Active Fingerprinting

Ad Targeting and Surveillance Pricing

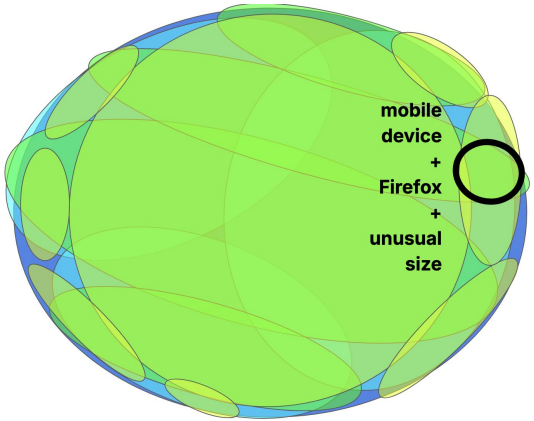
UC San Diego

Logistics…

- **Programming Assignment #2:** due November 19 (next Thurs)
- If you have questions about the exam, please come to office hours!

Recall...

- **stateful** vs. **stateless tracking:** stateful relies on data stored *on client* (which user can remove), while stateless stores nothing *on client*
- core intuition: attributes are shared by many users (even millions), but combinations of multiple attributes can be uncommon / uniquely identify



(Simplified) Likelihood Calculations

Attribute	
2 - Accept	22.41 %
3 - Content encoding	43.19 %
4 - Content language	21.53 %

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

- if every attribute is independent you simply multiply probabilities
- the likelihood of a combination of multiple attributes becomes rare

- attributes not always independent
  - conditional probabilities slightly more complicated
    P(A and B) = P(A) * P(B | A)

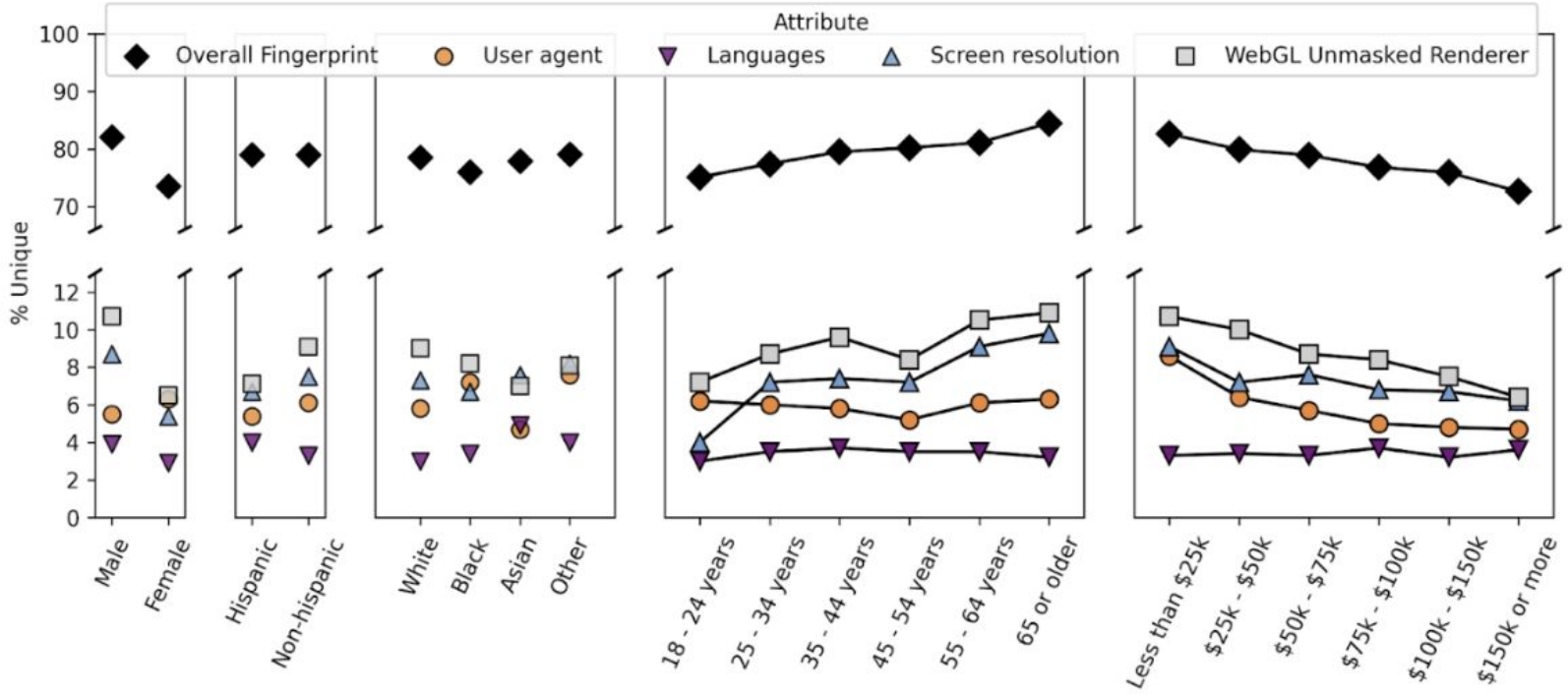
Windows	Apple
Segoe UI	San Francisco
Calibri	Helvetica Neue
Franklin Gothic	Menlo

(Simplified) Likelihood Calculations

- if every attribute is independent you simply multiply probabilities
- the likelihood of a combination of multiple attributes becomes rare

- attributes not always independent
  - conditional probabilities slightly more complicated
    $P(A \text{ and } B) = P(A) * P(B \mid A)$

Mobile	Desktop
720 × 1280	1920 × 1080
1080 × 2400	2560 × 1440
1440 × 3200	3840 × 2160



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

Active Fingerprinting

- **passive:** some data shared automatically by browser
- **active:** website runs scripts (using JS) to actively query browser, to read or compute specific properties

- vary from very simple (ex: screen resolution) to complex (audio fingerprinting)

Navigator

- **Navigator:** provides read-only information about browser, device, and operating environment; also provides interface to hardware& system APIs
- allows scripts to query it (& do some activities)

https://developer.mozilla.org/en-US/docs/Web/API/Navigator

Navigator

- **Navigator:** provides read-only information about browser, device, and operating environment; also provides interface to hardware& system APIs
- allows scripts to query it (& do some activities)
- provides access to:
- **user environment**: providing info about current settings
  ex: `navigator.userAgent, .language, .pdfViewerEnabled, .connection`
- **device hardware & sensors:** provide high level info like CPU cores
  ex: `navigator.geolocation, .hardwareConcurrency, .deviceMemory`
- **interfaces for other utilities:** allows web apps to interact with system utils
  ex: `navigator.clipboard, .share, .mediaDevices, .bluetooth`

Navigator Attributes

- **user environment**: providing info about current settings
  ex: navigator.userAgent, .language, .pdfViewerEnabled, .connection
- **PDF Viewer Enabled:** indicates whether browser supports inline display of PDF files (if false, PDF will be downloaded)
  values: true, false

```
const pdfView = navigator.pdfViewerEnabled;
```

Navigator Attributes

- **device hardware & sensors:** provide high level info like CPU cores
  ex: navigator.geolocation, .hardwareConcurrency, .deviceMemory
- **Hardware Concurrency:** the number of logical processor cores available to run tasks in the browser
  values: 1-32+ (4, 8, 12, 16 most common)

```
const cores = navigator.hardwareConcurrency;
```

Navigator Attributes

- **interfaces for other utilities:** allows web apps to interact with system utils ex: navigator.clipboard, .share, .mediaDevices, .bluetooth
- **Media Devices:** can provide a list of all connected media devices and their IDs (e.g., video cards, audio cards, headphones)
- ⚠ since it requires users to grant access, these are less frequently used for browser fingerprinting

Active Fingerprinting

- **Screen Resolution:** can reveal info about device type (e.g., smartphone vs. external monitors)

```
const width = screen.width;
const height = screen.height;
```



- can also use additional properties of the screen like `availWidth/Height`, `devicePixelRatio`, `colorDepth`

https://gs.statcounter.com/screen-resolution-stats/desktop/worldwide

Active Fingerprinting

- **Timezone:** reveals broad location information, easy way to divide individuals (split across zones)
- particularly identifying for those in less populated time zones

Eastern: 46.75%
Central: 28.91%
Mountain: 6.94%

Pacific: 16.75%
Hawaii-Aleutian: 0.43%
Alaska : 0.22%

https://andyarthur.org/p/157157

Active Fingerprinting

- **Timezone:** reveals broad location information, easy way to divide individuals (split across zones)
- particularly identifying for those in less populated time zones
- using Intl is preferred to Date or (newer, not fully supported) Temporal
- `Intl`: Javascript object that provides access to ECMAScript Internationalization API
  formats dates, numbers, currencies based on specific language/regions
- `.timeZone`: provides current timezone, offset from UTC

```
Intl.DateTimeFormat().resolvedOptions().timeZone
⚠ new Date(2026, 10, 10, 2, 30).toString()
⚠ Temporal.Now.timeZoneId()
```

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify



identical text rendered by
35 different browsers/devices

https://browserleaks.com/canvas

Font-based Fingerprinting

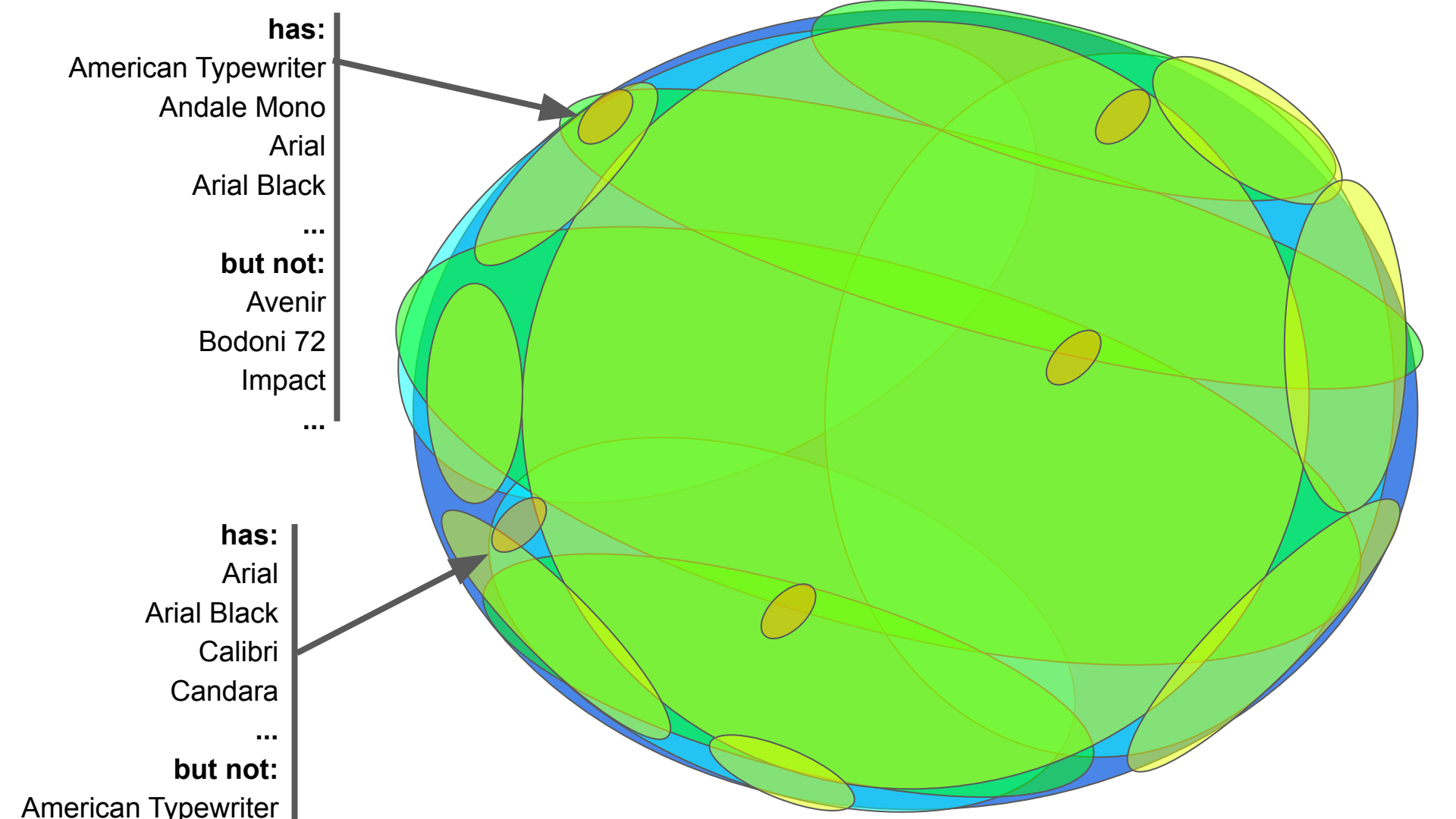
- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify

	User 10465...		User 10466...
Font Name	Present	Font Name	Present
American Typewriter	0	American Typewriter	1
Andale Mono	0	Andale Mono	0
Arial	1	Arial	1
Arial Black	1	Arial Black	1
Avenir	0	Avenir	1
Bodoni 72	1	Bodoni 72	1
Calibri	1	Calibri	1
Candara	1	Candara	0
Ebrima	1	Ebrima	0
Georgia	0	Georgia	0
Impact	1	Impact	1
Helvetica Neue	0	Helvetica Neue	1

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify

Font Name	User 10465...	User 10466...	User 10467...	User 10468...	User 10469...
American Typewriter	0	1	1	1	1
Andale Mono	0	0	0	0	0
Arial	1	1	1	1	1
Arial Black	1	1	0	0	1
Avenir	0	1	0	0	0
Bodoni 72	1	1	1	1	1
Calibri	1	1	1	1	1
Candara	1	0	0	1	1
Ebrima	1	0	0	1	0
Georgia	0	0	0	0	0
Impact	1	1	1	0	0
Helvetica Neue	0	1	1	1	0


has:
American Typewriter
Andale Mono
Arial
Arial Black
...
but not:
Avenir
Bodoni 72
Impact
...
has:
Arial
Arial Black
Calibri
Candara
...
but not:
American Typewriter

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- in the days of Flash (1996 → 2020) this was easy

  `Font.enumerateFonts()`

- now use properties of rendered text

Font-based Fingerprinting

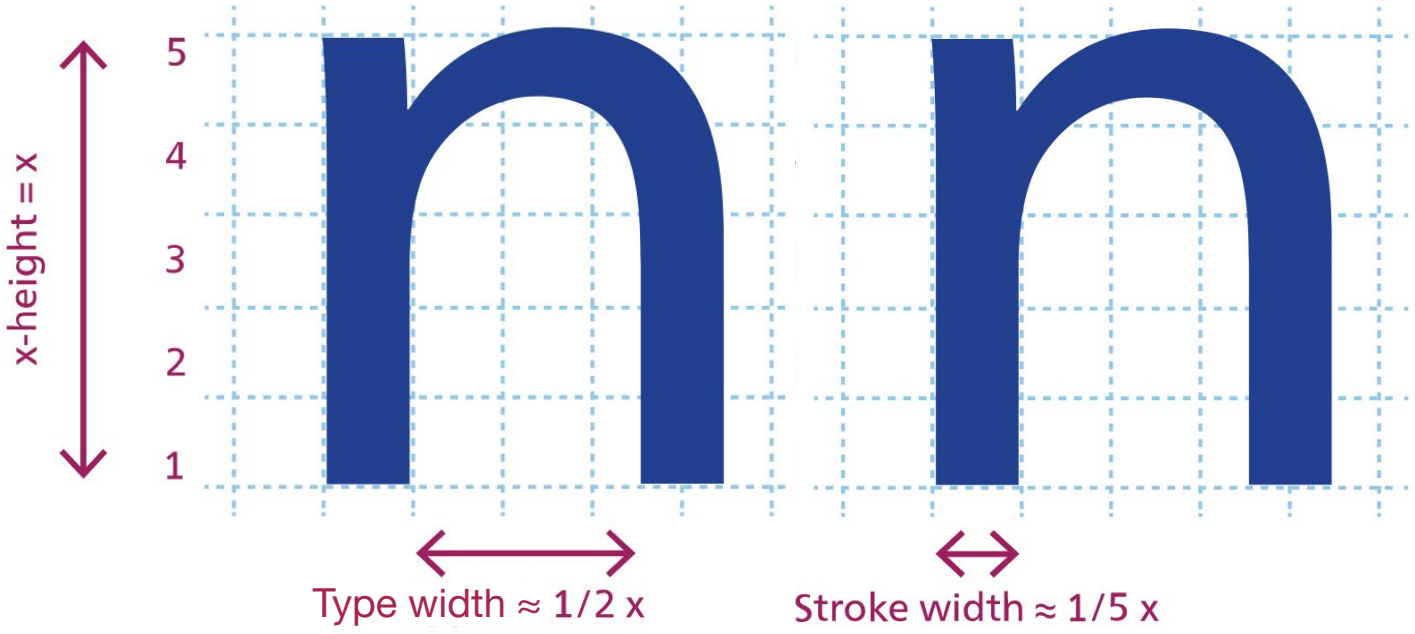
- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- in the days of Flash (1996 → 2020) this was easy

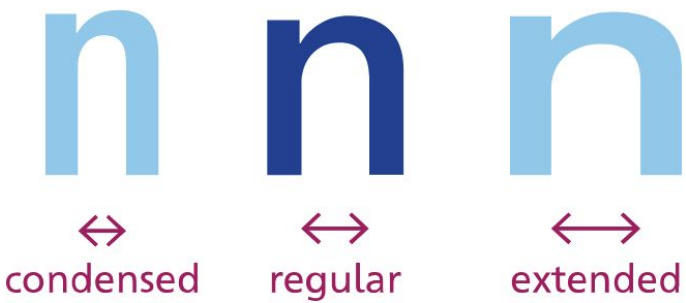
  `Font.enumerateFonts()`

- now use properties of rendered text

what properties of rendered text could we use for fingerprinting?

Type width

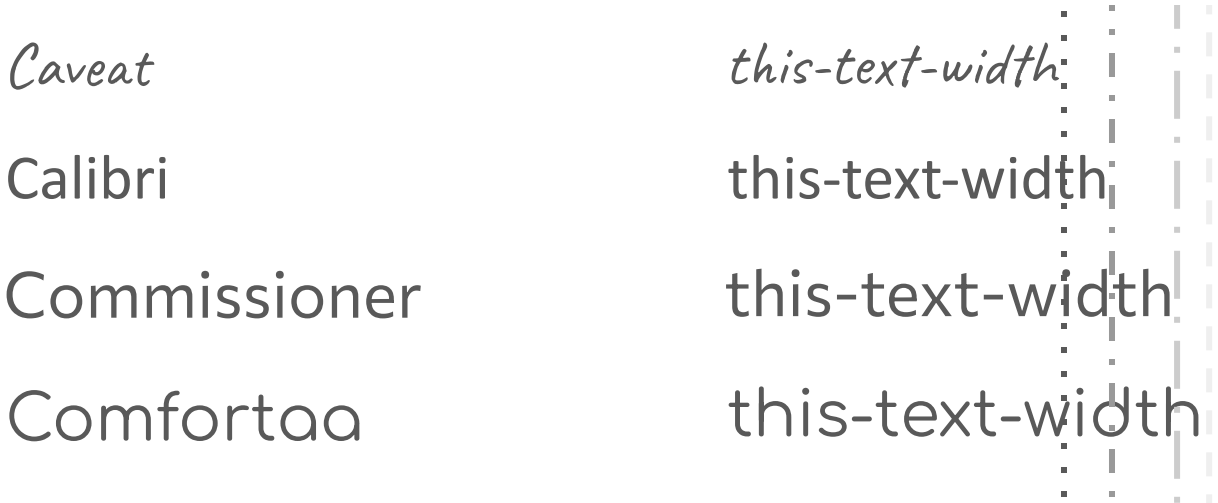




https://legibility.info

Type width

- since different fonts have different widths
- measure the width in browser to see if each one installed or not



note these are all size 24

Type width

- since different fonts have different widths
- measure the width in browser to see if each one installed or not

Caveat this-text-width

Calibri this-text-width



how can we use this to tell if a font is installed?

aka what happens if a font is not installed?

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- use properties of the rendered text, especially text width
- when a web page uses a font that is not installed, the browser substitutes with a "fallback" or default font

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- use properties of the rendered text, especially text width
- when a web page uses a font that is not installed, the browser substitutes with a "fallback" or default font

- using JS, specify the desired font, size, and a "fallback" font if the desired font is not installed

```
myTest.font = `18px 'Arial', 'sans-serif'`;
```

desired font

fallback

⚠ if you leave out size, browser will ignore entire line, default to 10px sans-serif

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- use properties of the rendered text, especially text width
- when a web page uses a font that is not installed, the browser substitutes with a "fallback" or default font
- *Steps:*
  1) pick a string to test the width of
  2) set the testing font and a fallback font
  3) calculate the width of the string, using the testing font *and* fallback

```
const testWidth = ctx.measureText(testString).width;
```

Font-based Fingerprinting

- many possible fonts (or extensions) installed, so many bits of information that can be used to uniquely identify
- use properties of the rendered text, especially text width
- when a web page uses a font that is not installed, the browser substitutes with a "fallback" or default font
- *Steps:*
  1) pick a string to test the width of
  2) set the testing font and a fallback font
  3) calculate the width of the string, using the testing font *and* fallback
  4) compare; if the two widths are the **same**, the font **is not** installed

how can you ensure you don't have a bunch of garbage text all over your website?





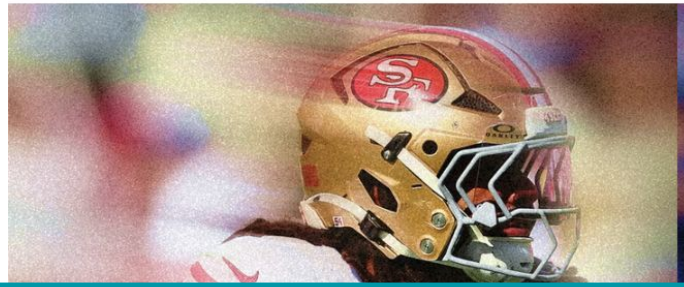
The five boxing wizards jump quickly.

The five boxing wizards jump quickly.

The five boxing wizards jump quickly.

The five boxing wizards jump quickly.

The five boxing wizards jump quickly.





Canvas

- <canvas> creates a space, used to draw graphics on the fly
- use scripting (typically Javascript) to add any content

- but **you don't need to attach** the <canvas> element **to the DOM** for fingerprinting to work!

Canvas

- <canvas> creates a space, used to draw graphics on the fly
- use scripting (typically Javascript) to add any content

- but **you don't need to attach** the <canvas> element **to the DOM** for fingerprinting to work!
- *Steps:*
  1) create the element *in memory*

     DO NOT append to document.body or any other element

```
const canvas = document.createElement('canvas')
document.body.appendChild(canvas);
```

Canvas

- <canvas> creates a space, used to draw graphics on the fly
- use scripting (typically Javascript) to add any content

- but **you don't need to attach** the <canvas> element **to the DOM** for fingerprinting to work!
- *Steps:*
  1) create the element *in memory* `document.createElement('canvas')`
  2) using context "draw" *also in memory* `ctx = getContext('2d')`
  3) measure / read values without ever adding to DOM or displaying `.measureText()` `.toDataURL()` `.toBlob()`

help George fix his code, so his fingerprinting isn't noticed by website visitors!

Easier



https://shorturl.at/kPCrA

More challenging



https://shorturl.at/kYb8j

Canvas

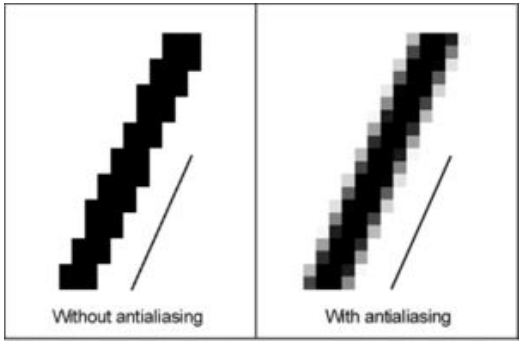
- <canvas> creates a space, used to draw graphics on the fly
- use scripting (typically Javascript) to add any content

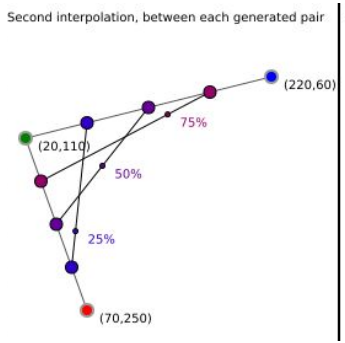
- but **you don't need to attach** the <canvas> element **to the DOM** for fingerprinting to work!
- *Steps:*
  1) create the element *in memory*
  2) using context "draw" *also in memory*
  3) measure / read values without ever adding to DOM or displaying

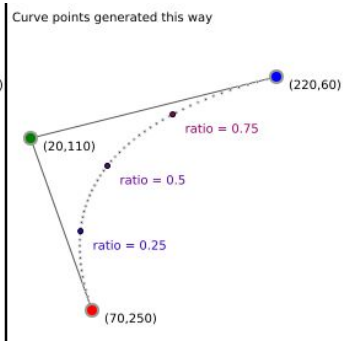
⚠ note this will make debugging any work-in-progress tricky, because you can't see it!

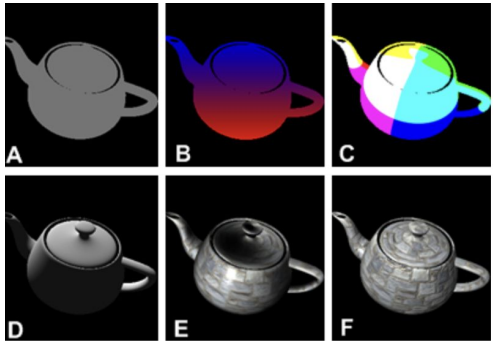
Graphics-based Fingerprinting

- intuition: graphics calculations rely heavily on floating-point arithmetic
- different parts (GPUs, CPUs, drivers, browsers) use different rounding rules and precision limits for floating-point arithmetic
- can trigger with:
  - anti-aliasing, gradient blends, bezier curves, shaders, color profiles









Scientific Notation

Sign Mantissa (or Significand) Exponent

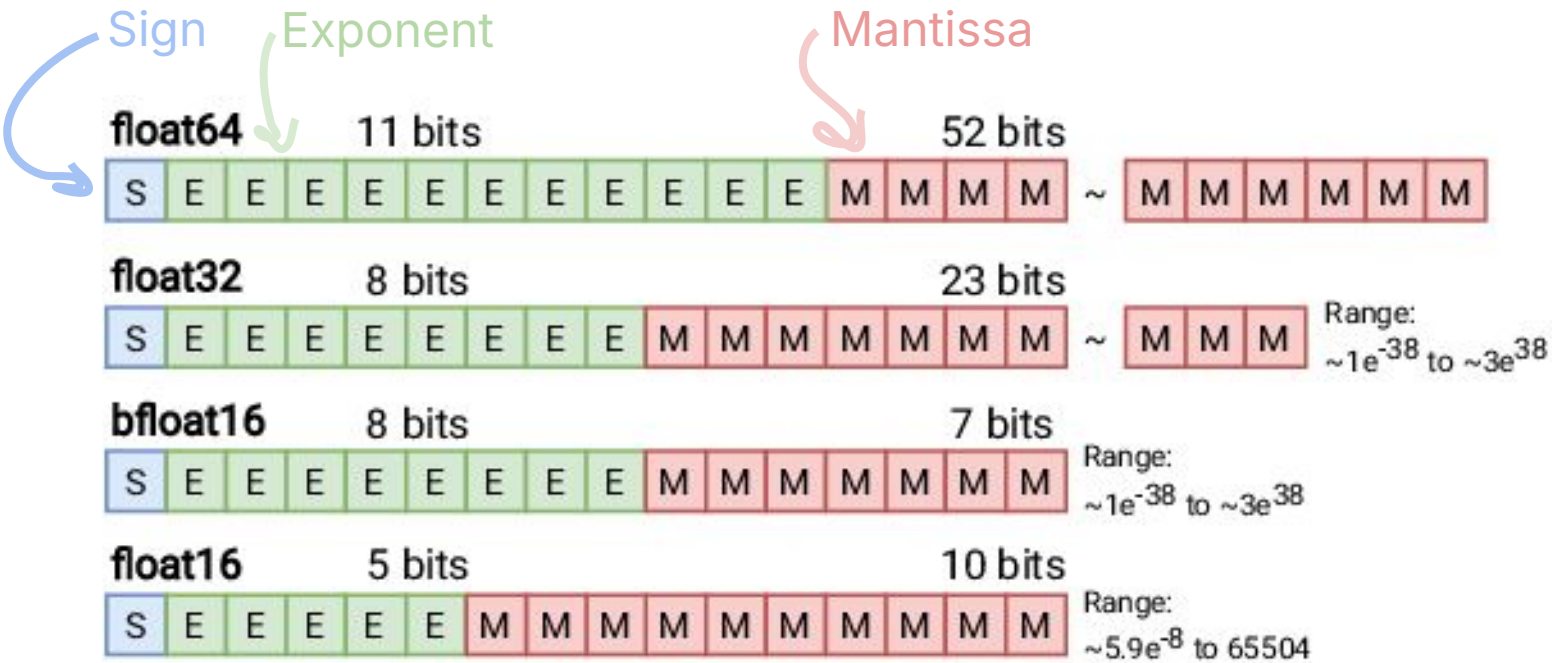
$$+\ 1.04 \times 10^{8}$$

Floating Point

Sign Mantissa (or Significand) Exponent

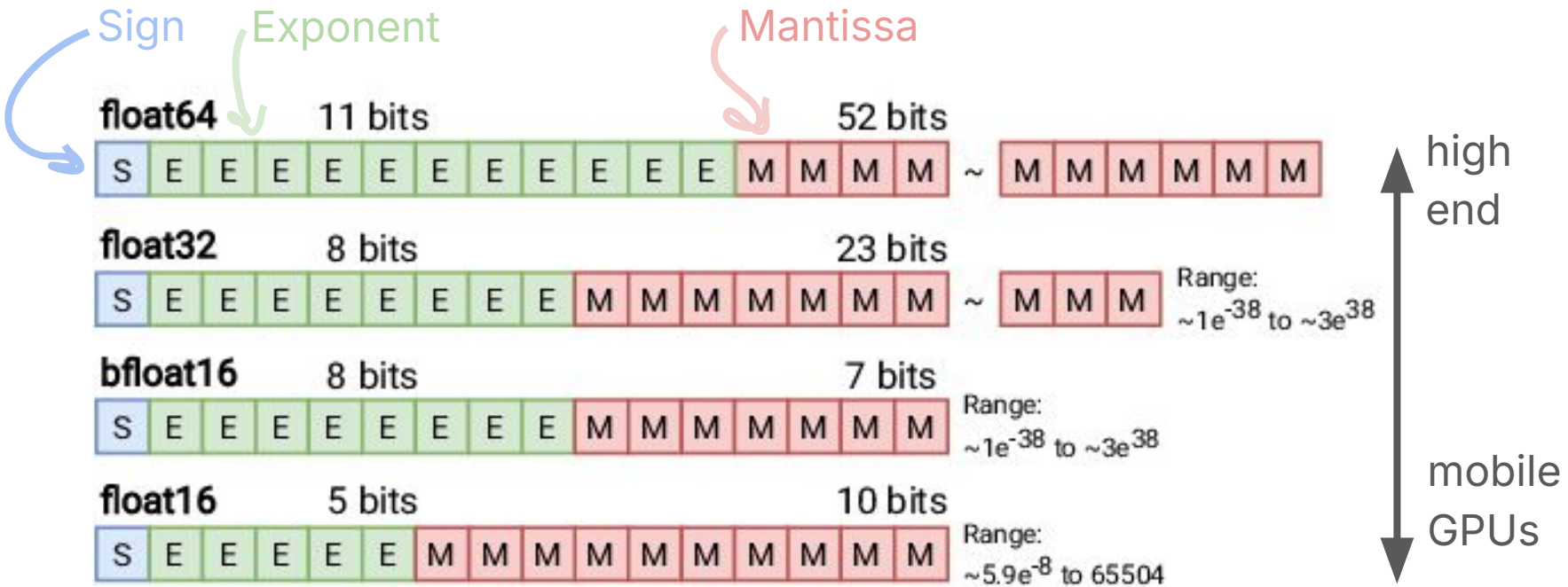
$$+\ 1.04 \times 2^{8}$$

Floating Point



"Deep Neural Networks to Correct Sub-Precision Errors in CFD" Akash Haridas 2022

Floating Point



Addition with Exact Values

$$
\begin{aligned}
A &= 2^{1} \times 1.00000000000000000000001 \\
B &= 2^{0} \times 1.00000000000000000000001 \\
C &= 2^{3} \times 1.00000000000000000000001 \\
(A+B)+C &= 2^{3} \times 1.01100000000000000000001011 \\
A+(B+C) &= 2^{3} \times 1.01100000000000000000001011
\end{aligned}
$$

$$
(A+B)+C \quad = \quad A+(B+C)
$$

"Floating Point and IEEE 754" Nathan Whitehead & Alex Fit-Florea (NVIDIA) 2026

Addition with Round-to-Nearest

$$
\begin{aligned}
A + B &= 2^1 \times 1.10000000000000000000110000\ldots \\
\mathsf{rn}(A + B) &= 2^1 \times 1.10000000000000000000010 \\
B + C &= 2^3 \times 1.00100000000000000000100100\ldots \\
\mathsf{rn}(B + C) &= 2^3 \times 1.00100000000000000000001 \\
A + B + C &= 2^3 \times 1.01100000000000000000101100\ldots \\
\mathsf{rn}\left(\mathsf{rn}(A + B) + C\right) &= 2^3 \times 1.01100000000000000000010 \\
\mathsf{rn}\left(A + \mathsf{rn}(B + C)\right) &= 2^3 \times 1.01100000000000000000001
\end{aligned}
$$

$$(A + B) + C \quad \neq \quad A + (B + C)$$

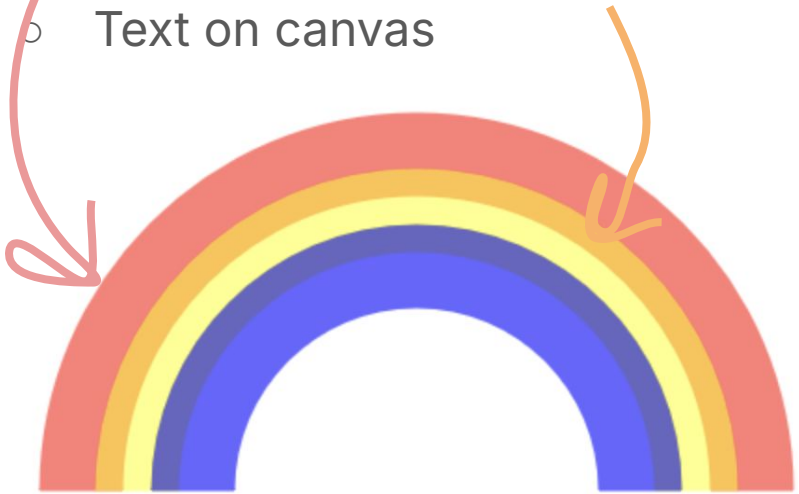
"Floating Point and IEEE 754" Nathan Whitehead & Alex Fit-Florea (NVIDIA) 2026

Graphics-based Fingerprinting

- intuition: graphics calculations rely heavily on floating-point arithmetic
- different parts (GPUs, CPUs, drivers) use different rounding rules and precision limits for floating-point arithmetic
- can trigger with:
  - anti-aliasing, gradient blends, bezier curves, shaders, color profiles
- differences occur due to:
  - floating point precision (16, 32, 64 bit)
  - order of operations
  - round-to-nearest vs. round-towards-zero
  - multiply-then-add vs. fused multiply-add (FMA)
  - ...

Graphics-based Fingerprinting

- Good triggers:
  - Curved shapes
  - Transparency/overlapping (alpha values)
  - Text on canvas



```
3 arcs (red, yellow, blue)
overlapping transparency
```

Some Useful Canvas Drawing Methods

- `getContext()`: returns an object with properties/methods for drawing

- those include
  `beginPath()`: declares we are about to draw (without drawing)
  `moveTo(x,y)`: sets start point in canvas (without drawing)
  `lineTo(x,y)`: sets end point in canvas (without drawing)
  `stroke()`: draws the line

- can also use
  `rect()`
  `arc()`



3 arcs (red, yellow, blue)
overlapping transparency

```
const centerX = canvas.width / 2; const centerY = [...]
const startAngle = Math.PI; const endAngle = 2 * Math.PI;
const rainbowStripeWidth = 30; const overlapWidth = [...]
const radiusRed = 180; const radiusYellow = [...]
const redColor = 'rgba(255, 0, 0, 0.6)'; const yellowColor [...]
function drawArc(radius, color, width){
 ctx.beginPath();
 ctx.arc(centerX, centerY, radius, startAngle, endAngle, false);
 ctx.lineWidth = width; ctx.strokeStyle = color;
 ctx.stroke();
drawArc(radiusRed, redColor, overlapWidth); drawArc( [...]
```



3 arcs (red, yellow, blue)
overlapping transparency

```
const centerX = canvas.width
const startAngle = Math.PI; c
const rainbowStripeWidth = 30
const radiusRed = 180; const radiusYellow = [...]
const redColor = 'rgba(255, 0, 0, 0.6)'; const yellowColor [...]
function drawArc(radius, color, width){
 ctx.beginPath();
 ct                                    ngle, endAngle, false);
 ct                                     color;
 ct
drawArc(radiusRed, redColor, overlapWidth); drawArc( [...]
```

good sign for floating point arithmetic differences using irrational numbers!

transparency in alpha values requires multiplying/adding channels (FP arithmetic)

Encodings 2.0

- `toDataURL()`: a built in method that converts current canvas into a Base64-encoded string representation of the image


```
"
Zr2n8qlSSeEBHJPJ9BACARINEDAC6LIggozK7N79LiOzjde+JwdWAZRv1l3xjMzR13HzxEZFmbHT1F3dGY9eHbFV
VBhAEVEJEiUAC1JIB2Szj3k3p0r9Z230ikDhCTdXbfuen/n1ElXhUP9qzpP1f/yvs8rgFEVEUgDYAOQ7v1JW5L8D
8xJAwAXgGrvT5cA1JjzVmiDoM1pQh8R0r2cBmDqZaKODP2rV4TeQdF7HwDlAM4J0rOSCRQWup+IQKxX2NO9P2kLl
/8BowT9JHbvdtYr/Hb5t8yYYaGPERFIvUzY9NZmtIfe9kOFX6t9E4IPFvpVEAe63LOHbMnyLxkjUQ/g9OAmDAwBm
MtgoQ9BHBhTDwo7h+9P0EHjeccQ0dN4n+E/ZEnc1wCY6xV3Iv9VhBSNXtGfFIBvQurKfMSUb3QRyAWQ792i5bvBh
DJuAEW0CUBxKF/ocJhG6CIwZ4i4x8t3gDEjHUNEf8oMNyCkhS4OjLMHxT3BDF8o4zNtQ0RP4/uQJOSELgITAVzn3
WhJjGHGCkXnfQngiAA0y0dDgJARundSjcQ9LwS+F0Z/jnsFHxKTeEEtdHEgZnzw7c0z5owa0Mz9Ea/oKUY/KAlKo
YtAHoCbMPCTYbTiBIDPBEg/g4qgEroI3AjgZgBTguouM6FGBYBPBeDzYLkwwwtdHFjnJnHTRhNtDGMUaMLuU6/oa
Z3esBhW6CKQMkTgEYa9gwwD9A0RfJ0Rb4jhhC4CdgB/A6DAiDeMYUahEMBfBcApHzEAhhG6CGQCWA5gkQHuC8MEC
q3H7xWASvmIjugudHEgr5sEfoOO94Fh1OKwV/CUR68bugldHMjvXuFdJmOYUOczAHsESPnzmqO50EUgwSvwWzS/W
obRn0+8gm/SsimaCl0E7gBwp5my5hhmGMgg4z0B+FA+ojKaCN2b/70aQIZ8ZoZhqgDs1CI/XlWhezPJSOAUi84wz
PBQLD0JXrWMOdWELg6shZPI2decYUaHTC1J7H+VjyiI4kIXgVnecThZIjMM4xtkZU3j92/lIwqgmNDFAXsmeoPzc
pnWTEIkJiECExGJ8QhHJMIQCcH7c+hnAeEIk5rXj4vohYjeS35+97kD/WhGLy6gDxfYQlkHaDmO3vBkexUwighdH
Fgqo7c4Gy0qCQkzC1ZMQTQmw4oUWDEBEYiRBB2JaEQgWqOhkRu9cKMPHehFJ3rRhj7UwYPz8KACbpTBIz0kGCWhR
Bl6u9OSXEAEJHPxoGeJycWz5YOM76IDggleMdtgPPgsSFI0EhFl/5tgoBEdaIOb1fDA5Y0IlEv7PcHOfANI4n27U
```

Encodings 2.0

- `toDataURL()`: a built in method that converts current canvas into a Base64-encoded string representation of the image
- hashing this into a shorter version makes it easier to send & compare


```
"
Zr2n8qlSSeEBHJPJ9BACARINEDAC6LIggozK7N79LiOzjde+JwdWAZRv1l3xjMzR13HzxEZFmbHT1F3dGY9eHbFV
VBhAEVEJEiUAC1JIB2Szj3k3p0r9Z230ikDhCTdXbfuen/n1ElXhUP9qzpP1f/yvs8rgFEVEUgDYAOQ7v1JW5L8D
8xJAwAXgGrvT5cA1JjzVmiDoM1pQh8R0r2cBmDqZaKODP2rV4TeQdF7HwDlAM4J0rOSCRQWup+IQKxX2NO9P2kLl
/8BowT9JHbvdtYr/Hb5t8yYYaGPERFIvUzY9NZmtIfe9kOFX6t9E4IPFvpVEAe63LOHbMnyLxkjUQ/g9OAmDAwBm
MtgoQ9BHBhTDwo7h+9P0EHjeccQ0dN4n+E/ZEnc1wCY6xV3Iv9VhBSNXtGfFIBvQurKfMSUb3QRyAWQ792i5bvBh
DJuAEW0CUBxKF/ocJhG6CIwZ4i4x8t3gDEjHUNEf8oMNyCkhS4OjLMHxT3BDF8o4zNtQ0RP4/uQJOSELgITAVzn3
WhJjGHGCkXnfQngiAA0y0dDgJARundSjcQ9LwS+F0Z/jnsFHxKTeEEtdHEgZnzw7c0z5owa0Mz9Ea/oKUY/KAlKo
YtAHoCbMPCTYbTiBIDPBEg/g4qgEroI3AjgZgBTguouM6FGBYBPBeDzYLkwwwtdHFjnJnHTRhNtDGMUaMLuU6/oa
Z3esBhW6CKQMkTgEYa9gwwD9A0RfJ0Rb4jhhC4CdgB/A6DAiDeMYUahEMBfBcApHzEAhhG6CGQCWA5gkQHuC8MEC
q3H7xWASvmIjugudHEgr5sEfoOO94Fh1OKwV/CUR68bugldHMjvXuFdJmOYUOczAHsESPnzmqO50EUgwSvwWzS/W
obRn0+8gm/SsimaCl0E7gBwp5my5hhmGMgg4z0B+FA+ojKaCN2b/70aQIZ8ZoZhqgDs1CI/XlWhezPJSOAUi84wz
PBQLD0JXrWMOdWELg6shZPI2decYUaHTC1J7H+VjyiI4kIXgVnecThZIjMM4xtkZU3j92/lIwqgmNDFAXsmeoPzc
pnWTEIkJiECExGJ8QhHJMIQCcH7c+hnAeEIk5rXj4vohYjeS35+97kD/WhGLy6gDxfYQlkHaDmO3vBkexUwighdH
```

WebGL Fingerprinting

- `<canvas>` can render both 2d and 3d graphics

- as in 2d case, get canvas context: `canvas.getContext('webgl')`
- as in 2d case, render some shape
  - additional steps required, like adding shaders
- as in 2d case, extract and hash drawing data with `toDataURL()`

WebGL Fingerprinting

- Some useful API interfaces/methods:

- `Float32Array`, `Int16Array...`: ⚠ WebGL functions **require** typed arrays, array literals must be converted
- `.createBuffer()`, `.bufferData()`: creates a buffer to store the binary data that will be uploaded to the GPU (coordinates, textures, etc.)
- `.createShader()`, `.compileShader()`, `.attachShader()`: calculate positions and colors for placement of 3d data on 2d screen (clip space), including lighting, shadows, color gradients, etc.
- `.drawArrays()`: renders from the buffer

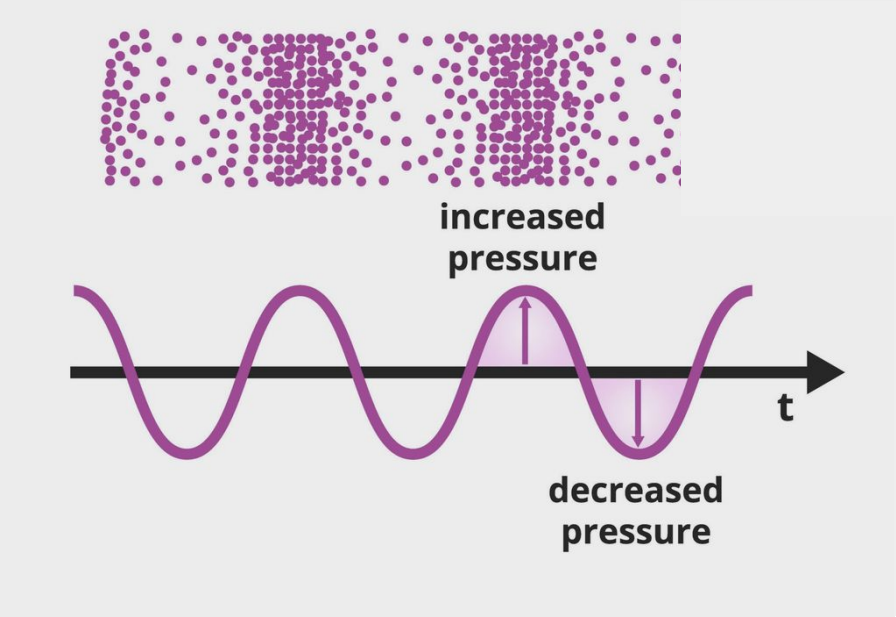
https://webgl2fundamentals.org/webgl/lessons/webgl-fundamentals.html

Audio Fingerprinting

- intuition: audio production *also* relies heavily on floating-point arithmetic
- in audio, different devices will *also* have different rounding rules and precision limits for floating-point arithmetic

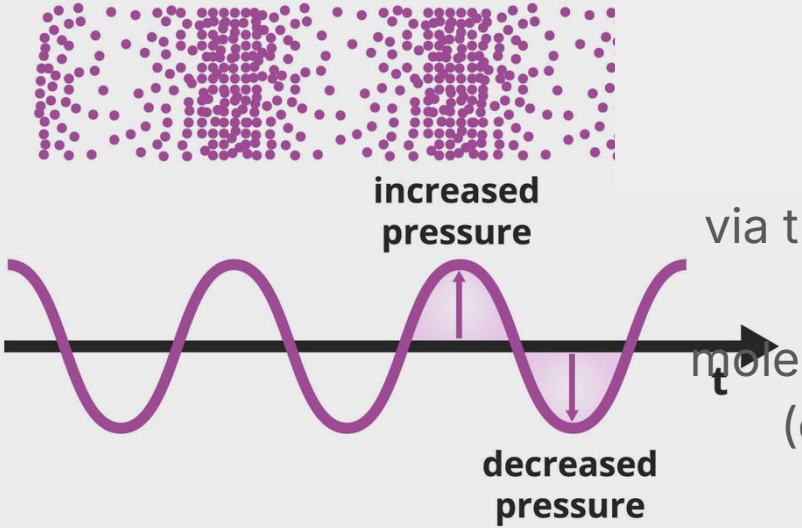
What is sound?

sound is produced by compressing & decompressing air (compression & rarefaction)



What is sound?

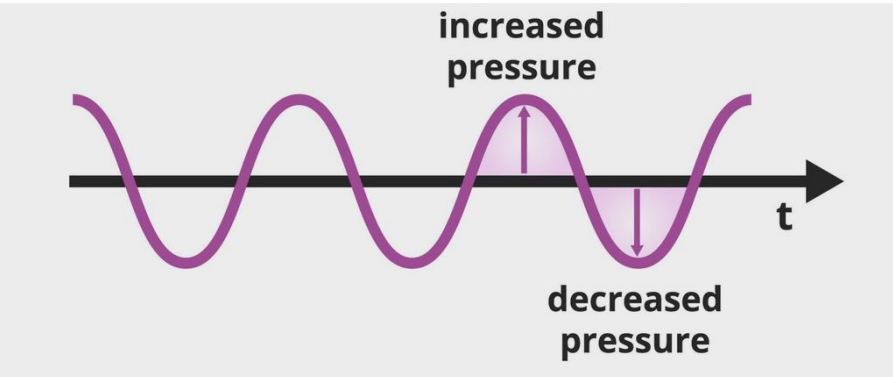
sound is produced by compressing & decompressing air (compression & rarefaction)



sound is measured via the pressure created by the motion of air molecules at destination (e.g., microphone or ear)

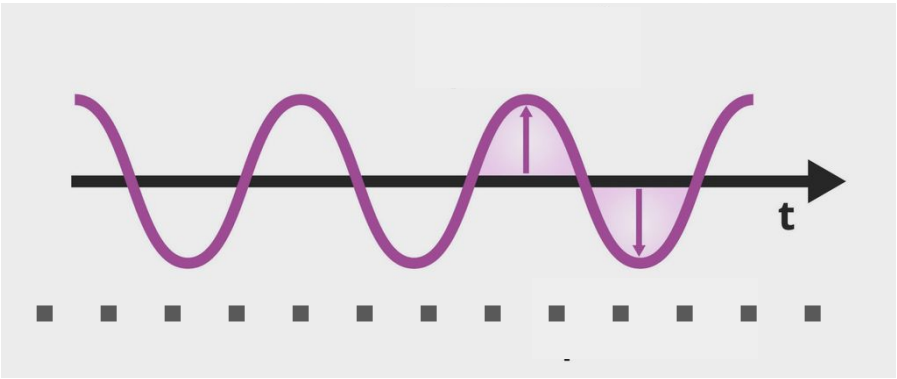
Pulse Code Modulation (PCM) Data

- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



Pulse Code Modulation (PCM) Data

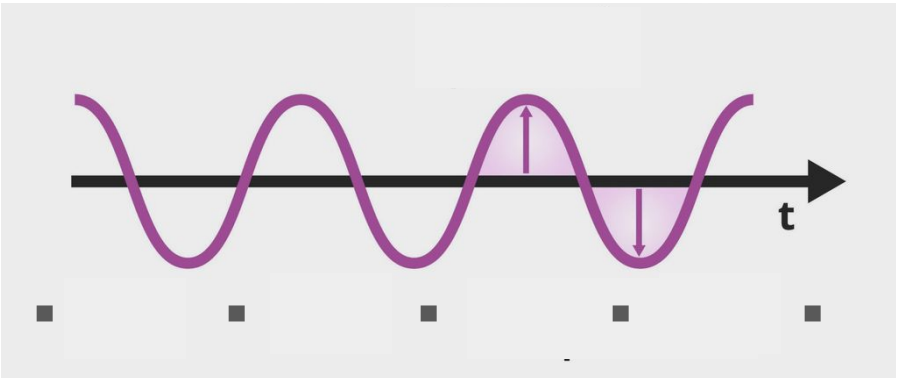
- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



sampling rate: the resolution in time (how often you measure per second)

Pulse Code Modulation (PCM) Data

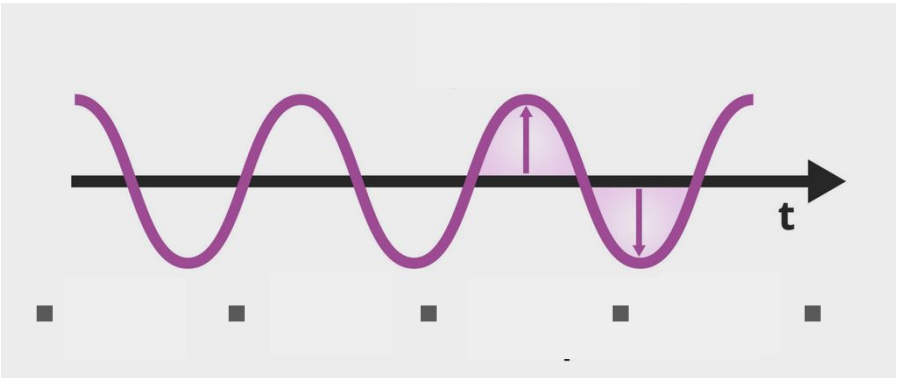
- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



sampling rate: the resolution in time (how often you measure per second)

Pulse Code Modulation (PCM) Data

- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary

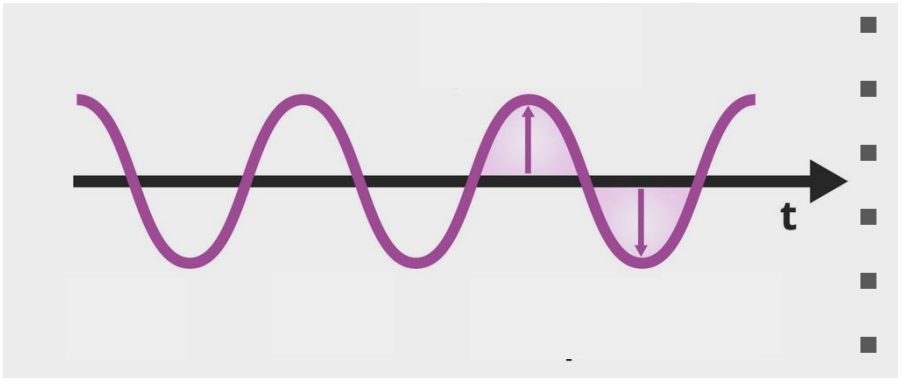


sampling rate: the resolution in time (how often you measure per second)

44,100 s/sec (44.1kHz) - audio
48,000 s/sec (48kHz) - video

Pulse Code Modulation (PCM) Data

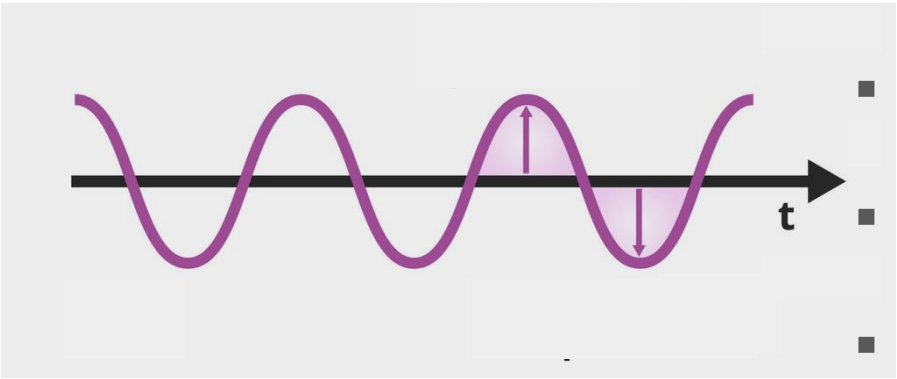
- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



bit depth: the resolution of amplitude (range of values available to store each sample)

Pulse Code Modulation (PCM) Data

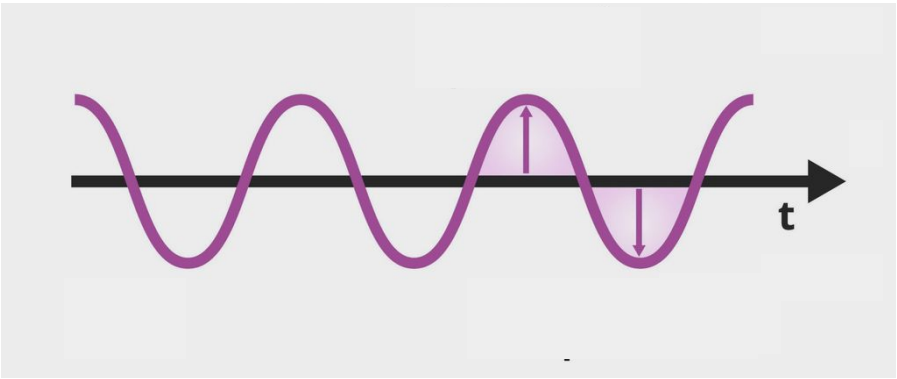
- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



bit depth: the resolution of amplitude (range of values available to store each sample)

Pulse Code Modulation (PCM) Data

- standard digital representation of analog (here, audio) signal
- 1) **Sample:** measure continuous wave at frequent, regular time intervals
- 2) **Quantize:** round measured height to nearest value on scale
- 3) **Encode:** represent in binary



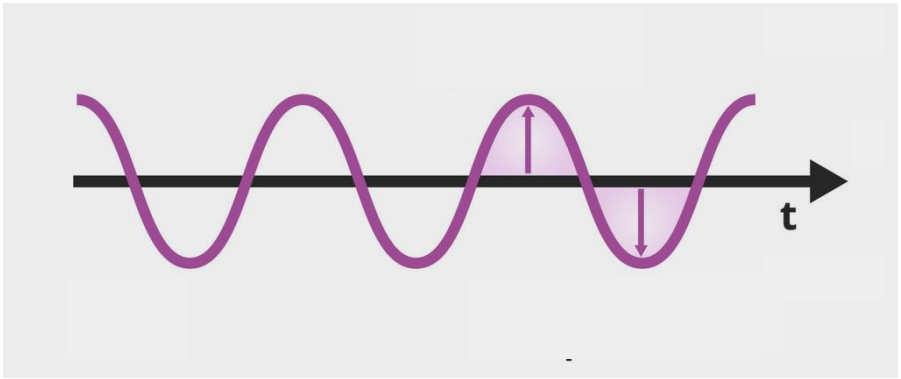
bit depth: the resolution of amplitude (range of values available to store each sample)

sampling rate: the resolution in time (how often you measure per second)

Pulse Code Modulation (PCM) Data

- standard digital representation
- 1) **Sample:** measure continuou
- 2) **Quantize:** round measured
- 3) **Encode:** represent in binary

can all be manipulated by someone fingerprinting your device with audio



bit depth: the resolution of amplitude (range of values available to store each sample)

sampling rate: the resolution in time (how often you measure per second)

Audio Fingerprinting

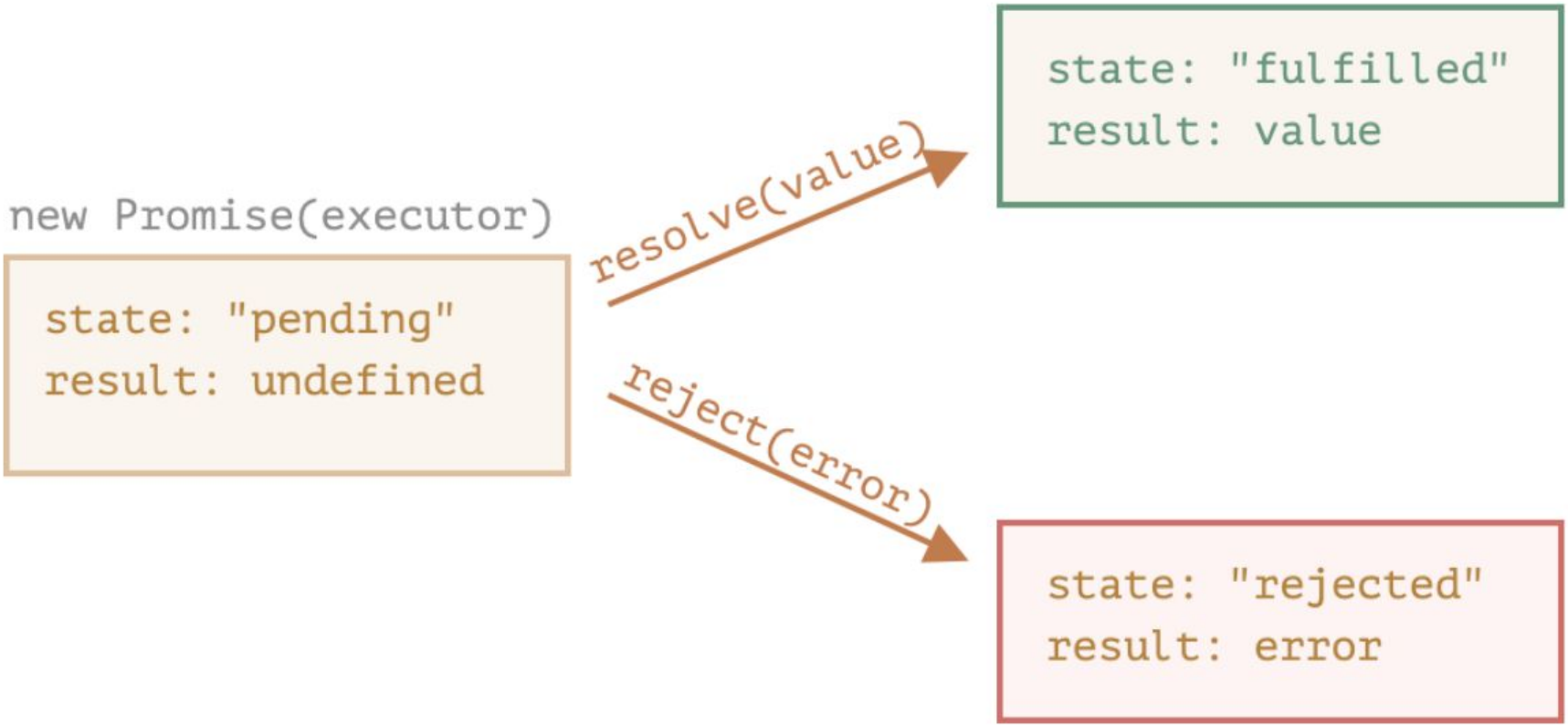
- Some useful API interfaces/methods:

- `OfflineAudioContext` will create audio signal in memory, but will not play through any speakers
- `.createOscillator()`: a built in method that creates a periodic waveform (single, constant tone), can set shape, frequency, duration
- `.startRendering()`: a built in method that will render the audio signal; this is a Promise that returns an `AudioBuffer` asset
- `.getChannelData()`: method that returns the digital representation of uncompressed analog signal, `0` is first (by convention, left) channel

Promises

- Rendering the audio signal takes more time than graphics rendering
- `.startRendering()` method uses a `Promise()`

- `Promise:` an object that helps manage asynchronous functions
  - has a status and a value
  - status will be either:
    - ✅ fulfilled
    - ❌ rejected
    - ⏳ pending
  - think of it like a container that promises it will have a value eventually


new Promise(executor)
state: "pending"
result: undefined
resolve(value)
reject(error)
state: "fulfilled"
result: value
state: "rejected"
result: error

Promises

- Rendering the audio signal takes more time than graphics rendering
- `.startRendering()` method uses a `Promise()`

- `Promise:` an object that helps manage asynchronous functions
  - to use a promise after it is fulfilled, either use
    - async / await ← preferred for readability
    - Promise chaining using .then()

```
async function myFingerprint(){
    const response = await [...].startRendering();
}
```

how long of an audio clip should a fingerprinter produce?

Audio Fingerprinting

- intuition: audio production *also* relies heavily on floating-point arithmetic
- in audio, different devices will *also* have different rounding rules and precision limits for floating-point arithmetic
- due to the demands of the targeted advertising context, generated audio will typically be 10s of milliseconds in length

https://amiunique.org/

https://coveryourtracks.eff.org/

https://panopticlick.org/about/

https://browserleaks.com/javascript