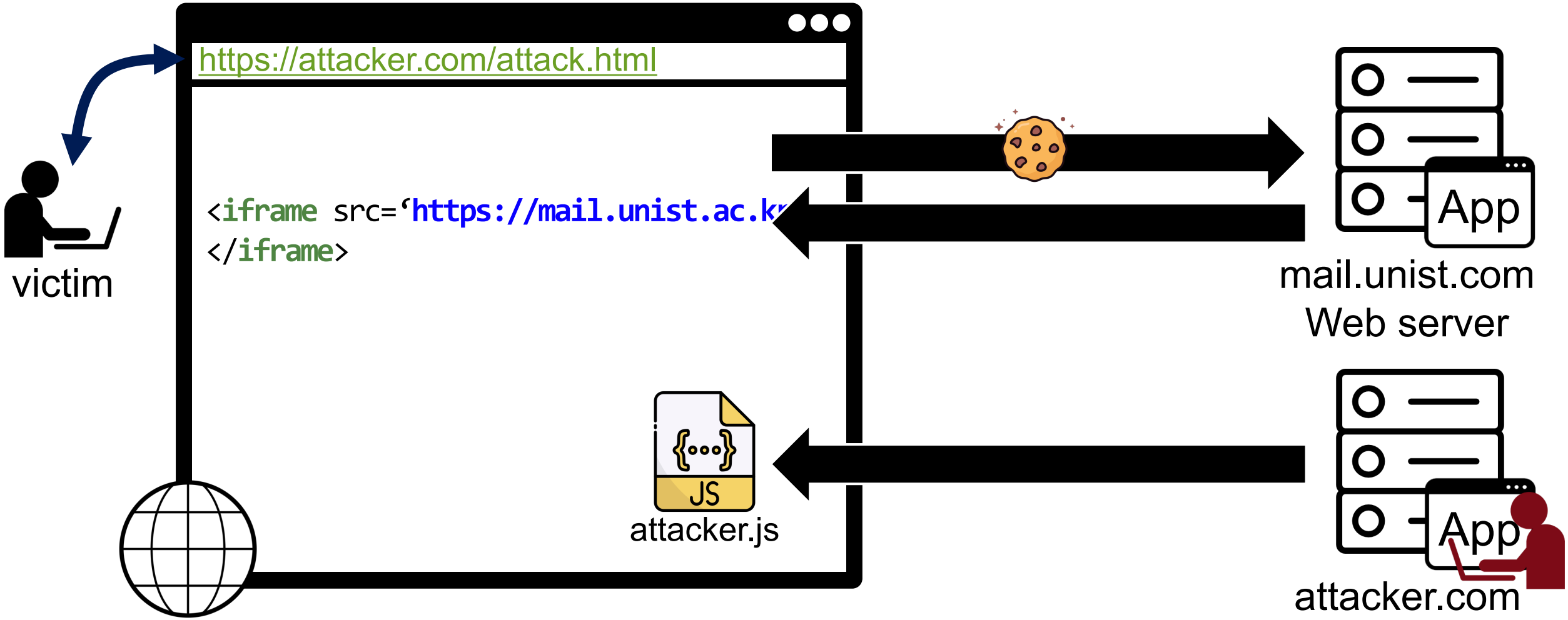
Motivation of the Client-side Security



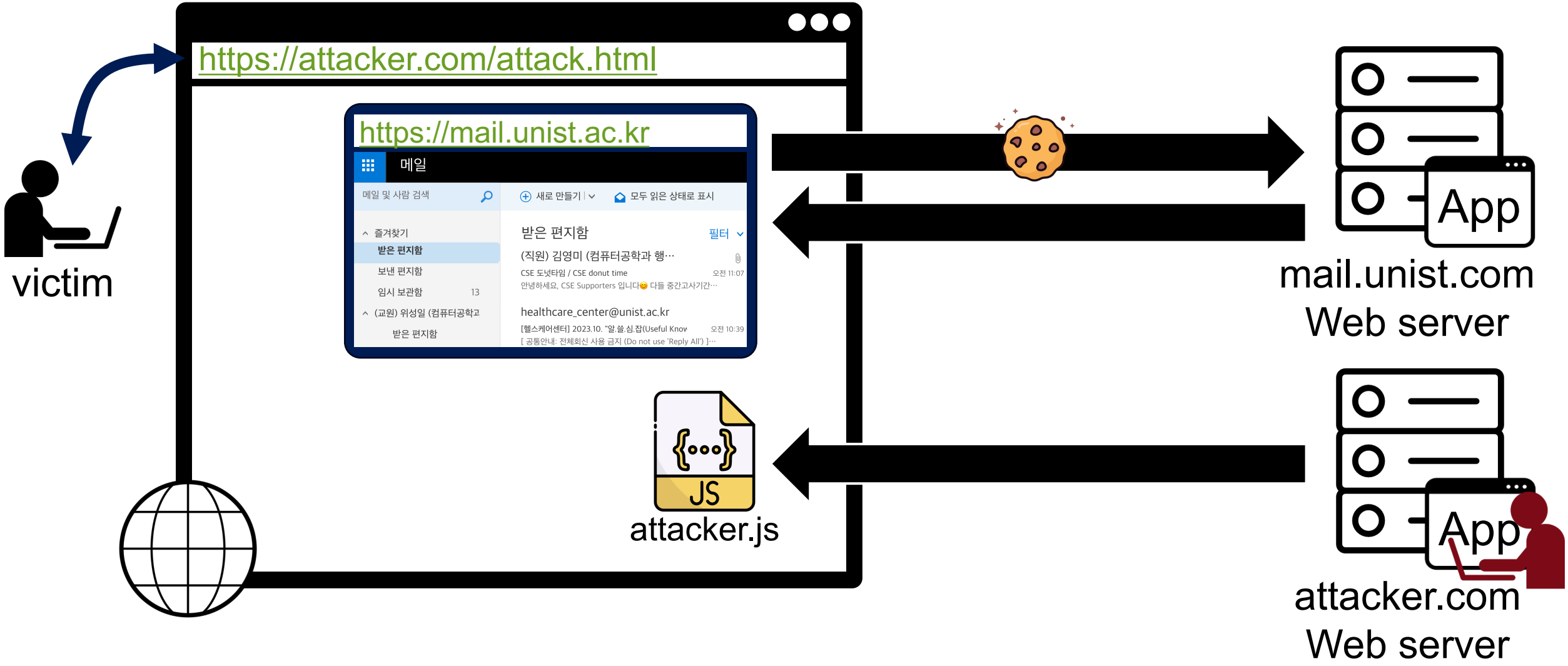
Motivation of the Client-side Security



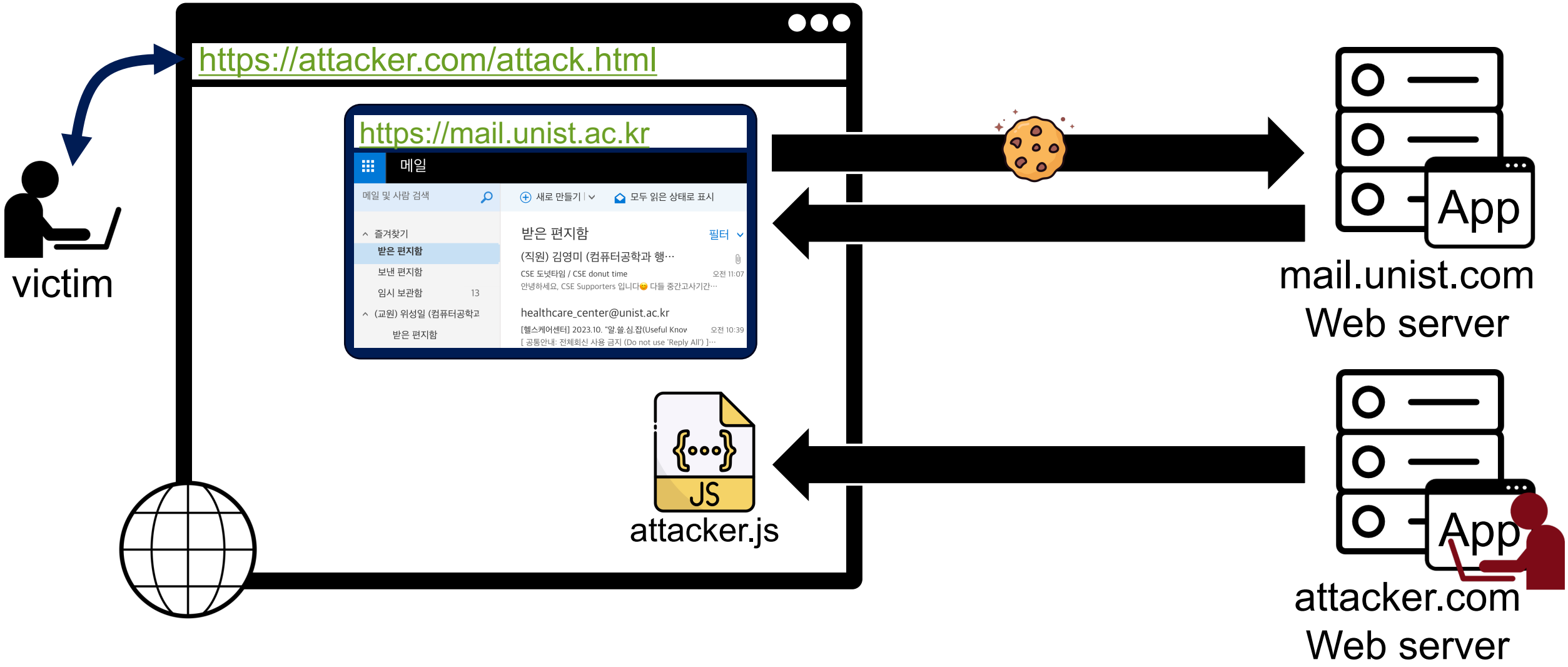
Motivation of the Client-side Security



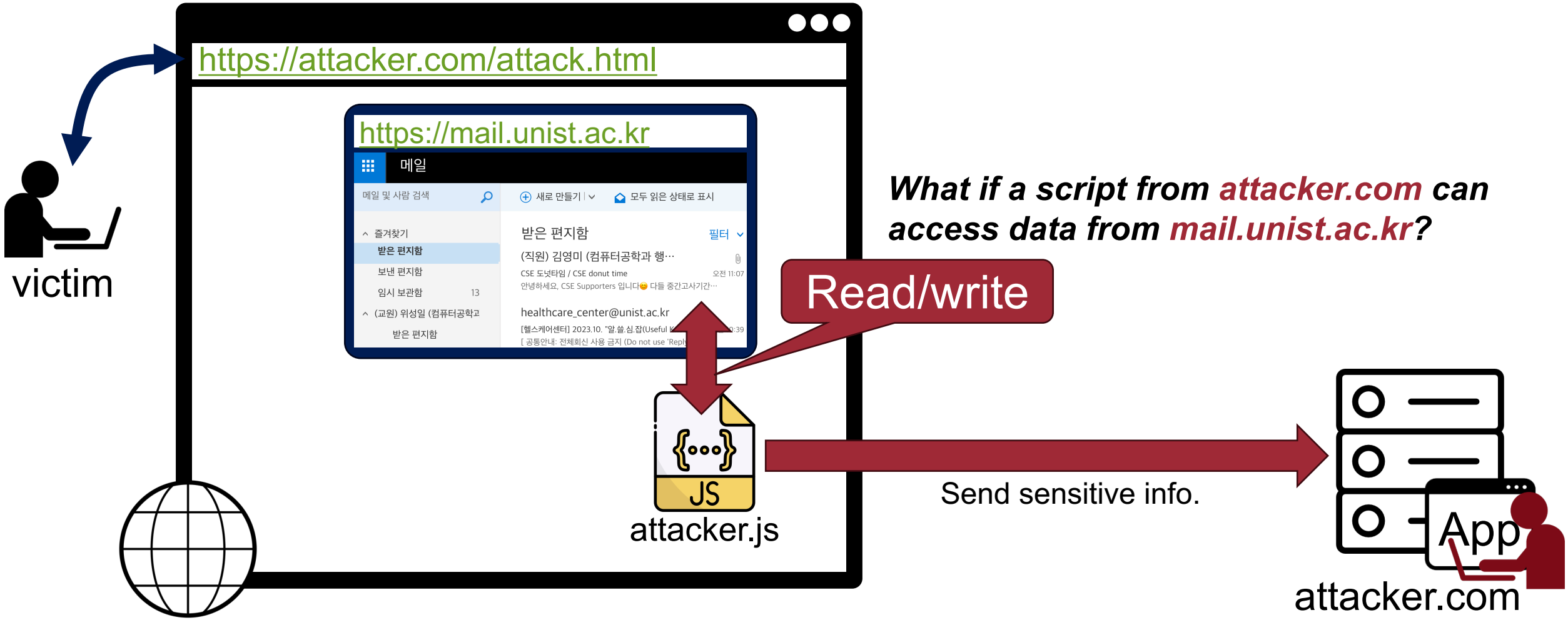
Motivation of the Client-side Security



Motivation of the Client-side Security



A World Without Separation between Sites



A World Without Separation between Sites

What if a script from attacker.com can access data from mail.unist.ac.kr?

It would be able to read your emails, private messages, authentication session cookies

Motivation of the Client-side Security

How can we prevent such malicious behaviors?

Question?