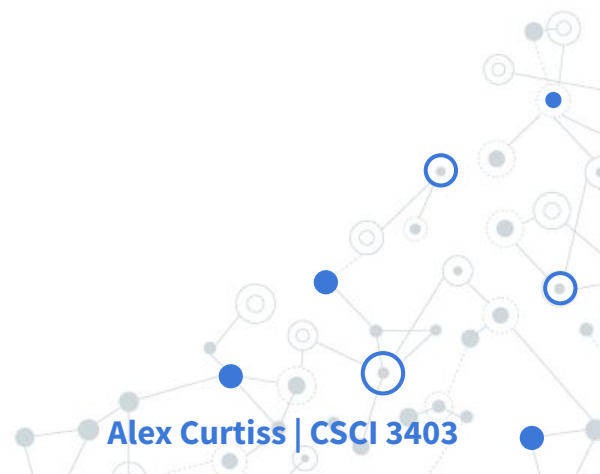




Web Headers and Cookies



Patch Notes

Last week's weekly exercise was due yesterday, this week's is available on Canvas

Python

Python data structures:

List (array)

```
users = ["alice", "bob", "carol"]  
print(users[1]) # Prints "bob"
```

Dictionary (hashtable)

```
user_passwords = {  
    "alice": "swordfish",  
    "bob": "hunter2",  
    "carol": "12345"  
}  
print(user_passwords["carol"]) # Prints "12345"
```

Web Fundamentals

Recap:

- Client code: Displays the webpage. Written in HTML.
- Server code: Responds to messages from the client.



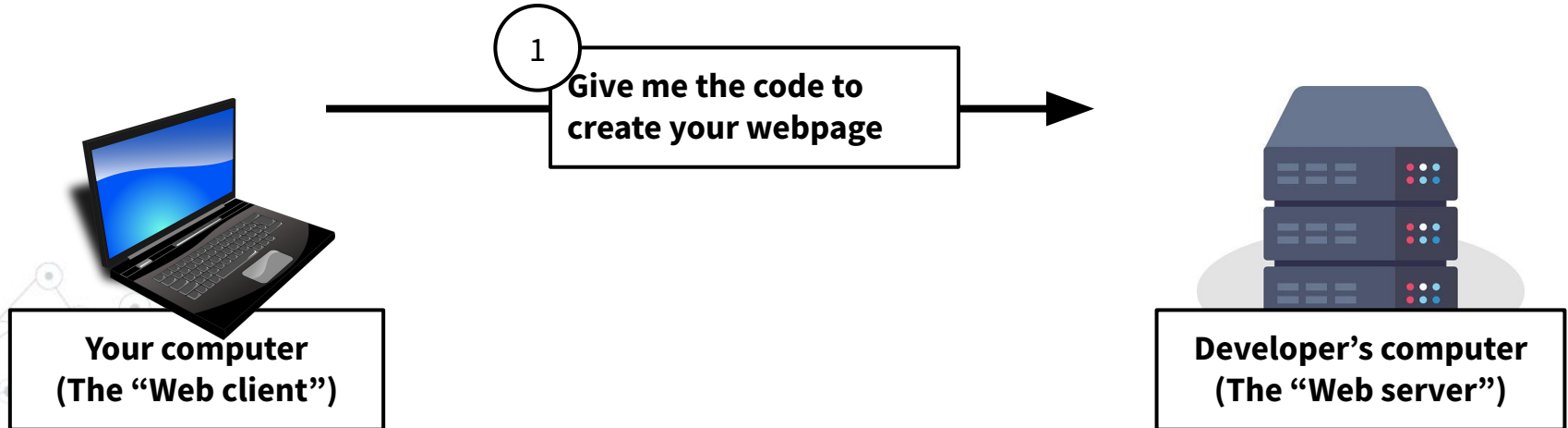
**Your computer
(The “Web client”)**



**Developer’s computer
(The “Web server”)**

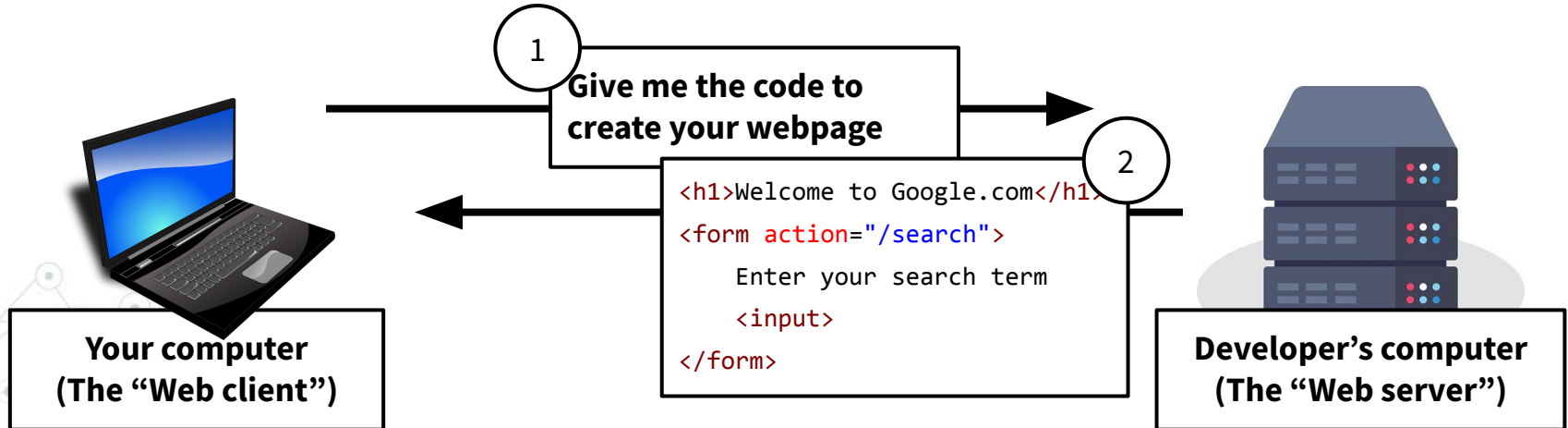
Web Fundamentals

1. Your computer (the “client”) asks for the website from the web developer’s computer (the “server”)



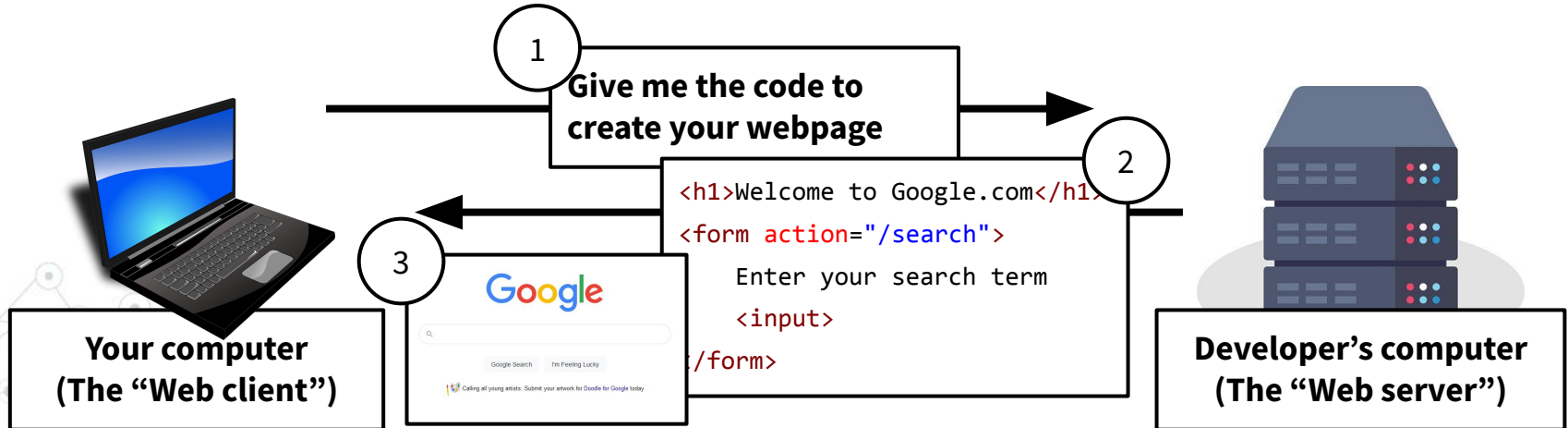
Web Fundamentals

1. Your computer (the “client”) asks for the website from the web developer’s computer (the “server”)
2. The server sends the code needed to create the website



Web Fundamentals

1. Your computer (the “client”) asks for the website from the web developer’s computer (the “server”)
2. The server sends the code needed to create the website
3. The client runs the code to display the website



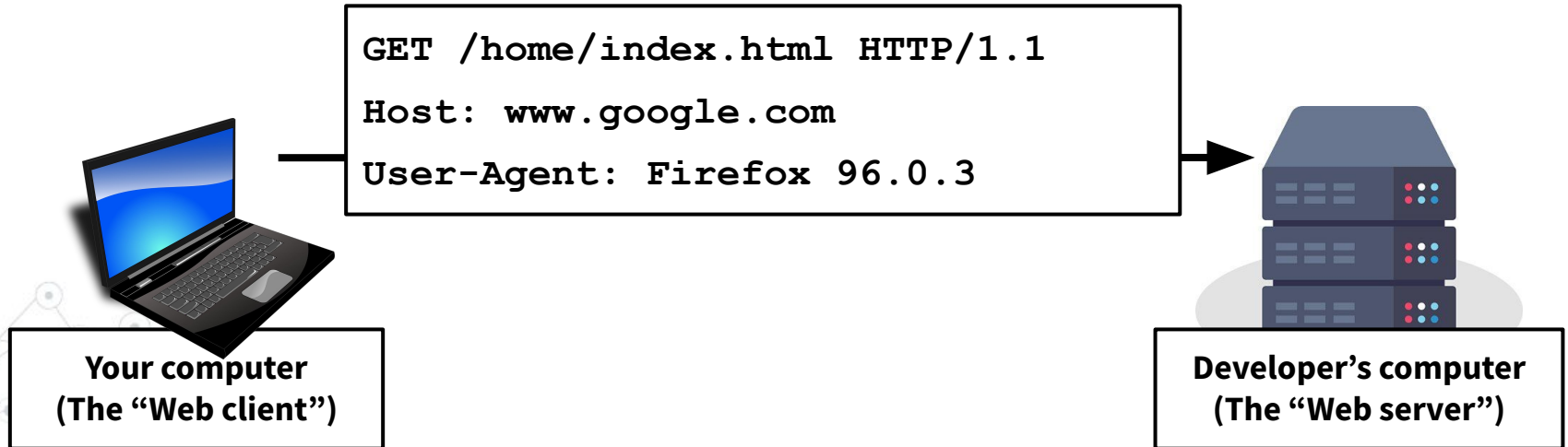
Web Fundamentals

Web request: A message sent from the client to the server, such as asking for the web page.



HTTP

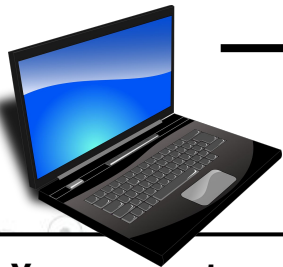
Web clients and server use a specific format, called **HyperText Transfer Protocol (HTTP)**.



HTTP

Path

```
GET /home/index.html HTTP/1.1  
Host: www.google.com  
User-Agent: Firefox 96.0.3
```



**Your computer
(The “Web client”)**



**Developer’s computer
(The “Web server”)**

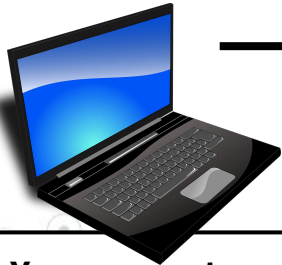
HTTP

Method

- **GET:** Get data
- **POST:** Send data

Path

```
GET /home/index.html HTTP/1.1  
Host: www.google.com  
User-Agent: Firefox 96.0.3
```



**Your computer
(The “Web client”)**



**Developer's computer
(The “Web server”)**

HTTP

Method

- **GET:** Get data
- **POST:** Send data

Path

POST /login HTTP/1.1

Host: www.google.com

User-Agent: Firefox 96.0.3

username:alex

Data (POST only)

Your computer
(The “Web client”)

Developer's computer
(The “Web server”)

HTTP

```
<form action="/login" method="post">  
  <input type="text" name="username" value="alex">  
</form>
```

```
POST /login HTTP/1.1
```

```
Host: www.google.com
```

```
User-Agent: Firefox 96.0.3
```

```
username:alex
```

HTTP

Headers: Optional metadata about the request, e.g.

- **User-Agent:** Which browser and OS you are running
- **Referer:** The previous website which linked to this one

```
POST /login HTTP/1.1
```

```
Host: www.google.com
```

```
User-Agent: Firefox 96.0.3
```

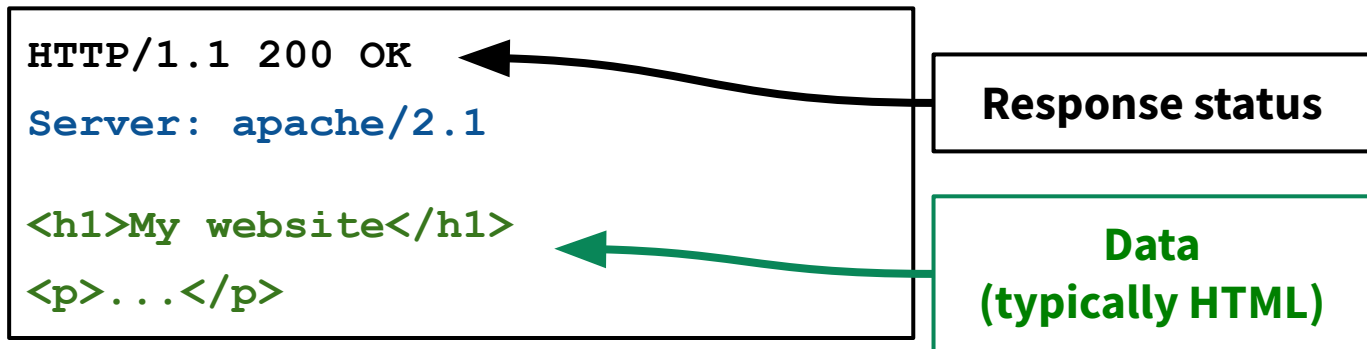
```
username:alex
```

} Headers

HTTP

The response from the server contains a “status”, data (often HTML), and different headers than the request, e.g.

- **Server:** The server language or framework



HTTP

The developer tools “network” tab shows HTTP requests:

The screenshot displays the Chrome DevTools Network tab. The top toolbar includes icons for Inspector, Network, Console, Debugger, Style Editor, Performance, Memory, Storage, Accessibility, and other tools. The Network tab is active, showing a list of requests. The selected request is a GET request to <https://www.colorado.edu/>. The details pane on the right shows the request headers and response headers.

St...	Met...	Domain	File	Initiator	Type	Trans...	S...
	POST	tr.snapchat.com	p	scevent.m...			
200	POST	mx.technolutions....	ping	ping:1 (be...	plain	712 B	2...
200	POST	mx.technolutions....	ping	ping:1 (be...	plain	712 B	2...
200	GET	www.colorado.edu	/	document	html	17.02...	6...
❌	POST	translate.googlea...	log?hasfast	m=el mai...	plain	NS_E...	0...
200	GET	www.colorado.edu	css_PAN7a...	stylesheet	css	7.09 ...	2...
200	GET	www.colorado.edu	css_zc2VkN	stylesheet	css	63.96...	6...
200	GET	fonts.googleapis.c...	css2?family	stylesheet	css	2.54 ...	2...
200	GET	www.colorado.edu	css_l4ZG2k...	stylesheet	css	10.22...	4...
200	GET	www.colorado.edu	css_SDUpU...	stylesheet	css	9.07 ...	4...
200	GET	www.colorado.edu	js_R4UkafM	script	js	996 B	4...
200	GET	www.colorado.edu	gtag.js?t0y...	script	js	1.34 ...	1...
200	GET	www.colorado.edu	gtag.js?t0ya	script	js	1.32 ...	1...
301	GET	cdn.colorado.edu	cu-boulder	img	svg	5.74 ...	1...
200	GET	translate.google.c...	element.js?	script	js	28.15...	7...
301	GET	cdn.colorado.edu	search-whit	img	svg	686 B	4...

Headers

Filter Headers

GET https://www.colorado.edu/

Status: 200

Version: HTTP/2

Transferred: 17.02 kB (68.50 kB size)

Request Priority: Highest

DNS Resolution: System

Response Headers (955 B)

Raw

- accept-ranges: bytes
- age: 72255
- cache-control: max-age=86400, public
- content-encoding: gzip
- content-language: en
- content-length: 16065
- content-type: text/html; charset=UTF-8
- date: Mon, 01 Sep 2025 14:37:47 GMT
- etag: W/"1756665212"
- expires: Sun, 19 Nov 1978 05:00:00 GMT

HTTP

The developer tools “network” tab shows HTTP requests:

The screenshot shows the Chrome DevTools Network tab. The left pane displays a list of requests, and the right pane shows the details for the selected GET request to `https://www.colorado.edu/`.

List of requests (Left Pane):

St...	Met...	Domain	File	Initiator	Type	Trans...	S...
	POST	tr.snapchat.com	p	scevent.m...			
200	POST	mx.technolutions....	ping	ping:1 (be...	plain	712 B	2...
200	POST	mx.technolutions....	ping	ping:1 (be...	plain	712 B	2...
200	GET	www.colorado.edu	/	document	html	17.02...	6...
400	POST	translate.googlea...	log?hasfast	m=el.maj...	plain	NS_E...	0...
200	GET	www.colorado.edu	css_PAN7a5	stylesheet	css	7.09 ...	2...
200	GET	www.colorado.edu	css_zc2VkN	stylesheet	css	63.96...	6...
200	GET	fonts.googleapis.c...	css2?family	stylesheet	css	2.54 ...	2...
200	GET	www.colorado.edu	css_l4ZG2k	stylesheet	css	10.22...	4...
200	GET	www.colorado.edu	css_SDUpUj	stylesheet	css	9.07 ...	4...
200	GET	www.colorado.edu	js_R4UkafM	script	js	996 B	4...
200	GET	www.colorado.edu	gtag.js?t0y	script	js	1.34 ...	1...
200	GET	www.colorado.edu	gtag.js?t0ya	script	js	1.32 ...	1...
301	GET	cdn.colorado.edu	cu-boulder	img	svg	5.74 ...	1...
200	GET	translate.google.c...	element.js?	script	js	28.15...	7...
301	GET	cdn.colorado.edu	search-whit	img	svg	686 B	4...

Request Details (Right Pane):

Headers:

- Status: 200
- Version: HTTP/2
- Transferred: 17.02 kB (68.50 kB size)
- Request Priority: Highest
- DNS Resolution: System

Response Headers (955 B):

- accept-ranges: bytes
- age: 72255
- cache-control: max-age=86400, public
- content-encoding: gzip
- content-language: en
- content-length: 16065
- content-type: text/html; charset=UTF-8
- date: Mon, 01 Sep 2025 14:37:47 GMT
- etag: W/"1756665212"
- expires: Sun, 19 Nov 1978 05:00:00 GMT

List of requests

Data (in separate tabs)

Request method and path

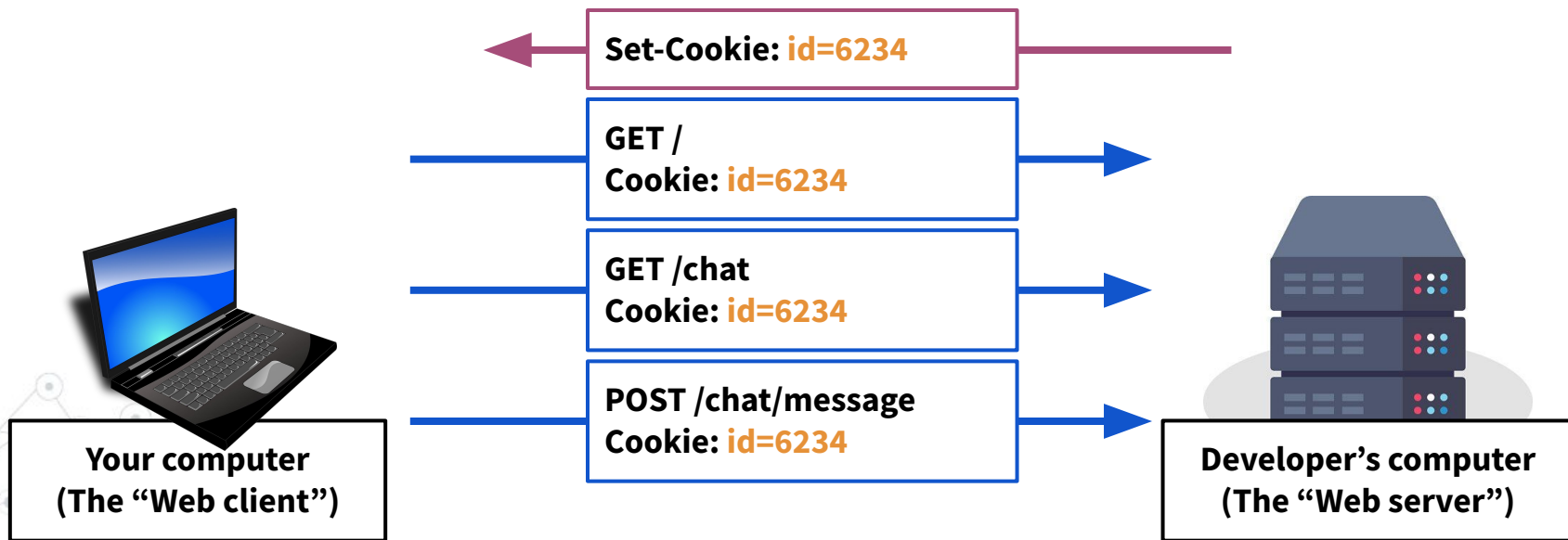
Headers



Cookies

Cookies

Cookies: Header values which are saved by the browser and sent back to the server with each request





[Demo: Login with cookies]

Server-side code

Login with cookies demo code:

```
from flask import Flask, redirect, render_template, request

app = Flask(__name__)

@app.route("/")
def index():
    current_user = request.cookies.get("logged_in_as")
    return render_template("index.html", current_user=current_user)

@app.route("/login", methods=["GET"])
def login_get():
    return render_template("login.html")

@app.route("/login", methods=["POST"])
def login_post():
    username = request.form["username"]
    password = request.form["password"]

    if username == "alex" and password == "swordfish":
        response = redirect("/")
        response.set_cookie("logged_in_as", "alex")
        return response

app.run(debug=True)
```

```
<!-- index.html -->
<h1>Example Site</h1>

<p>Logged in as: {{ user }}</p>

<a href="/login">Login</a>

<!-- login.html -->
<h1>Example Site</h1>

<form action="/login" method="post">
    Username:
    <input type="text" name="username">

    Password:
    <input type="password" name="password">

    <input type="submit">
</form>

<form action="/login" method="post">
    <input type="text" name="username">
</form>
```



What is wrong with that approach?

Cookies

Cookies are often used for authentication. This makes them just as valuable as passwords!

Cookies

Cookies are often used for authentication. This makes them just as valuable as passwords!

- **Terribly insecure:** Cookie is just the username
- **Better:** Cookie is a long, random string and the server keeps a mapping of cookie/username values

```
{  
  "7ce5a141-6431-4d83-8913-25447d35e7a0": "alex",  
  "d4622f93-59b2-4a49-a104-39af77a01cc0": "admin",  
}
```




[Demo: Secure cookies]

Cookies

Secure cookies demo code:

```
import random
from flask import Flask, redirect, render_template, request

app = Flask(__name__)

cookie_map = {}

@app.route("/")
def index():
    cookie = request.cookies.get("logged_in_as")
    current_user = cookie_map.get(cookie)

    return render_template("index.html", current_user=current_user)

@app.route("/login", methods=["GET"])
def login_get():
    return render_template("login.html")

@app.route("/login", methods=["POST"])
def login_post():
    username = request.form["username"]
    password = request.form["password"]

    if username == "alex" and password == "swordfish":
        cookie = random.randbytes(20).hex()
        cookie_map[cookie] = "alex"

        response = redirect("/")
        response.set_cookie("logged_in_as", cookie)
        return response

app.run(debug=True)
```

```
<!-- index.html -->
<h1>Example Site</h1>

<p>Logged in as: {{ user }}</p>

<a href="/login">Login</a>

<!-- login.html -->
<h1>Example Site</h1>

<form action="/login" method="post">
    Username:
    <input type="text" name="username">

    Password:
    <input type="password" name="password">

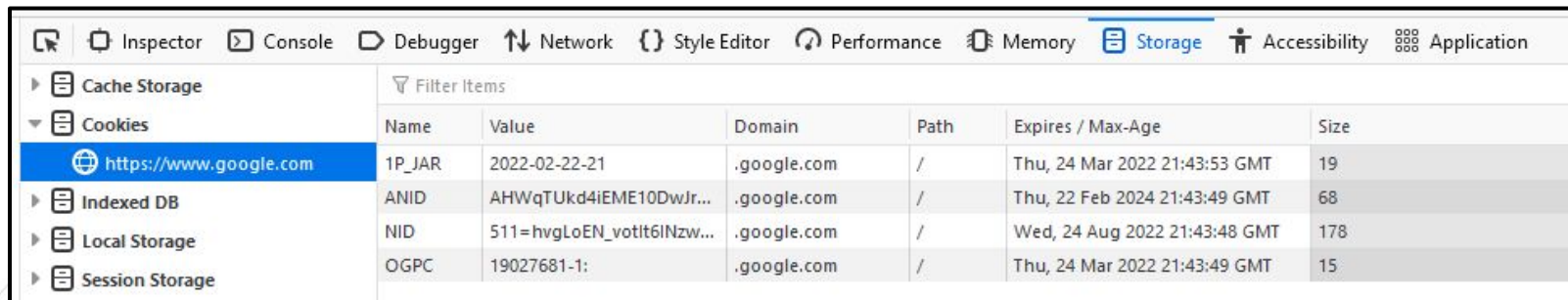
    <input type="submit">
</form>

<form action="/login" method="post">
    <input type="text" name="username">
</form>
```

Cookies

Even if cookies are long and random, they can still be stolen off either the client or the server.

- **Example:** Somebody with physical access to your computer can open the cookie menu and read them



The screenshot shows the Chrome DevTools Storage panel. The left sidebar lists storage types: Cache Storage, Cookies, Indexed DB, Local Storage, and Session Storage. The 'Cookies' section is expanded, and a specific cookie for 'https://www.google.com' is selected. The main panel displays a table of cookies for this domain.

Name	Value	Domain	Path	Expires / Max-Age	Size
1P_JAR	2022-02-22-21	.google.com	/	Thu, 24 Mar 2022 21:43:53 GMT	19
ANID	AHWqTUKd4iEME10DwJr...	.google.com	/	Thu, 22 Feb 2024 21:43:49 GMT	68
NID	511=hvgLoEN_votit6INzw...	.google.com	/	Wed, 24 Aug 2022 21:43:48 GMT	178
OGPC	19027681-1:	.google.com	/	Thu, 24 Mar 2022 21:43:49 GMT	15



[Demo: Stealing a cookie]



The lecture is over go home