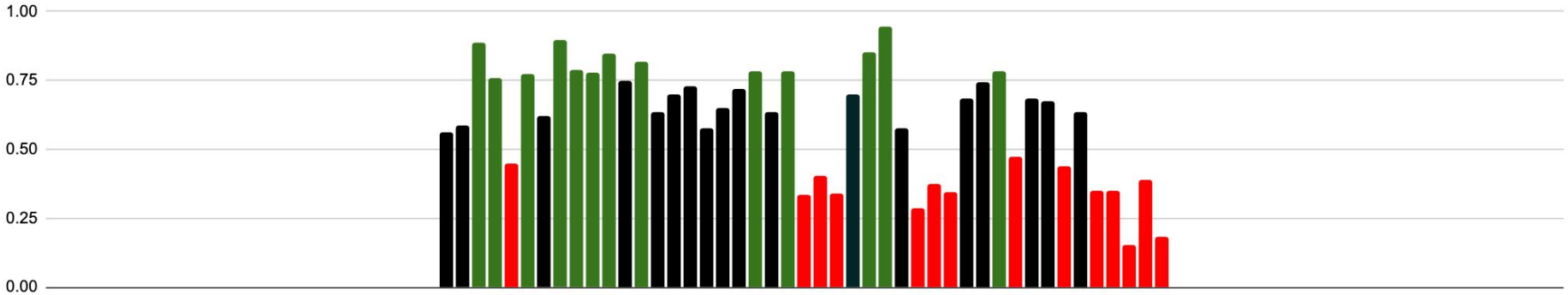
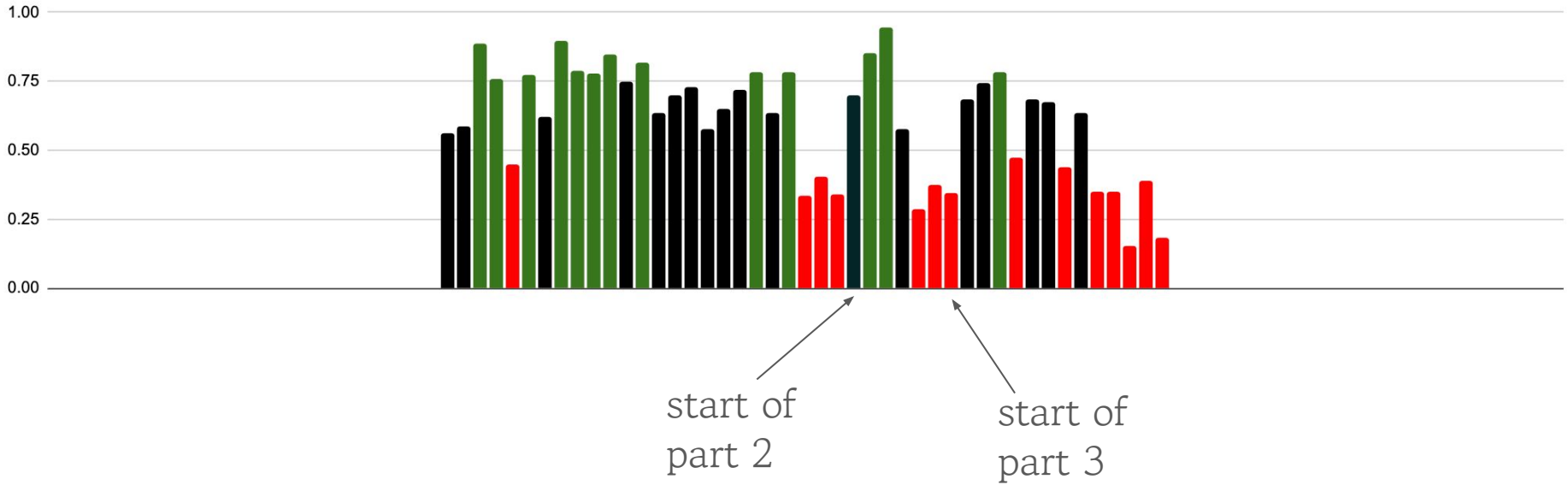
Midterm Recap

Ad Targeting and Surveillance Pricing

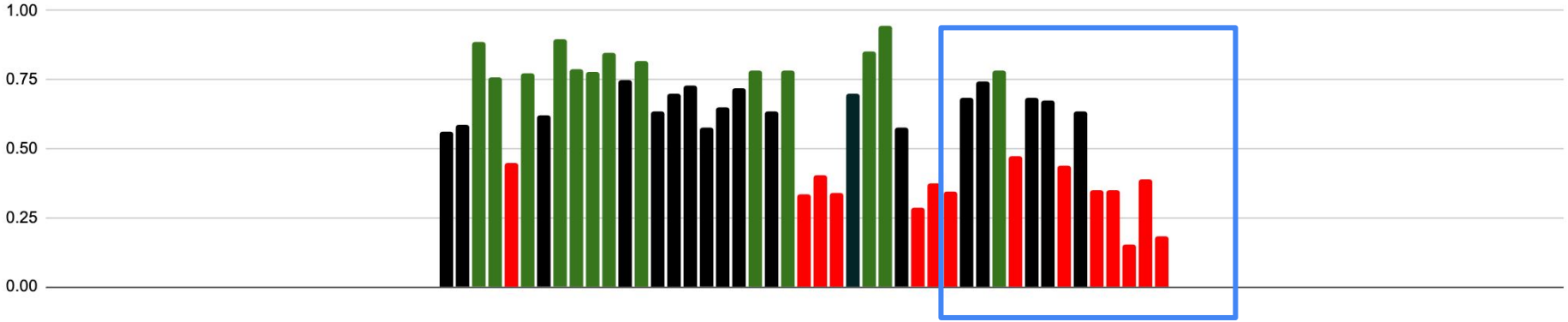
UC San Diego



190 (undergraduate): 52.0 (58%)
291 (graduate): 59.8 (66%)



Leave yourself time for all parts of the exam!



Students who did not use LLMs for homework earned ~5 more points on average.

1. (2 points) Which of the following are examples of network protocols?

- ◯ HTTP
- ◯ Ethernet
- ◯ TCP
- ◯ DNS
- ◯ Flask

Strategy: don't guess if you're not sure!
False positives penalized (1 pt each) more than many
Correct positives earn (m/n pt each)

1. (2 points) Which of the following are examples of network protocols?

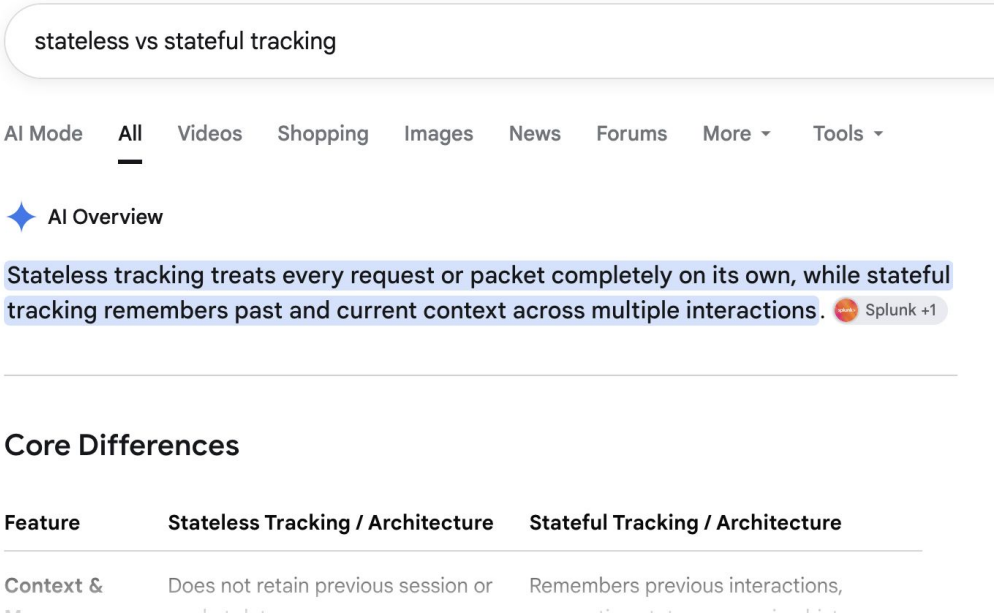
- ◯ HTTP
- ◯ Ethernet
- ◯ TCP
- ◯ DNS
- ◯ Flask

m = # points
n = # correct answers

$$\frac{\text{2 points}}{\text{4 correct answers}}$$

Each correct answer earns + ½ point
Each incorrect answer earns - 1 point

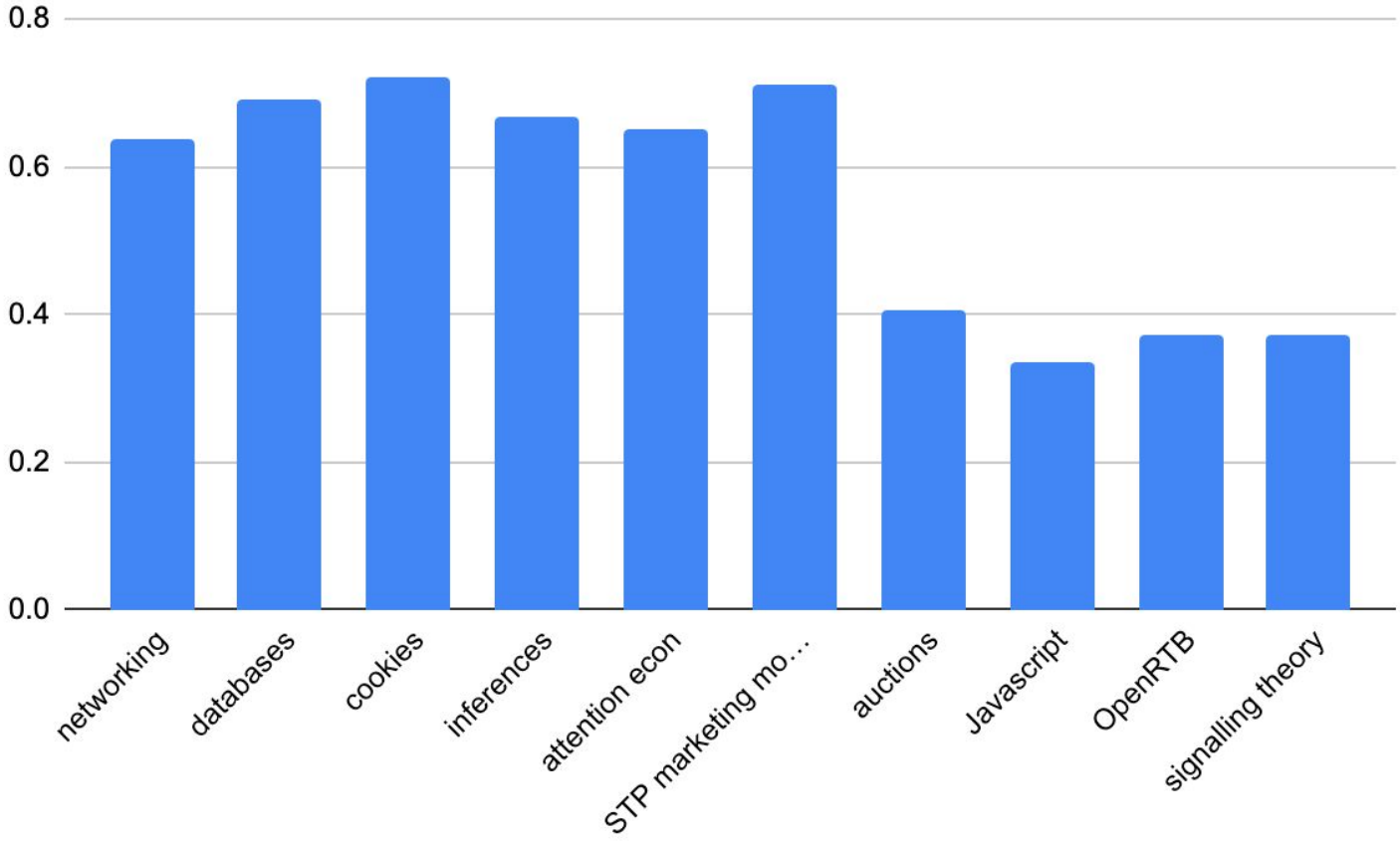
Strategy: don't guess if you're not sure!
False positives penalized (1 pt each) more than many
Correct positives earn (m/n pt each)

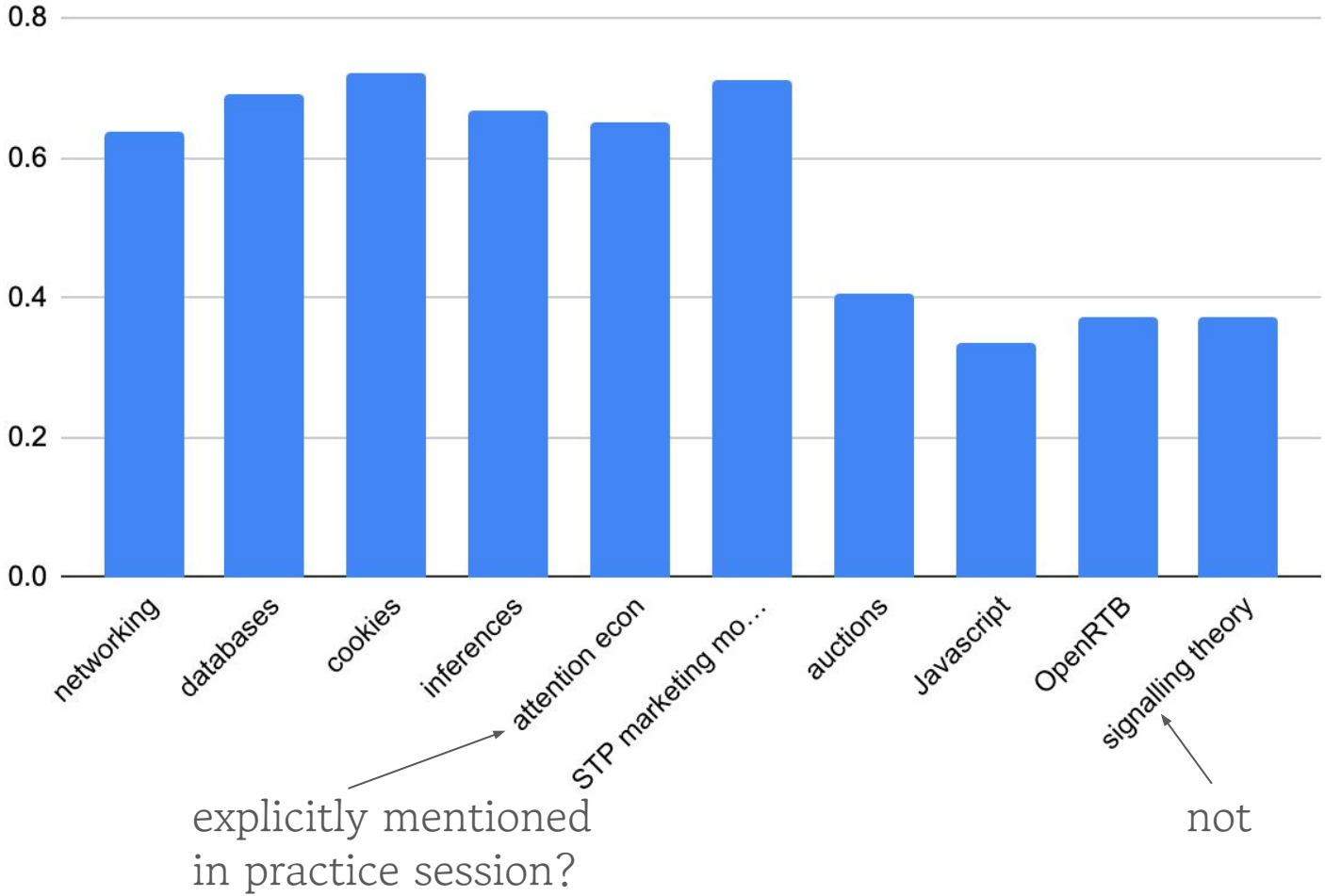


Search/chat results less reliable

Important to know
(↑ confidence):
what you know
what you do not

Strategy: don't guess if you're not sure!
False positives penalized (1 pt each) more than many
Correct positives earn (m/n pt each)


0.8
0.6
0.4
0.2
0.0
networking
databases
cookies
inferences
attention econ
STP marketing mo…
auctions
Javascript
OpenRTB
signalling theory


0.8
0.6
0.4
0.2
0.0
networking
databases
cookies
inferences
attention econ
STP marketing mo…
auctions
Javascript
OpenRTB
signalling theory
explicitly mentioned
in practice session?
not

18. (3 points) Describe the difference between a cheap signal and a costly signal. Give an example of each in the context of advertising.

Signaling Theory

- Explains how people & animals communicate by sending information using observable actions/traits
- examples:



colorful feathers as a signal of a bird's ability to gather nutrients

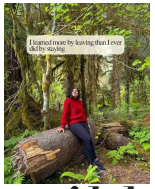


college degree as a signal of a job seeker's productivity

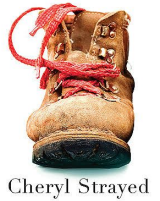
Michael Spence "Job Market Signaling" 1973 & Amotz Zahavi "Mate Selection—A Selection for a Handicap" 1975

\+ cost for person giving signal (not receiver)
\+ explain your reasoning

To communicate love of hiking:



create Instagram post



write a book

18. (3 points) Describe the difference between a cheap signal and a costly signal. Give an example of each in the context of advertising.

Signaling Theory

- Explains how people & animals communicate by sending information using observable actions/traits
- examples:



colorful feathers as a signal of a bird's ability to gather nutrients

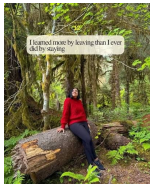


college degree as a signal of a job seeker's productivity

Michael Spence "Job Market Signaling" 1973 & Amotz Zahavi "Mate Selection—A Selection for a Handicap" 1975

\+ cost for person giving signal (not receiver)
\+ explain your reasoning

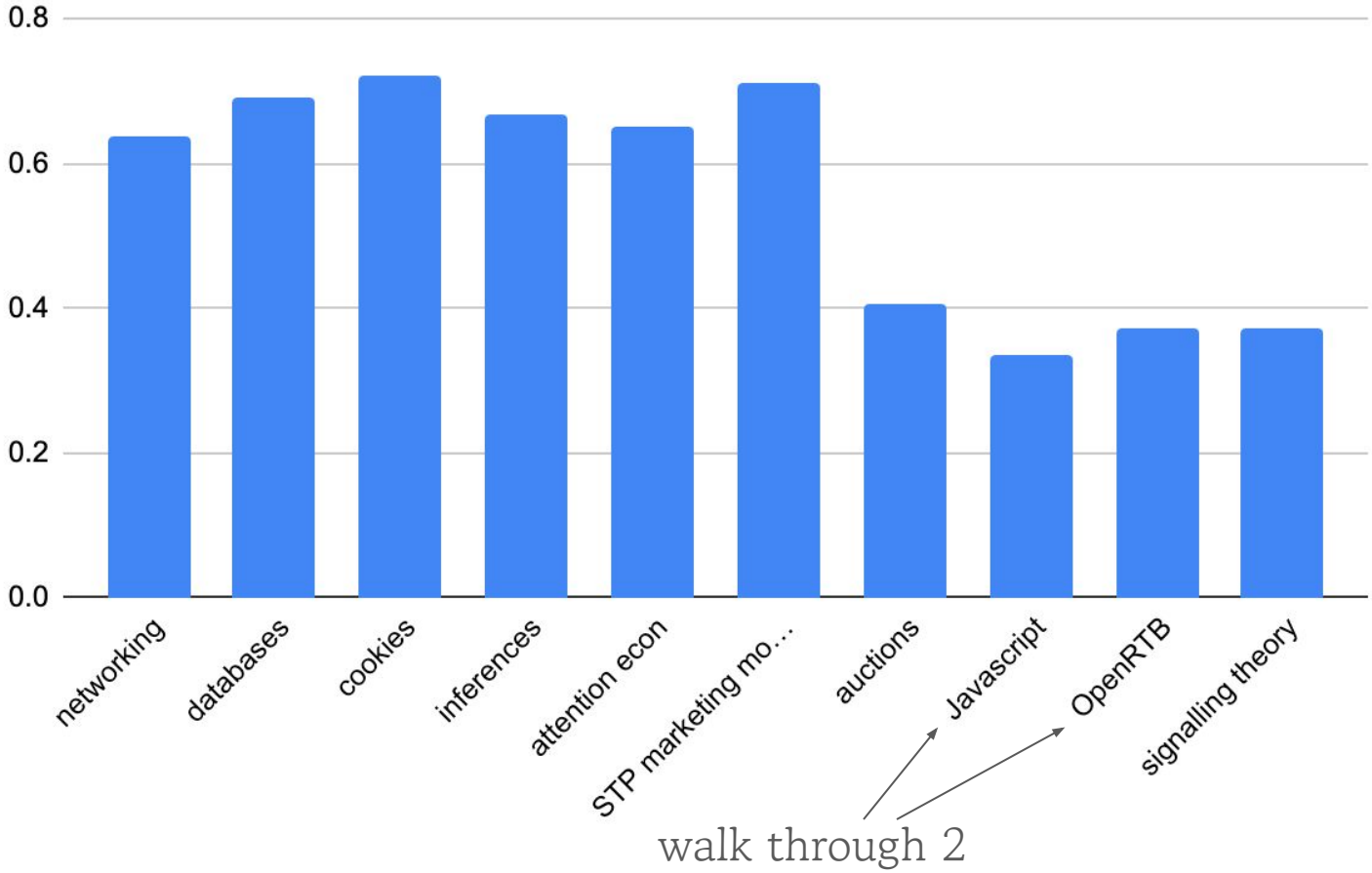
To communicate love of hiking:



create Instagram post



wear Patagonia gear


0.8
0.6
0.4
0.2
0.0
networking
databases
cookies
inferences
attention econ
STP marketing mo…
auctions
Javascript
OpenRTB
signalling theory
walk through 2

13. (2 points) Your page, `boogle.com`, includes a piece of JavaScript by including the following line of code in its HTML `<head>`:

```
<script src="https://kumarde.com/script.js"></script>
```

Which of the following are true of `script.js`?

- ◯ `script.js` can set first-party cookies
- ◯ `script.js` can redirect the browser
- ◯ `script.js` can read email addresses put into web forms
- ◯ `script.js` can communicate with the backend server database
- ◯ `script.js` can return `HTTP 404` to the web browser

Which of the following are true of `script.js`?

- ◯ `script.js` can set first-party cookies
- ◯ `script.js` can redirect the browser
- ◯ `script.js` can read email addresses put into web forms
- ◯ `script.js` can communicate with the backend server database
- ◯ `script.js` can return `HTTP 404` to the web browser

JAVASCRIPT CAN HURT YOU

- Many tutorials on the web have you include things like the following:

```
<script src="jquery.com/min.js"></script>
```

- You should know that loading JavaScript like this **gives this script first-party access to your page**
- This means that the JS can…
  - Read your cookies
  - Modify any variables you store on the client
  - Read most browser properties
  - **And send this data anywhere it wants on the web.** **Proceed with caution!**

Which of the following are true of `script.js`?

- ◯ `script.js` can set first-party cookies
- ◯ `script.js` can redirect the browser
- ◯ `script.js` can read email addresses put into web forms
- ◯ `script.js` can communicate with the backend server database
- ◯ `script.js` can return `HTTP 404` to the web browser

anything you can do with Javascript (as website owner), this script.js can do too!

```
document.querySelector(
'input[type="email"]')
```

```
document.getElementById
("email")
```

JAVASCRIPT CAN HURT YOU

- Many tutorials on the web have you include things like the following:

```
<script src="jquery.com/min.js"></script>
```

- You should know that loading JavaScript like this **gives this script first-party access to your page**
- This means that the JS can…
  - Read your cookies
  - Modify any variables you store on the client
  - Read most browser properties
  - **And send this data anywhere it wants on the web.** **Proceed with caution!**

Which of the following are true of `script.js`?

- ◯ `script.js` can set first-party cookies
- ◯ `script.js` can redirect the browser
- ◯ `script.js` can read email addresses put into web forms
- ◯ `script.js` can communicate with the backend server database → use HTTP request methods (GET, POST)
- ◯ `script.js` can return `HTTP 404` to the web browser → by the time client-side JS runs, HTTP response header already received (cannot change status)

JAVASCRIPT CAN HURT YOU

- Many tutorials on the web have you include things like the following:

```
<script src="jquery.com/min.js"></script>
```

- You should know that loading JavaScript like this **gives this script first-party access to your page**
- This means that the JS can…
  - Read your cookies
  - Modify any variables you store on the client
  - Read most browser properties
  - **And send this data anywhere it wants on the web.** **Proceed with caution!**

14. OpenRTB Specifics.

(a) (2 points) Select all information that is valid to send via the OpenRTB protocol.

- ◯ Impression size
- ◯ Bid floor
- ◯ Authentication tokens
- ◯ Device details (User Agent, IP, ...)
- ◯ User cookies
- ◯ Email address

(b) (2 points) Select only information that is *both* valid to send via the OpenRTB protocol and used to personalize advertisements (i.e., target a specific user).

- ◯ Impression size
- ◯ Bid floor
- ◯ Authentication tokens
- ◯ Device details (User Agent, IP, ...)
- ◯ User cookies
- ◯ Email address

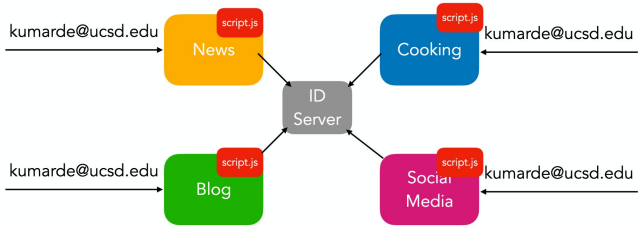
14. OpenRTB Specifics.

(a) (2 points) Select all information that is valid to send via the OpenRTI

- ◯ Impression size
- ◯ Bid floor
- ◯ Authentication tokens
- ◯ Device details (User Agent, IP, ...)
- ◯ User cookies
- ◯ Email address

PII is forbidden in OpenRTB
email addresses for "enrichment"
sent directly to server using JS

Post-Cookie Identifier Tracking



script.js will capture JS variables (e.g., email) and send to an ID server!

A simple OpenRTB Request

```
{
  "id": "req-123", "at": 1, "tmax": 150,
  "imp": [{ "id": "1", "banner": { "w": 300, "h": 250 }, "bidfloor": 0.50 }],
  "site": { "domain": "news.example", "page": "https://news.example/sports/
game" },
  "device": { "ua": "Mozilla/5.0 ...", "ip": "203.0.113.0" },
  "user": { "buyeruid": "dsp-u-77f1" }
}
```

an ID that matches the cookie ID
(but not a cookie!)

The user has an identifier (likely, a cookie) and device information (including IP addresses!)

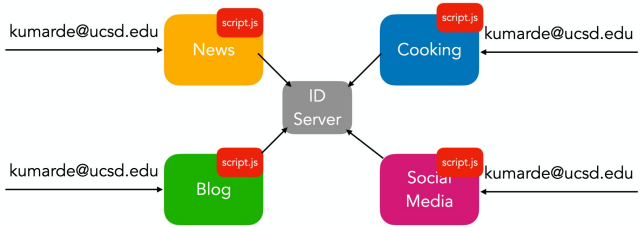
14. OpenRTB Specifics.

(a) (2 points) Select all information that is valid to send via the OpenRTI

- ◯ Impression size
- ◯ Bid floor
- ◯ Authentication tokens
- ◯ Device details (User Agent, IP, ...)
- ◯ User cookies
- ◯ Email address

PII is forbidden in OpenRTB
email addresses for "enrichment"
sent directly to server using JS

Post-Cookie Identifier Tracking



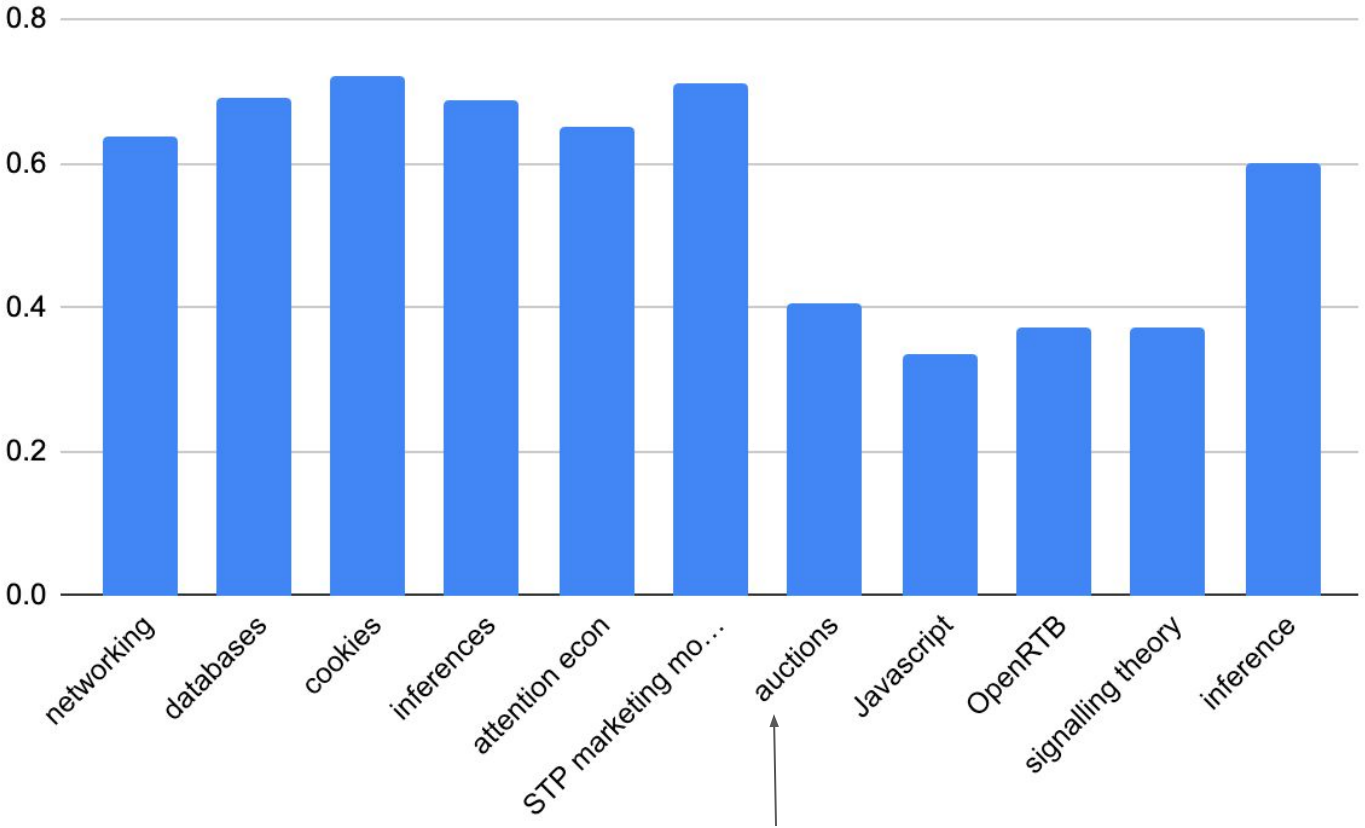
script.js will capture JS variables (e.g., email) and send to an ID server!

A simple OpenRTB Request

```
{
  "id": "req-123", "at": 1, "tmax": 150,
  "imp": [{ "id": "1", "banner": { "w": 300, "h": 250 }, "bidfloor": 0.50 }],
  "site": { "domain": "news.example", "page": "https://news.example/sports/
game" },
  "device": { "ua": "Mozilla/5.0 ...", "ip": "203.0.113.0" },
  "user": { "buyeruid": "dsp-u-77f1" }
}
```

an ID that matches the cookie ID
(but not a cookie!)

The user has an identifier (likely, a cookie) and device information (including IP addresses!)


0.8
0.6
0.4
0.2
0.0
networking
databases
cookies
inferences
attention econ
STP marketing mo…
auctions
Javascript
OpenRTB
signalling theory
inference
end of PA style problem
very low performance

4. (2 points) Consider the outcome of running this query on the Users and Visits tables:

```
SELECT Users.name
FROM Users
JOIN Visits ON Users.
    user_id = Visits.
    user_id;
```

Users

user_id	name
1	Anna
2	Bolin
3	Chinmay

Visits

visit_id	user_id
10	1
11	1
12	2

JOINS

- allows you to retrieve data from two or more tables
- all tables must share a related column

```
SELECT table_name_1.column1
    FROM table_name_1
    JOIN table_name_2
    ON table_name_1.column1=table_name_2.column2 ;
```

- if there is no related column, you cannot join those two tables
- ⚠️ the related column is not *required* to be a Foreign Key, but often faster
- ex: joining customers and orders tables based on customer email address even though no official relationship/key

CRUD (Create, **Read**, Update, Delete)

Some other useful tips:

- To return only unique rows:
  `SELECT DISTINCT column FROM table_name;`
- To return only the first N values:
  `SELECT column FROM table_name LIMIT N;`
- SQL provides many built-in math operations (SUM, ROUND, EXP, RAND):
  `SELECT SUM(column) FROM table_name;`
- ⚠️ Aggregate functions like SUM, COUNT cannot be used inside WHERE

AUCTIONS

(f) Finally, the tracking and inference code is working! To monetize on their website, Yalp decides to build a custom SSP to start running second-price auctions. They work with the same four DSPs from Boogle: (*steady*, *niche*, *shady*, and *glitch*).

Each auction uses a bid floor of 1.00 CPM, and the winner pays

$$\text{price} = \min\big(\text{winner's bid},\ \max(\text{second-highest valid bid},\ \text{floor}) + 0.01\big).$$

Unfortunately, SSP revenue is lower than expected and the engineers want to get to the bottom of it.

i. Here is the part of Yalp's SSP that sends bid requests to the DSPs:

```
start = time.time()
for dsp in ["steady", "shady", "glitch", "niche"]:
    resp = requests.post(BASE + dsp + "/bid", json=bid_req, timeout=0.25)
    elapsed_ms = (time.time() - start) * 1000
    if elapsed_ms > bid_req["tmax"]:
        continue            # arrived after tmax: discard
    process(resp)           # validate and keep this DSP's bids
```

`tmax` is 250 ms, and every DSP responds in about 90 ms. Yalp has verified that its bid requests include the correct segments. About half of Yalp's users carry a `parent` segment, which the *niche* DSP is known to bid on aggressively.

Over the last 1,000 auctions, Yalp logged these win counts:

DSP	Auctions won
steady	214
shady	786
glitch	0
niche	0

α) (2 points) Using the log and the code, explain why *glitch* and *niche* have never won.

β) (1 point) How would you fix the problem?

(f) Finally, the tracking and inference code is working! To monetize on their website, Yalp decides to build a custom SSP to start running second-price auctions. They work with the same four DSPs from Boogle: (*steady*, *niche*, *shady*, and *glitch*).

Each auction uses a bid floor of 1.00 CPM, and the winner pays

$$\text{price} = \min\big(\text{winner's bid},\ \max(\text{second-highest valid bid},\ \text{floor}) + 0.01\big).$$

Unfortunately, SSP revenue is lower than expected and the engineers want to get to the bottom of it.

i. Here is the part of Yalp's SSP that sends bid requests to the DSPs:

```
start = time.time()
for dsp in ["steady", "shady", "glitch", "niche"]:
    resp = requests.post(BASE + dsp + "/bid", json=bid_req, timeout=0.25)
    elapsed_ms = (time.time() - start) * 1000
    if elapsed_ms > bid_req["tmax"]:
        continue            # arrived after tmax: discard
    process(resp)           # validate and keep this DSP's bids
```

`tmax` is 250 ms, and every DSP responds in about 90 ms. Yalp has verified that its bid requests include the correct segments. About half of Yalp's users carry a `parent` segment, which the *niche* DSP is known to bid on aggressively.

Over the last 1,000 auctions, Yalp logged these win counts:

DSP	Auctions won
steady	214
shady	786
glitch	0
niche	0

α) (2 points) Using the log and the code, explain why *glitch* and *niche* have never won.

β) (1 point) How would you fix the problem?

why is increasing tmax not a good solution?

ii. With the bug fixed, Yalp decides to split its users into two equally common groups: users with a `parent` segment and all other users.

After running for a while, the Yalpers notice that *shady* DSP's ads often draw complaints from users. When a user complains, this costs the Yalp SSP \$0.0002 (\$2.00 CPM), so a teammate adds *shady*'s ad domain to `badv` on every bid request so that *shady* stops bidding entirely. Throughout this period, assume every DSP bids the same amount on every user within a group (parent/other).

Yalp's *net profit* for an impression is its clearing price minus the expected complaint cost. Express all answers below in CPM.

Here are Yalp's logs from before blocking *shady* and after blocking *shady*:

	Before			**After**		
	Winner	Clearing price	Complaint rate	Winner	Clearing price	Complaint rate
`parent` users	niche	2.51	0%	niche	1.51	0%
Other users	shady	1.51	50%	steady	1.01	0%

α) (2 points) Compute Yalp's average net profit per impression, across both groups of users, before and after the change.

ii. With the bug fixed, Yalp decides to split its users into two equally common groups: users with a `parent` segment and all other users.

After running for a while, the Yalpers notice that *shady* DSP's ads often draw complaints from users. When a user complains, this costs the Yalp SSP $0.0002 ($2.00 CPM), so a teammate adds *shady*'s ad domain to `badv` on every bid request so that *shady* stops bidding entirely. Throughout this period, assume every DSP bids the same amount on every user within a group (parent/other).

Yalp's *net profit* for an impression is its clearing price minus the expected complaint cost. Express all answers below in CPM.

Here are Yalp's logs from before blocking *shady* and after blocking *shady*:

	Before			**After**		
	Winner	Clearing price	Complaint rate	Winner	Clearing price	Complaint rate
`parent` users	niche	2.51	0%	niche	1.51	0%
Other users	shady	1.51	50%	steady	1.01	0%

β) (1 point) Did blocking *shady* increase profits? Why or why not?

γ) (2 points) Using only the logs, what is one possible value *shady* bid on `parent` users before the block? And what can you say about *niche*'s bid?

every DSP bids same amount throughout period -- what position was shady in parent and other users auctions **before**?

ii. With the bug fixed, Yalp decides to split its users into two equally common groups: users with a `parent` segment and all other users.

After running for a while, the Yalpers notice that *shady* DSP's ads often draw complaints from users. When a user complains, this costs the Yalp SSP \$0.0002 (\$2.00 CPM), so a teammate adds *shady*'s ad domain to `badv` on every bid request so that *shady* stops bidding entirely. Throughout this period, assume every DSP bids the same amount on every user within a group (parent/other).

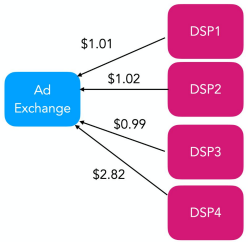
Yalp's *net profit* for an impression is its clearing price minus the expected complaint cost. Express all answers below in CPM.

Here are Yalp's logs from before blocking *shady* and after blocking *shady*:

	Before			After		
	Winner	Clearing price	Complaint rate	Winner	Clearing price	Complaint rate
`parent` users	niche	2.51	0%	niche	1.51	0%
Other users	shady	1.51	50%	steady	1.01	0%

Second-Price Auction

- In a second-price auction, the **highest bidder** wins the auction, but only pays **the second-highest bid price** (+0.01)
- Which DSP should win the auction?
  - DSP4! How much should they pay? **\$1.03**



ɜ) (1 point) Did blocking *shady* increase profits? Why or why not?

γ) (2 points) Using only the logs, what is one possible value *shady* bid on `parent` users before the block? And what can you say about *niche*'s bid?

(f) Finally, the tracking and inference code is working! To monetize on their website, Yalp decides to build a custom SSP to start running second-price auctions. They work with the same four DSPs from Boogle: (*steady*, *niche*, *shady*, and *glitch*).

Each auction uses a bid floor of 1.00 CPM, and the winner pays

$$\text{price} = \min\big(\text{winner's bid},\ \max(\text{second-highest valid bid},\ \text{floor}) + 0.01\big).$$

Here are Yalp's logs from before blocking *shady* and after blocking *shady*:

	Before			**After**		
	Winner	Clearing price	Complaint rate	Winner	Clearing price	Complaint rate
`parent` users	niche	2.51	0%	niche	1.51	0%
Other users	shady	1.51	50%	steady	1.01	0%

δ) (2 points) Propose a better `badv` policy that uses information Yalp already has when it builds each request. Using the logs, compute your policy's average net profit to justify your choice.

Browser Fingerprinting

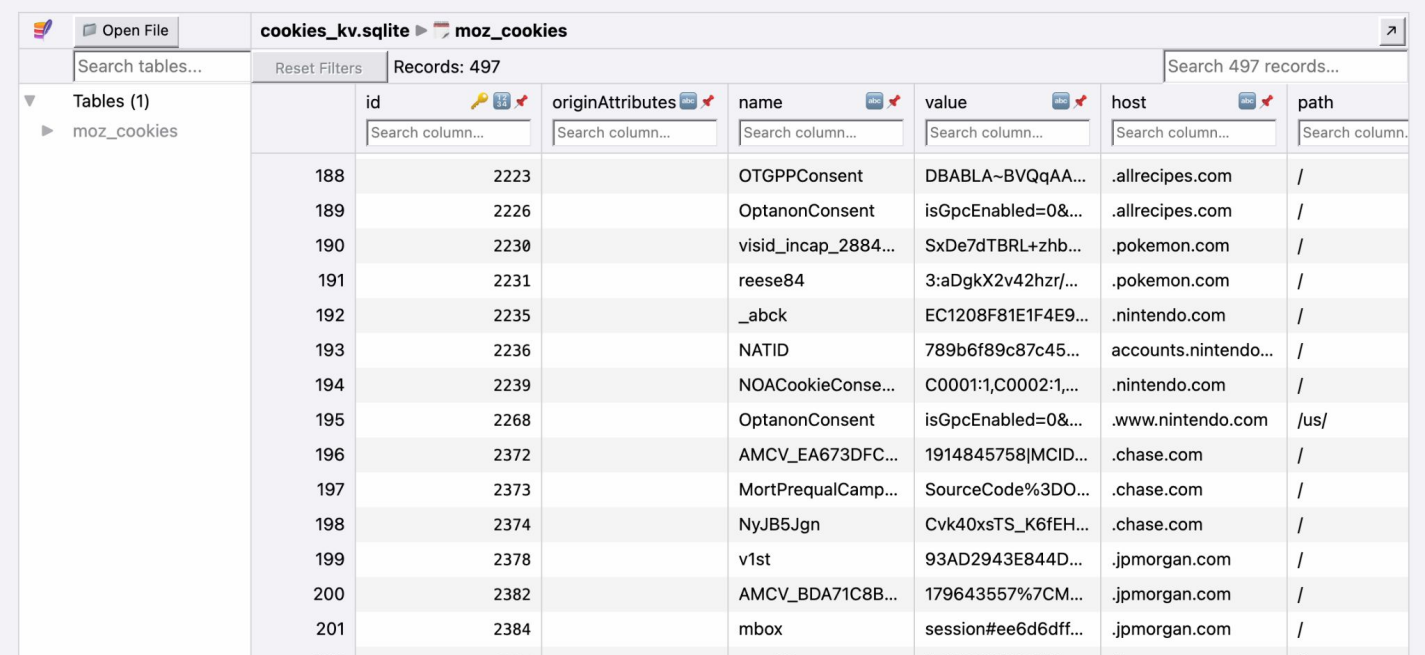
Ad Targeting and Surveillance Pricing

UC San Diego

How do you know to track website visitors?

Cookies

- small file, used to be text now often DB (Chrome, Firefox, Edge → SQLite)
- saved locally on your phone, tablet, computer



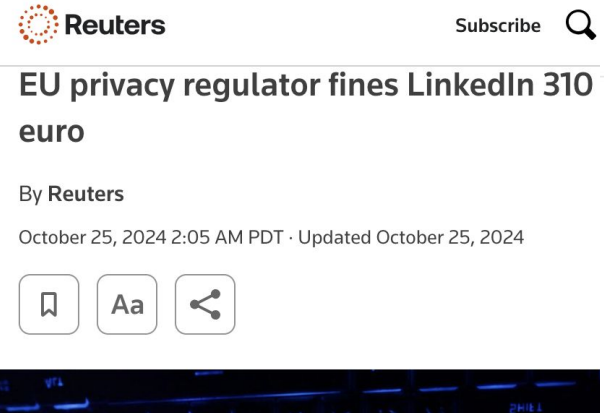
	id	originAttributes	name	value	host	path
	Search column...	Search column...	Search column...	Search column...	Search column...	Search column...
188	2223		OTGPPConsent	DBABLA~BVQqAA...	.allrecipes.com	/
189	2226		OptanonConsent	isGpcEnabled=0&...	.allrecipes.com	/
190	2230		visid_incap_2884...	SxDe7dTBRL+zhb...	.pokemon.com	/
191	2231		reese84	3:aDgkX2v42hzr/...	.pokemon.com	/
192	2235		_abck	EC1208F81E1F4E9...	.nintendo.com	/
193	2236		NATID	789b6f89c87c45...	accounts.nintendo...	/
194	2239		NOACookieConse...	C0001:1,C0002:1,...	.nintendo.com	/
195	2268		OptanonConsent	isGpcEnabled=0&...	.www.nintendo.com	/us/
196	2372		AMCV_EA673DFC...	1914845758|MCID...	.chase.com	/
197	2373		MortPrequalCamp...	SourceCode%3DO...	.chase.com	/
198	2374		NyJB5Jgn	Cvk40xsTS_K6fEH...	.chase.com	/
199	2378		v1st	93AD2943E844D...	.jpmorgan.com	/
200	2382		AMCV_BDA71C8B...	179643557%7CM...	.jpmorgan.com	/
201	2384		mbox	session#ee6d6dff...	.jpmorgan.com	/

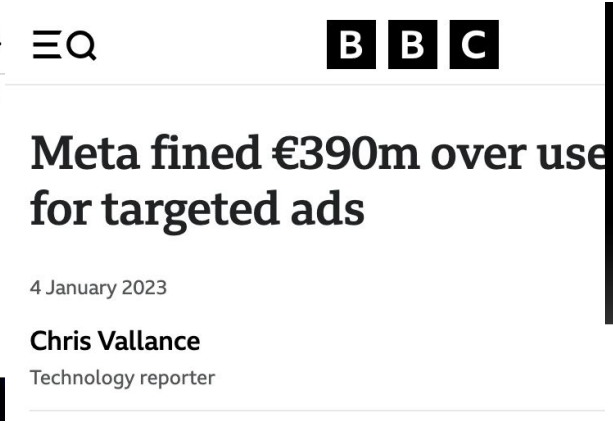
Cookies

- small file
- saved locally on your phone, tablet, computer
- store info like session ID to keep you logged in, language/font size, navigation tracking, shopping cart items
- **HTTP:** `Set-Cookie: <cookie-name>=<cookie-value>`
- **JS:** `document.cookie = "cookie-name=cookie-value";`
- Session vs. persistent cookies

Cookie Regulations

- **GDPR:** established
  1) higher standard for consent (specific, informed, clear affirmative)
  2) massive fines for non-compliance



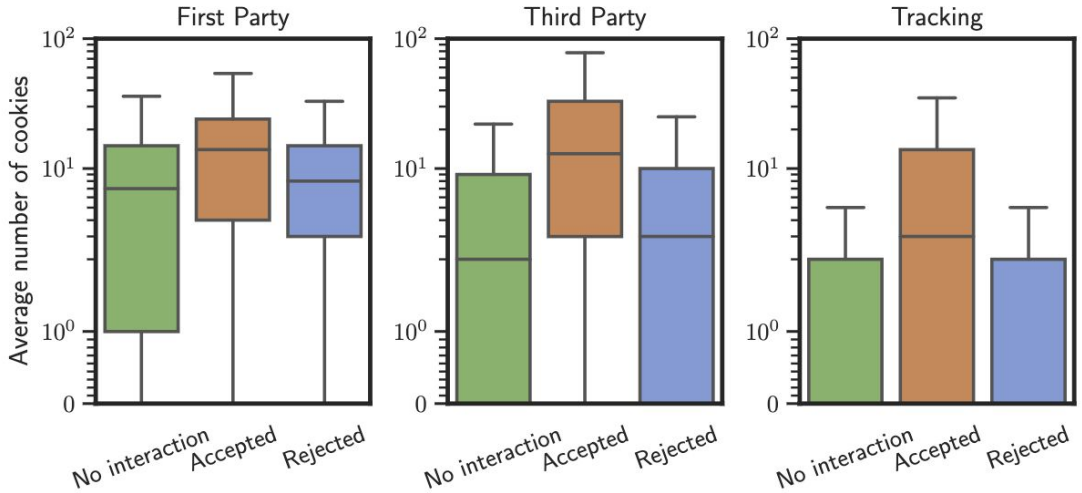




"Tracking Walls, Take-It-Or-Leave-It Choices, the GDPR, and the ePrivacy Regulation"
Frederik J Zuiderveen Borgesius 2017

Advertisers Respond to Regulation

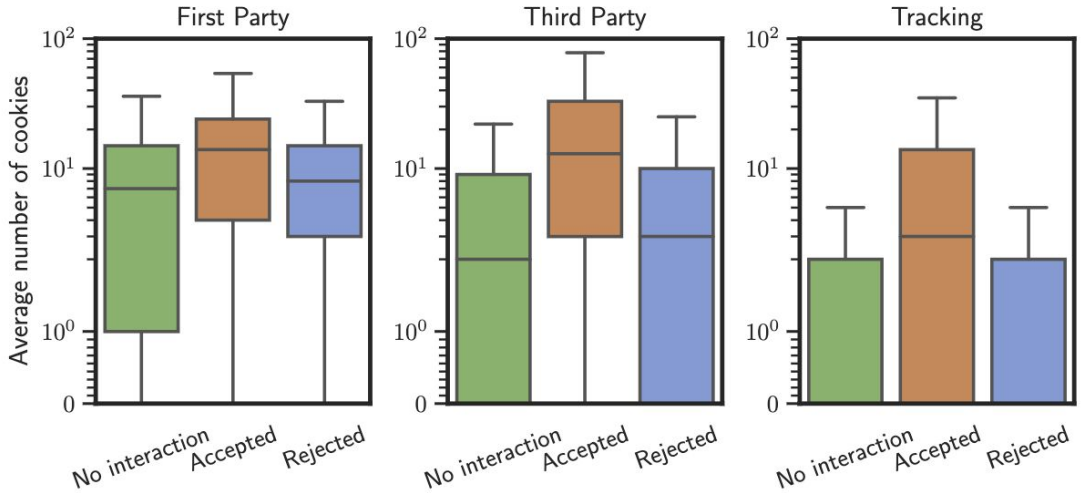
- adding cookie banners
- lying; some websites store tracking cookies before accept/despite reject
- finding ways to track without storing cookies



"Exploring the Cookieverse" Ali Rasaii 2023

Advertisers Respond to Regulation

- adding cookie banners
- lying; some websites store tracking cookies before accept/despite reject
- **finding ways to track without storing cookies**



"Exploring the Cookieverse" Ali Rasaii 2023

Stateful vs. Stateless Tracking

- both stateful and stateless tracking retain memory of past interactions
- key difference is where the information is stored

- **stateful tracking:** relies on data stored on *client*
  - user can remove this data (e.g., clear cache, delete cookies)
  - ex: cookies, browser cache, Local Storage

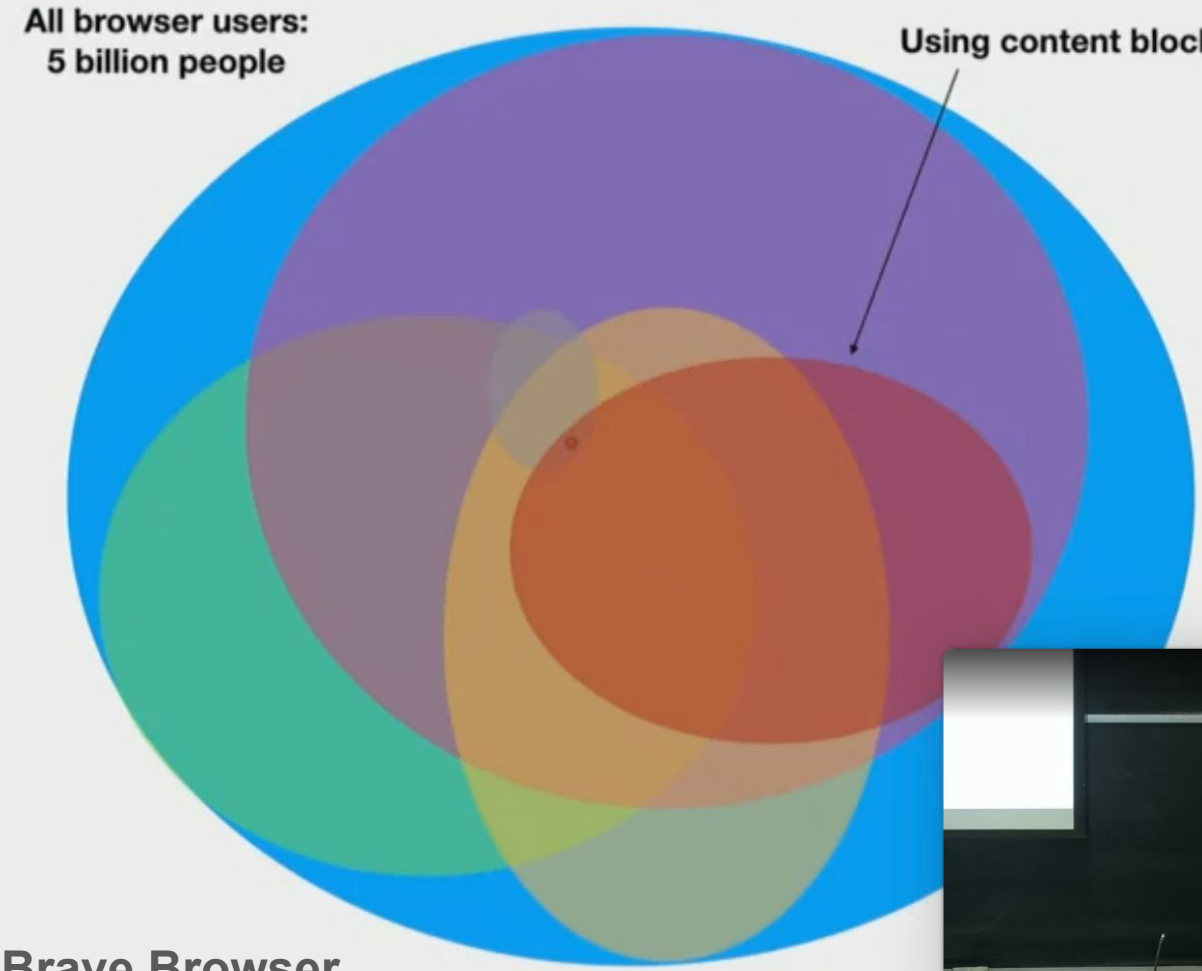
- **stateless tracking:** no data is stored on client
  - data likely stored on servers
  - user does not control this data
  - ex: Week 0! Also next few lectures!

Browser Fingerprinting

- stateless tracking (no data stored on client)
- can often uniquely identify users using data about device & browser

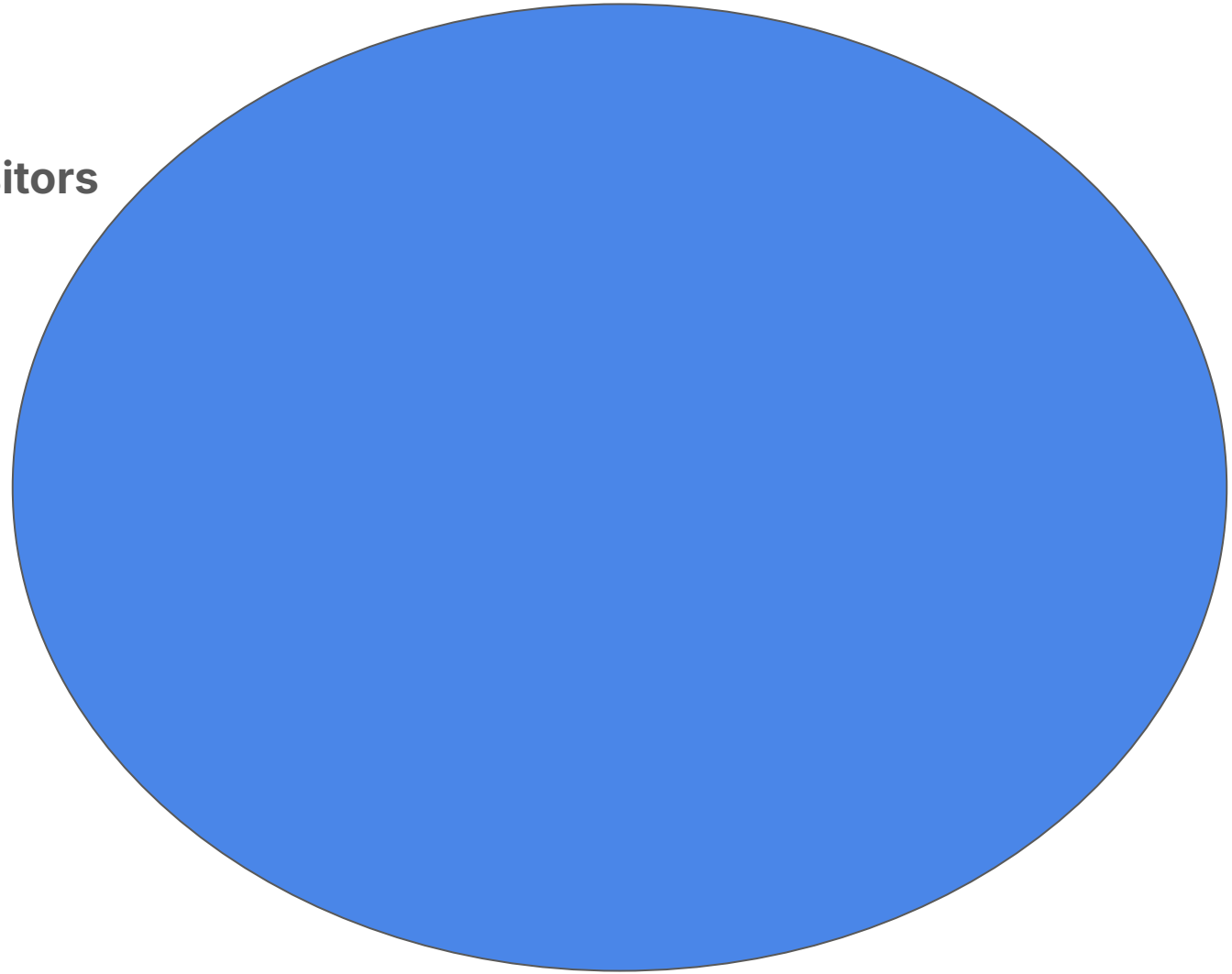
Browser Fingerprinting

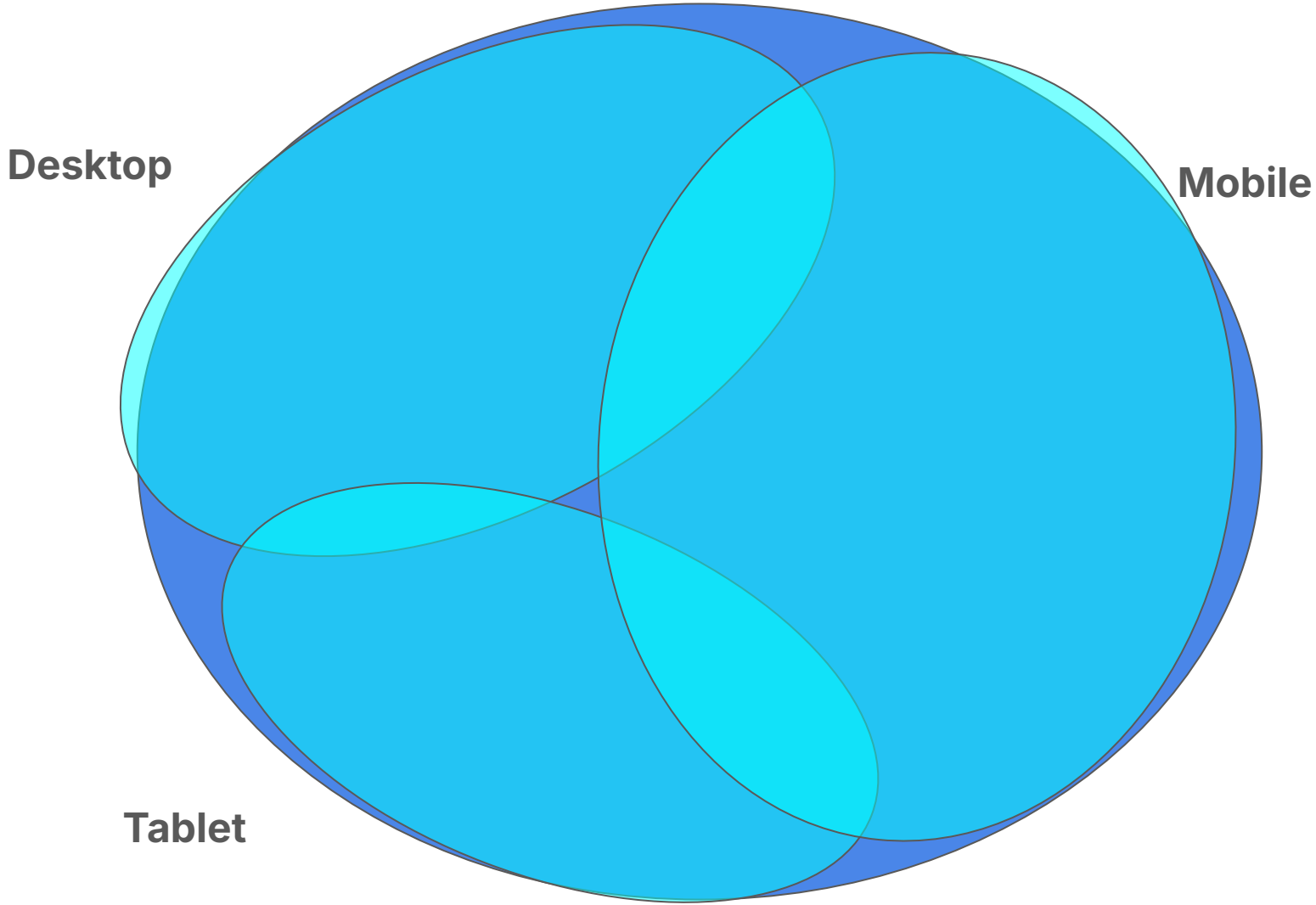
- stateless tracking (no data stored on client)
- can often uniquely identify users using data about device & browser
- **core intuition:** individual attributes are shared by many (even millions) *but*
- combining multiple attributes often specific enough to uniquely identify

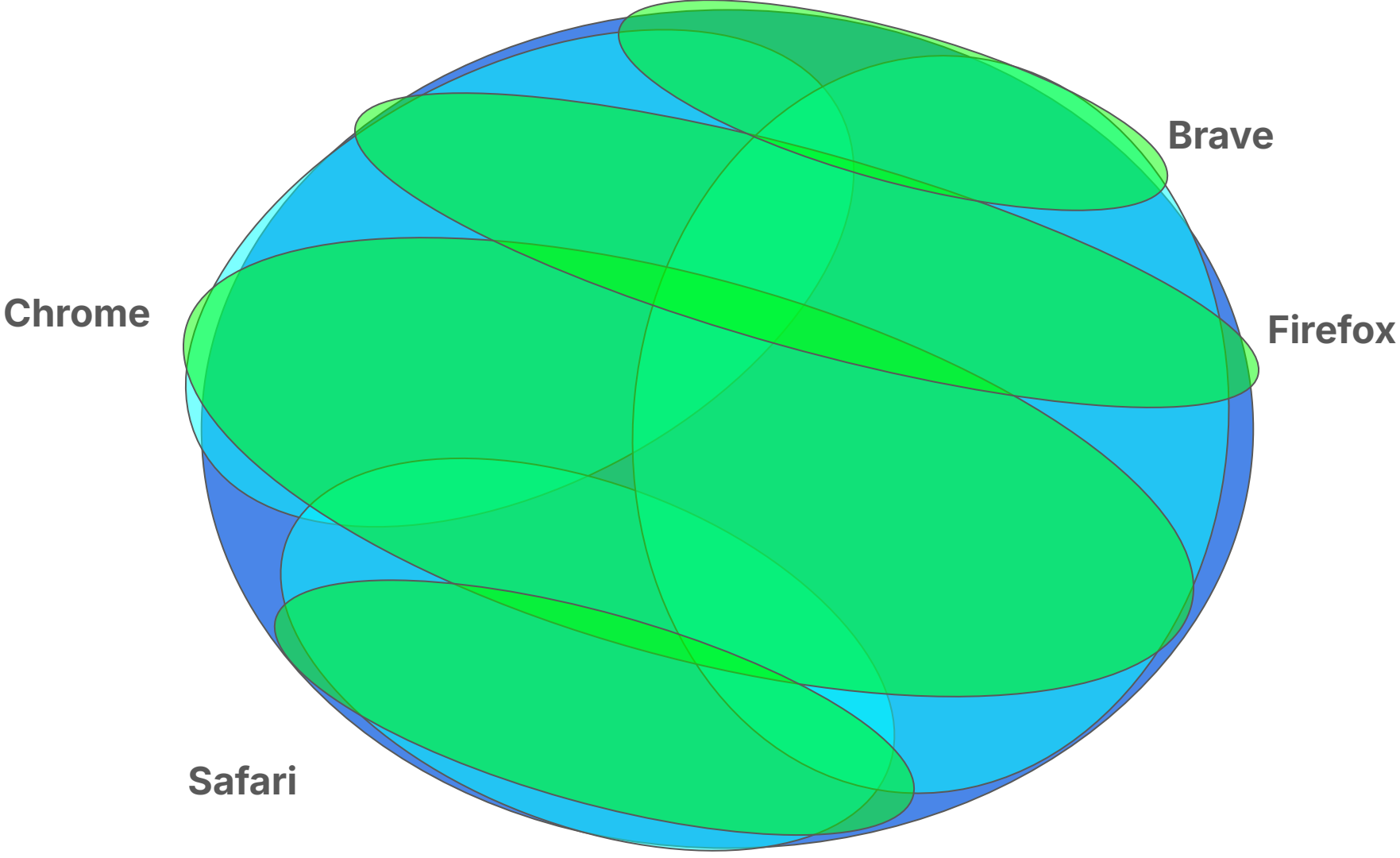


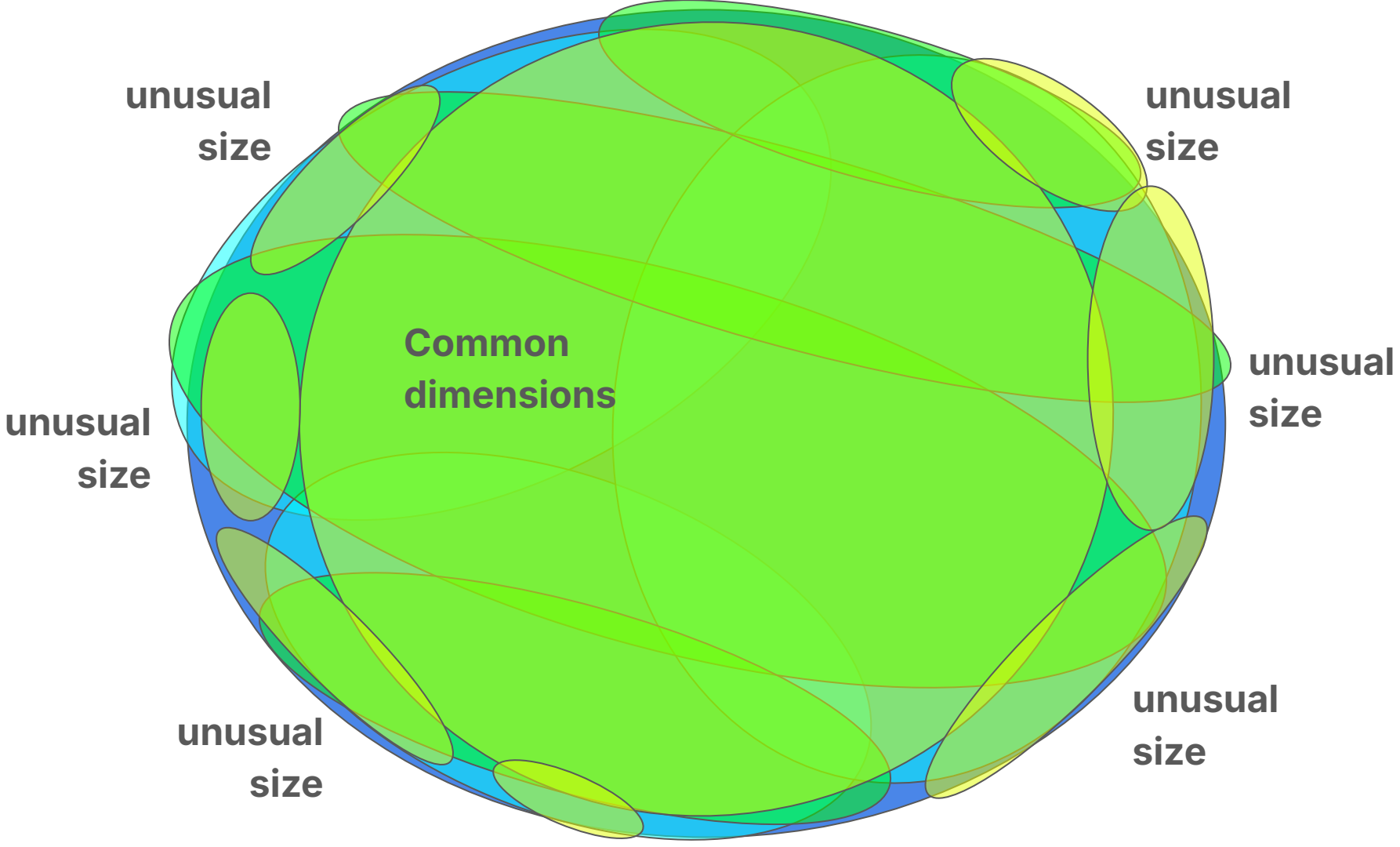
Pete Snyder, Brave Browser

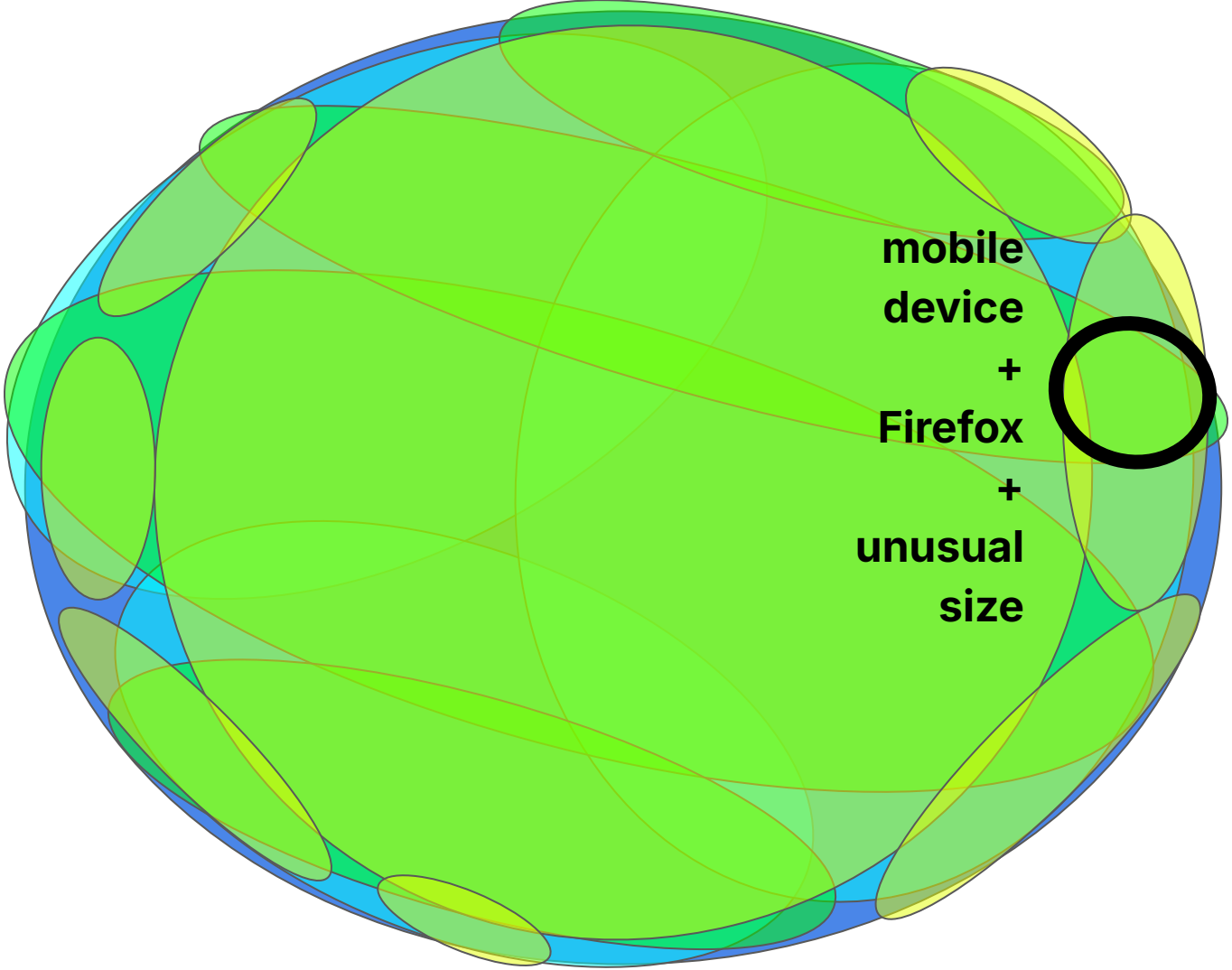
All visitors




Desktop
Mobile
Tablet


Brave
Chrome
Firefox
Safari


unusual size
unusual size
Common dimensions
unusual size
unusual size
unusual size
unusual size


mobile
device
+
Firefox
+
unusual
size

Browser Fingerprinting

- stateless tracking (no data stored on client)
- can often uniquely identify users using data about device & browser
- **core intuition:** individual attributes are shared by many (even millions) *but* combining multiple attributes often specific enough to uniquely identify

- Two basic types:
- **passive:** some data shared automatically by browser
- **active:** website runs scripts (using JS) to actively query browser for more info

Passive Fingerprinting:

Read default metadata sent automatically in HTTP headers properties

IP address, user-agent strings, language prefs

Active Fingerprinting:

Execute JS code to collect metadata not included in HTTP headers

device hardware (memory, CPUs), timezone

\+ **Canvas Fingerprinting:**

Uses <canvas> element to render hidden graphics/text to ID installed fonts, GPU drivers

Installed fonts, rendering behaviors/anomalies

\+ **WebGL Fingerprinting:**

3D version of <canvas>, render 3D scene using WebGL API

Render limits, shading language support

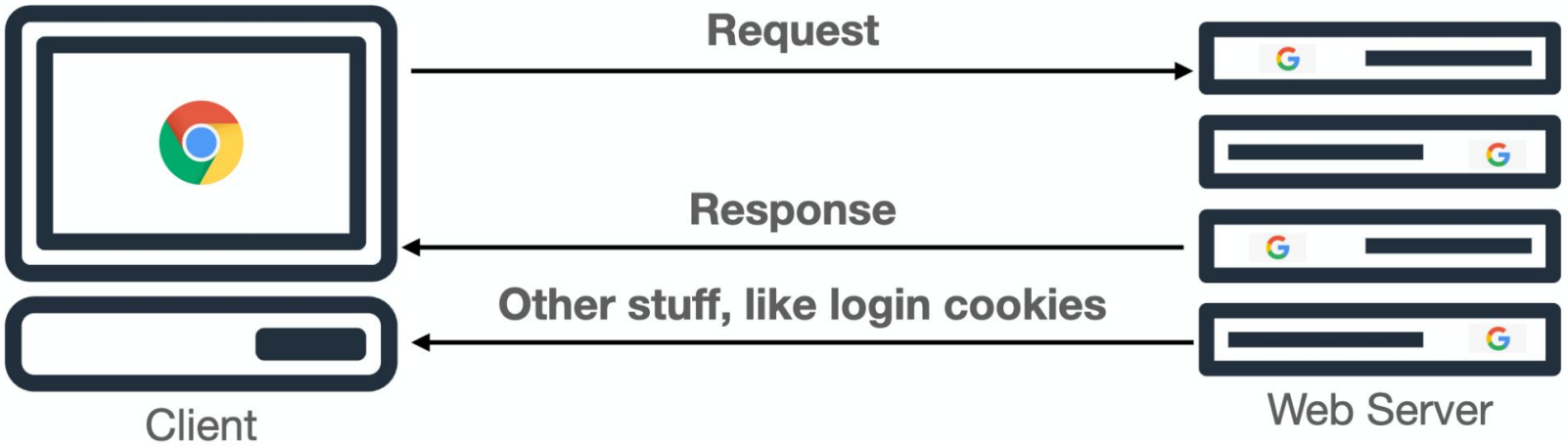
\+ **AudioContext:**

Generates audio signal, measures differences in how audiostack/hardware drivers process

Hardware response, sample rates, anomalies

Passive Fingerprinting

- Read default metadata sent automatically in HTTP headers
- **HTTP headers:** key-value metadata pairs sent between a client & server
- Request headers contain info about request, client capabilities, what expect back



https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Headers

HTTP Request Anatomy

- What does an HTTP Request look like?

what could be used for fingerprinting?

```
GET /index.html HTTP/1.1

Accept: image/gif, image/x-bitmap, image/jpeg, */*
Accept-Language: en
Connection: Keep-Alive
User-Agent: Mozilla/1.22 (compatible; MSIE 2.0; Windows 95)
Host: www.example.com
Referer: http://www.google.com?q=dingbats
```

Body (usually empty for requests)

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request

```
Mozilla/5.0 (Linux; Android 16; Pixel 9) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/143.0.12.45 Mobile Safari/537.36
```

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request

```
Mozilla/5.0 (Linux; Android 16; Pixel 9) AppleWebKit/537.36
```

legacy prefix

(operating system, device)

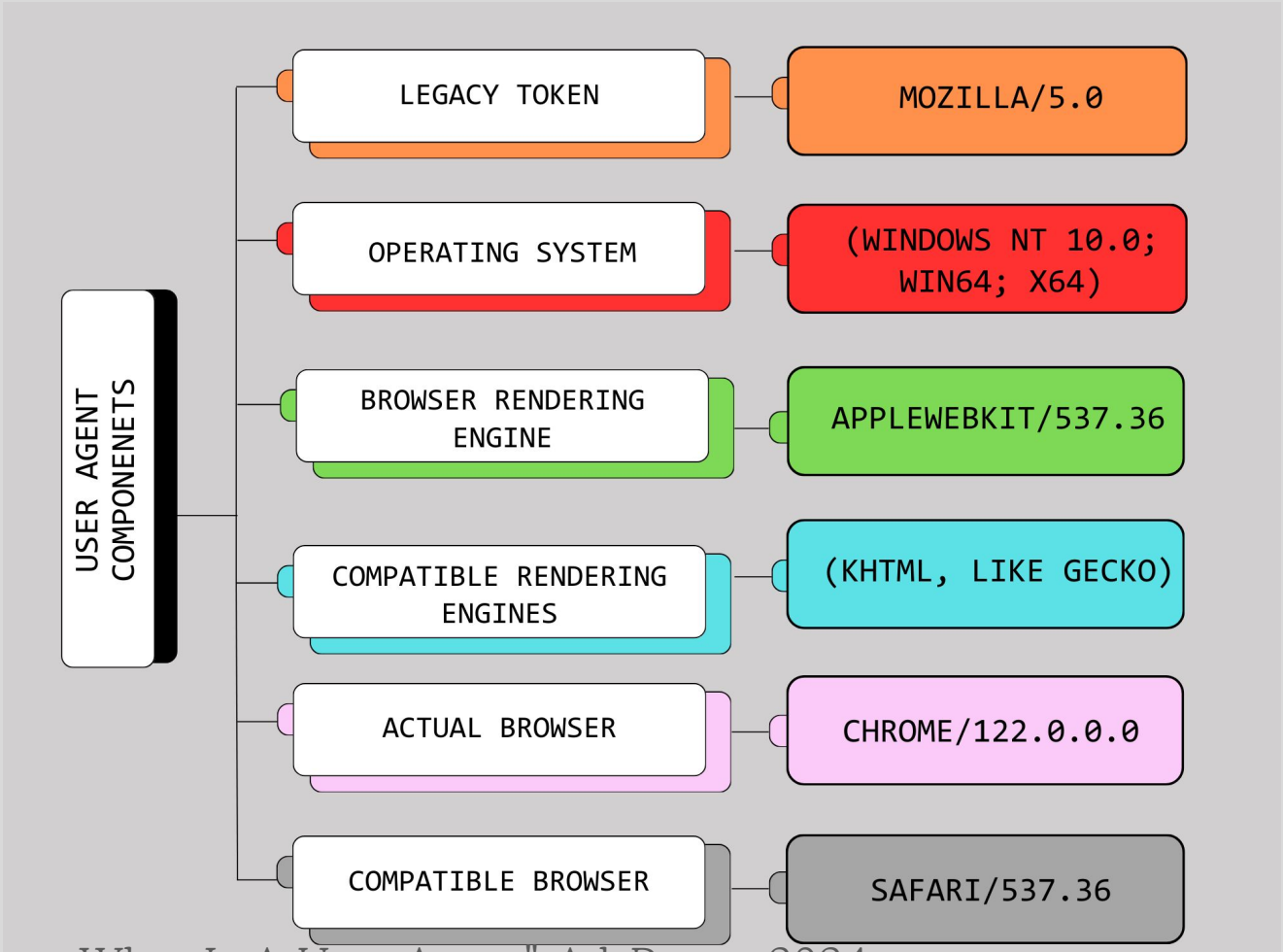
often legacy rendering engine

```
(KHTML, like Gecko) Chrome/143.0.12.45 Mobile Safari/537.36
```

(compatible layout engines)

browser version details

more historical compatibility info



"Breaking Down What Is A User Agent" AdsPower 2024

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request

```
Mozilla/5.0 (Linux; Android 16; Pixel 9) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/143.0.12.45 Mobile Safari/537.36
```

What device & browser?

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request

```
Mozilla/5.0 (Linux; Android 16; Pixel 9) AppleWebKit/537.36
    (KHTML, like Gecko) Chrome/143.0.12.45 Mobile Safari/537.36
```

Pixel 9 phone using Chrome web browser version 143

*note: Chrome version 143.0.12.45 is not a real version number

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request

```
Mozilla/5.0 (Linux; Android 16; Pixel 9) AppleWebKit/537.36
      (KHTML, like Gecko) Chrome/143.0.12.45 Mobile Safari/537.36
```

- messy, unstandardized, many legacy tokens
- avoid custom regex to parse
- use existing libraries/APIs

  `Javascript: ua-parser-js`
  `Python: ua-parser`

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- messy, unstandardized, many legacy tokens
- avoid custom regex to parse
- use existing libraries/APIs
- allows server to return compatible layouts, media formats, specific content tailored to device
- example: user requests video file
  If **Safari** User-Agent: deliver HTTP Live Streaming (HLS), Apple supports with full hardware acceleration
  If **Chrome** User-Agent: does not support HLS, instead deliver WebM file

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- messy, unstandardized, many legacy tokens
- avoid custom regex to parse
- use existing libraries/APIs
- allows server to return compatible layouts, media formats, specific content tailored to device
- ex: send optimized file types, block outdated/insecure browsers, link to OS-specific downloaders, etc.

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- provided in order of preference (best to worst)
- **quality values**: weights, used to express importance or priority, [0 1]

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- provided in order of preference (best to worst)
- **quality values**: weights, used to express importance or priority, [0 1]

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

what languages does this user prefer?

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- provided in order of preference (best to worst)
- **quality values**: weights, used to express importance or priority, [0 1]

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```



Top choice is Danish, then British English, then any other English

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- **Accept-Encoding:** list the compression algorithms the browser supports

```
Accept-Encoding: deflate, gzip;q=1.0, *;q=0.9
```

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- **Accept-Encoding:** list the compression algorithms the browser supports
- **Accept:** list the media types the browser is willing to receive

```
Accept: text/html, application/xhtml+xml,
application/xml;q=0.9, */*;q=0.8
```

```
Accept: image/avif, image/webp, image/png, image/svg+xml,
image/*;q=0.8,*/*;q=0.5
```

Passive Fingerprinting - HTTP Headers

- **User-Agent:** identifies the web browser, operating system, and device making the request
- **Accept-Language:** list the languages (& locations) the user prefers
- **Accept-Encoding:** list the compression algorithms the browser supports
- **Accept:** list the media types the browser is willing to receive
- & more



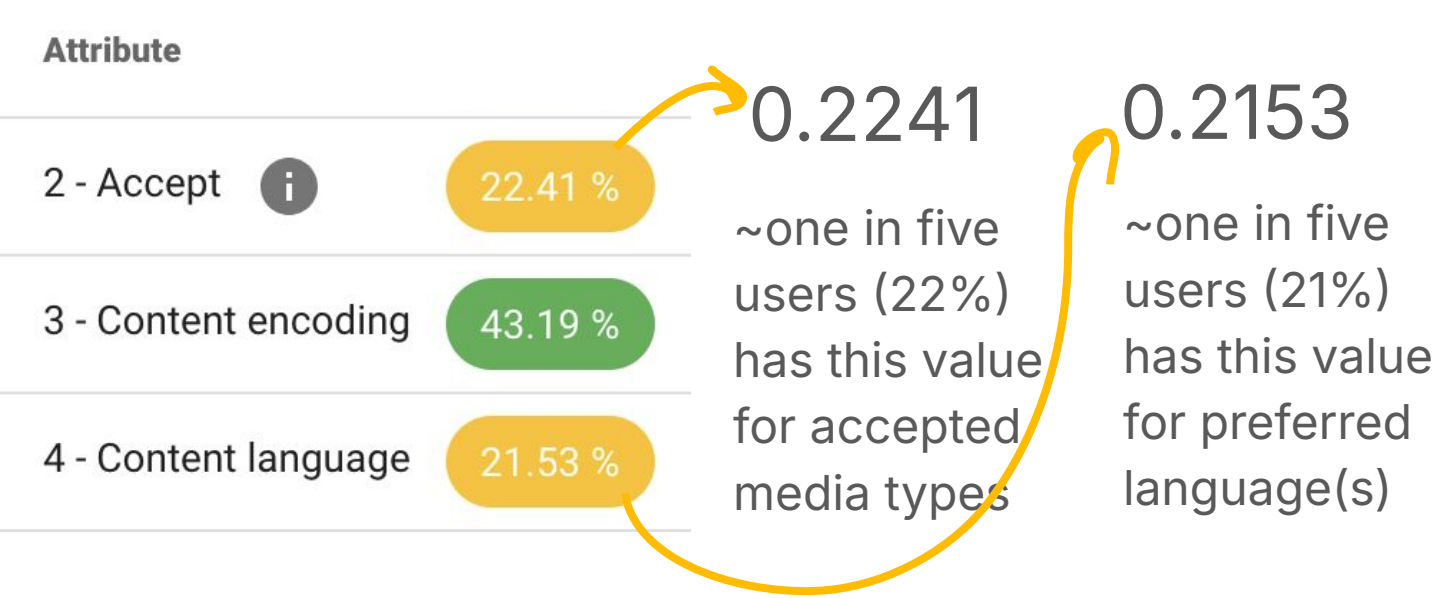
https://amiunique.org/fingerprint

My HTTP Header Data

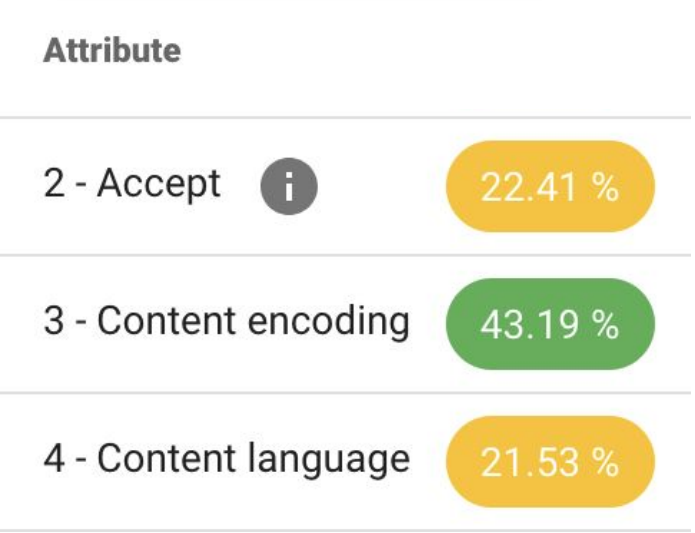
My HTTP Header Data



(Simplified) Likelihood Calculations



(Simplified) Likelihood Calculations



0.2241

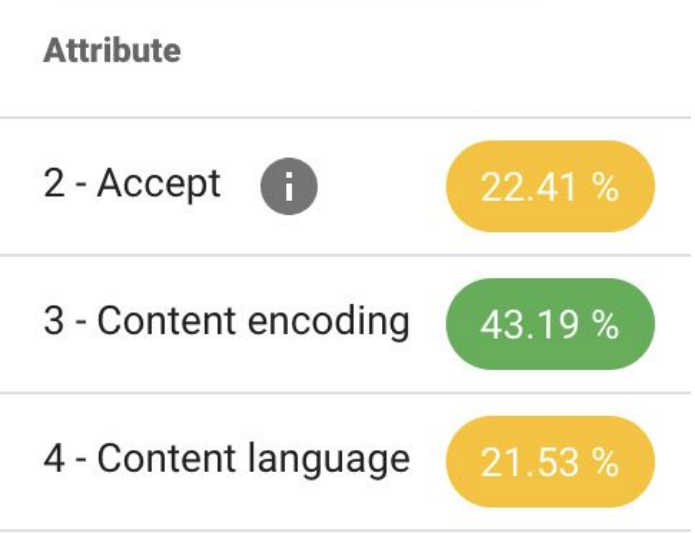
~one in five users (22%) has this value for accepted media types

0.2153

~one in five users (21%) has this value for preferred language(s)

How likely is one user is to have both of these properties?

(Simplified) Likelihood Calculations



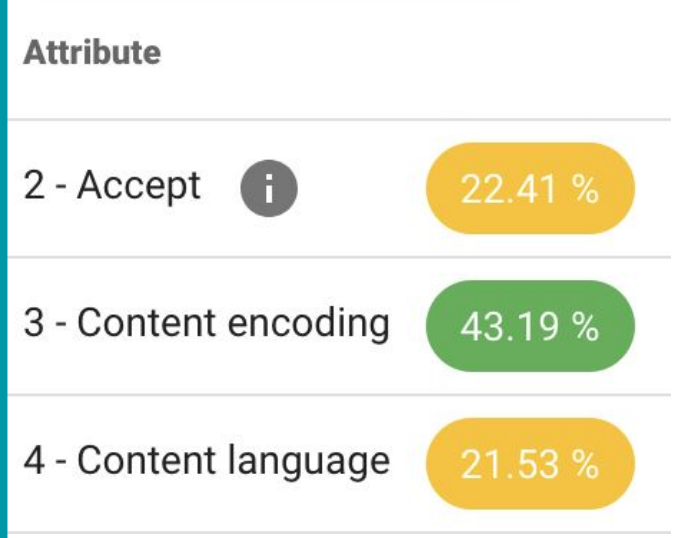
0.2241 x 0.2153 = 0.0482

~one in five users (22%) has this value for accepted media types

~one in five users (21%) has this value for preferred language(s)

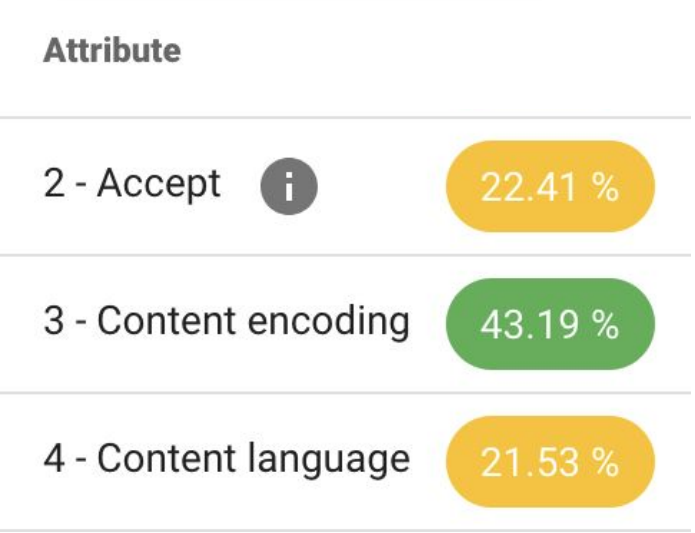
how likely one user is to have both of these properties (4.8%)

(Simplified) Likelihood Calculations



how likely is one user to have all 3 properties?

(Simplified) Likelihood Calculations



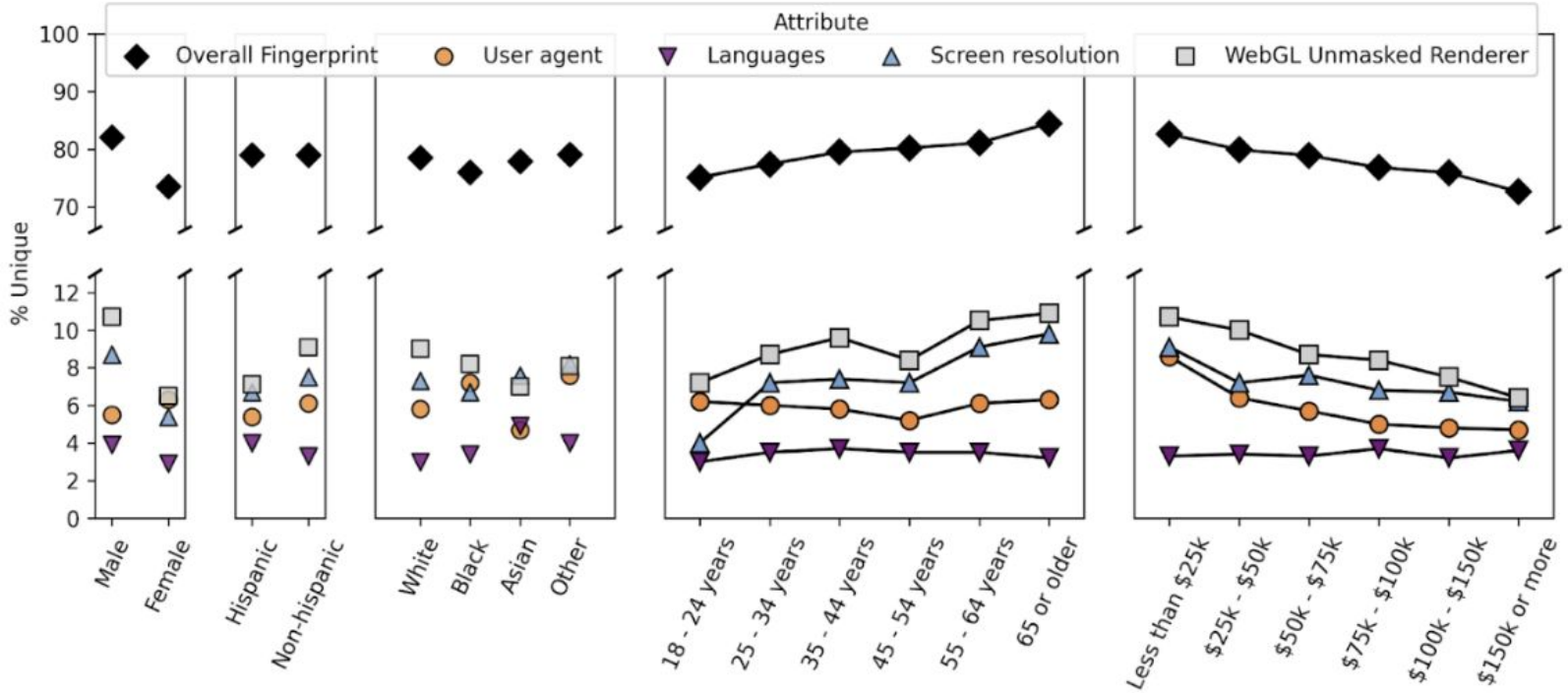
$$0.2241 \times 0.4319 \times 0.2153$$

$$= 0.0208$$

how likely one user is to have all three properties (2%)

(Simplified) Likelihood Calculations

- if every attribute is independent (they're not) you simply multiply probabilities
  - conditional probabilities slightly more complicated
    P(A and B) = P(A) * P(B | A)
- as you have seen, as attributes are added, the likelihood of one specific combination occurring becomes very rare
- i.e., the joint probabilities become very small
- even with large populations, individuals can be identified



"How Unique is Whose Web Browser? The Role of Demographics in Browser Fingerprinting among US Users" Alex Berke 2025