Technical analysis of browser fingerprinting techniques based on FingerprintJS

James Bergfeld
Technical University Munich
Munich, Germany
j.bergfeld@tum.de

Samuel Scheit
Technical University Munich
Munich, Germany
tum@samuelscheit.com

Contents

1. Introduction

what goes here: short intro: what is a fingerprint? what is tracking and how does it work? keep brief.

main question:

- "How does modern browser fingerprinting work (in practice)?"

why our paper:

other academic work focuses on proposing, identifying or improving individual browser properties that can be used to fingerprint browsers.

relevance:

- with the deprecation of third-party cookies, a rising demand to find other ways of tracking users to analyze web traffic can be seen. Many content delivery and advertising companies previously relied on these cookies to track their users.
- to get a realistic view of how fingerprinting actually works, we take a look at a modern fingerprinting library and analyze its client-side operation of data collection.
- FingerprintJS is currently the most widely-used fingerprinting library on npm

key points:

- reverse-engineering the FingerprintJS demo website gives a direct look at the practical implementation of browser fingerprinting technology
- types of data collected by the demo website (don't?) match the properties outlined in prominent literature
- by deploying a website that collects the same data as FPJS, a data set of fingerprints may be built
- collected data can be used to devise and test possible fingerprinting algorithms

1. General

Websites use browser fingerprinting to create a unique identifier of each website visitor by collecting data about the visitor's device and browser settings and combining them into a unique "fingerprint."

The aim is for website operators to identify users across multiple website visits without them having to actively accept cookies or log in with their user accounts.

2. Advantages

The purpose is to create a detailed profile of each user to display personalized content, serve advertising or analyze user behavior. This can be used both to improve the user experience and to detect fraudulent activity.

3. Disadvantages

In order to create a unique browser fingerprint, extensive information about a user's devices and browser settings must be collected. However, this violates the user's privacy unless they have explicitly agreed. Especially since there is no way to opt out of fingerprinting and the data can be used to track users across multiple websites. This allows websites to create a comprehensive profile of a person's online activities and draw conclusions about a person's identity and behavior.

4. Relevance

Because website operators require unique user profiles, even without users' consent, to provide personalized content and to analyze user behavior, browser fingerprinting has become an important tool. This is evidenced by the fact that 30.6% of the top 1k websites in the Alexa ranking use fingerprinting techniques. [1]

Since the majority of all browsers deactivate third-party cookies by default in the future[1], or need explicit consent to use third-party cookies, browser fingerprinting is a significant alternative to identify users across different websites.

5. Application

In order to assign a unique identity or "fingerprint" to each user, various details are collected via the browser. For example, a combination of rare fonts, a specific screen resolution, or a specific browser plugin can help generate a unique fingerprint.

JavaScript libraries can be used for this, such as *FingerprintJS* (FPJS), which collects a variety of information about a user's browser environment. In the commercial version, FPJS claims to be able to create a 99.5% unique fingerprint. [2]

FPJS is the most popular JavaScript browser fingerprinting library according to npm downloads.[2]

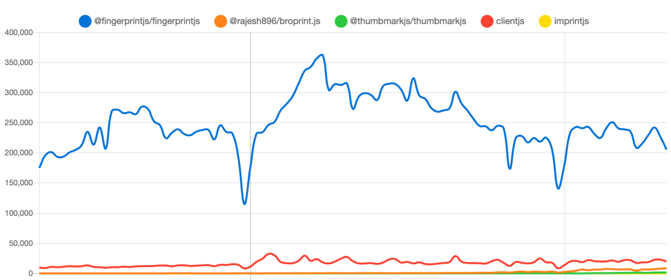


Figure 1: NPM downloads per day, comparison of different JS fingerprinting libraries (as of 2024)

2. Background

our paper is closely related to [3], we update data used, look at actual implementations and devise a similar though less extensive algorithm to parse data

3. Methodology

[1]https://developer.mozilla.org/en-US/blog/goodbye-third-party-cookies/

[2]https://npmtrends.com/@fingerprintjs/fingerprintjs-vs-@rajesh896/broprint.js-vs-@thumbmarkjs/thumbmarkjs-vs-clientjs-vs-imprintjs

4. Results

1. Parameters

1) Statistical properties

- window.devicePixelRatio
- window.screen
- window.indexedDB
- window.requestFileSystem
- window.screen.colorDepth
- window.screen.width
- window.screen.height
- window.sessionStorage
- window.localStorage
- window.indexedDB
- navigator.deviceMemory
- navigator.hardwareConcurrency
- navigator.storage.estimate()
- navigator.storage.getDirectory()
- navigator.doNotTrack
- navigator.webdriver
- navigator.oscpu
- navigator.languages
- navigator.onLine
- navigator.platform
- navigator.plugins
- navigator.vendor
- navigator.language
- navigator.languages
- navigator.pdfViewerEnabled
- navigator.webdriver
- navigator.userAgent
- navigator.appVersion
- navigator.connection.rtt

2) Integrity

3) Fonts

4) TLS

5) Audio

The Web Audio API is a browser API that can be used to artificially generate sounds and audio data.

However it can also be used create a unique audio profile of the browser by:

1. Generating a series of tone signals with predefined properties such as frequency, volume and distortion.
2. Playing and recording the sound at the same time via the Web Audio API.
3. Analyzing the recorded audio data and encoding as a hash to create a unique audio fingerprint.

FPJS first creates a triangle oscillator tone signal with a frequency of `10.000` Hz. Then a compressor is created with the following parameters:

Value	Description
`-50db`	“value above which the compression will start taking effect”
`40db`	“value representing the range above the threshold where the curve smoothly transitions to the compressed portion.”
`12db`	“amount of change needed in the input for a 1 dB change in the output.”
`0s`	“the amount of time required to reduce the gain by 10 dB.”
`0.25s`	“the amount of time required to increase the gain by 10 dB.”

The open source FPJS version uses a square oscillator with a base frequency of `1.000` Hz and an additional bi-quad filter. This can be visualized by the following diagram that compares the audio values of different browser implementations:

Figure 2: Audio API browser comparison

Due to subtle differences in audio processing and playback of different browsers and systems, the recorded data will vary slightly from the original.

To prevent fingerprint Firefox has the ability to disable the audio API.

6) **WebRTC**

Web Real-Time Communication (WebRTC) is a browser API used to transmit video-/audio data in realtime over a (optionally peer-to-peer) connection.

a) **IP Address**

Interactive Connectivity Establishment (ICE) is used in WebRTC to establish connections between clients that may be behind different network configurations or firewalls. This is achieved by connecting to a STUN (Session Traversal Utilities for NAT) server which resolves possible ICE candidates (public IP address and port of the device). Additionally the browser exposes the local IP address of the device's local area network (LAN) to enable local connections in intranets. This information can be retrieved by creating a new `RTCPeerConnection` with a specified ICE server and a unique username to correlate the STUN connection with the current browser session.
By adding an `icecandidate` event listener, the ICE candidates can be retrieved. The following string is an example candidate:

```
candidate:2079771436 1 udp 2122260223 123.234.1.250
50012 typ host generation 0 ufrag qRGm network-id 3
```

The candidate includes the IP address, port, network transport protocol, a unique identifier and other key value parameters.
Specifically the local IP address can be used to recognize a device even if the public IP address changes e.g. when using a Virtual Private Network (VPN).
For this reason the TOR Browser has disabled the WebRTC protocol and the Brave Browser has the ability to disable the usage of LAN IP addresses for WebRTC.
However, it should be noted local IP addresses are not unique and different LAN subnets have a limited address room. Specifically, 17.891.328 IPv4 addresses are reserved for LAN networks and similar subnets and IP addresses are reused on many different networks and therefore can only be used for fingerprinting in conjunction with other parameters.

b) **Codecs**

Additionally the supported audio and video codecs can further help to fingerprint a device as different Browser and Device configurations support different codecs. The `RTCPeerConnection` created in the previous step can be queried via `connection.getStats()` and returns a `RTCStatsReport`, which contains statistics of used audio and video codecs for the connection. For example the VP8 video codec is represented as the following object:

```
id: "HjD6dszXj",
type: "codec",
clockRate: 90000,
mimeType: "video/VP8",
direction: "sendrecv",
uri: "urn:ietf:params:rtp-hdrext:toffset",
```

and contains various information about the supported audio and video codecs e.g. support for CPU acceleration, forward error correction, stereo audio, bit-rate, codec version, frame size and other codec specific parameters.
These parameters are partially stable as browser updates might add support for different codecs, but processor specific codec acceleration does not change without a hardware modification.

c) **Media devices**

WebRTC media devices are audio and video sources of the browser as well as audio playback and video display devices. These can be microphones, cameras, speakers and screens. WebRTC allows websites to access these devices via the `MediaDevices` API.

The `navigator.mediaDevices.enumerateDevices()` API returns a "list of the currently available media input and output devices". Each media device contains the following properties:

- `deviceId` (unique and persistent device identifier)
- `groupId` (optional identifier that groups multiple ids of the same physical device)
- `kind` (`"videoinput"`,`"audioinput"`,`"audiooutput"`)
- `label` (optional human readable name for the device)

Note that the all device properties except `kind` are `null` if the website has never requested a media stream before.
FPJS uses this to determine the amount of audio an video devices the user has connected. As most websites don't use the media stream, the devices don't have a unique identifier and the media devices are a weak indicator for a unique fingerprint.

7) **Speech synthesis**

SpeechSynthesis is part of the Web Speech Browser API that allows websites to convert text to audio data (so-called Text-to-speech or TTS). The browser exposes the function `SpeechSynthesis.getVoices()` that lists all locally and remotely available voices that can be used for TTS.
Each voice contains the following properties:

- `voiceURI` (unique voice identifier)
- `name` (human-readable name of the voice)
- `lang` (ISO language code of the voice)
- `localService` (boolean indicating if the voice is locally available or a remote service)
- `default` (boolean indicating if the voice is set as default)

FPJS converts this list of voices to a string with `JSON.stringify` and then hashes it with `Murmurhash3_128_x64` [4]. Additionally FPJS also sends a boolean indicating if any `"Google"` voices are installed on the system. As browsers return the list in order this hash is stable and only changes when the browser or the user adds a new voice to their system. However this hash only identifies specific browser versions and operating systems and is not unique. Firefox prevents this when `resistFingerprinting` is enabled by returning an empty list.

8) **Canvas**

Canvas is a browser API that allows websites to display dynamic 2D graphics. However it can also be used to create a unique identifier for the user's graphic engine.

Canvas fingerprinting works by using the Canvas API to draw text, shapes, and images onto a canvas element and then extracting the pixel data to create a unique identifier. This identifier is based on subtle differences in the way browsers and devices render the same graphics instructions.

1. **Text Rendering:** By rendering specific text onto a hidden canvas element, the browser's font rendering and antialiasing techniques contribute to the uniqueness of the fingerprint.
2. **Shape Drawing:** Drawing shapes and applying transformations (scaling, rotation, etc.) can reveal details about the graphics rendering engine and hardware acceleration capabilities.
3. **Image Manipulation:** Using images and manipulating them at a pixel-level level can reveal information about image processing algorithms and rendering accuracy.

FPJS uses the canvas API to render the following text, emojis and geometry:

The pixel data is then retrieved by calling `canvas.toDataURL()` and hashed using `Murmurhash3_128_x64` [4]. However browsers such as Brave or Firefox add noise to the retrieved canvas data. To verify if canvas noise is added FPJS calls `toDataURL()` twice and compares the resulting buffers. Additionally FPJS uses an embedded image to check if the PNG image data returned by `toDataURL()` matches the data of the embedded image.
If one of the checks fail noise was added to the canvas and the resulting hash is always unique per session and therefore unusable for identification without any further parameters.

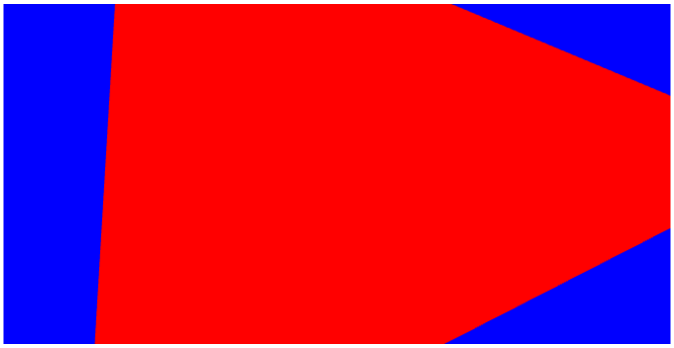
9) **WebGL**

The WebGL (Web Graphics Library) is an additional API on top of the canvas element that allows websites to render 3D graphics, shaders and can also be used to create a unique identifier of the graphics engine.

a) **Rendering**

By rendering specific shaders and geometric shapes the GPU capabilities for texturing and rendering complexity can uniquely be identified.

FPJS uses WebGL fingerprinting by rendering the following graphic:



with the following shaders:

```
attribute vec2 p;
uniform float t;
void main() {
  float s = sin(t);
  float c = cos(t);
  gl_Position = vec4(p * mat2(c, s, -s, c), 1, 1);
}
```

```
void main() { gl_FragColor = vec4(1, 0, 0, 1); }
```

The data is retrieved, hashed and verified in the same way as with the Canvas API.

b) **Extensions**

Additionally the GPU capabilities can be queried by calling `context.getSupportedExtensions()`, `context.getContextAttributes()`, `context.getParameter()` and `context.getExtension()` functions of the `WebGLRenderingContext`-API.

The list of all queried WebGL extensions and parameters by FPJS are available as an attachment in Section 8.1.

FPJS then concatenates the result of the queries and creates a hash over the following categories of WebGL parameters:

```
contextAttributes:
"6b1ed336830d2bc96442a9d76373252a",
parameters: "57a2cddb99538d50a0138430ed0720c5",
parameters2: "7bd4d913de3e22461894a997d864dcb8",
shaderPrecisions:
"f223dfbcd580cf142da156d93790eb83",
extensions: "57233d7b10f89fcd1ff95e3837ccd72d",
extensionParameters:
"fa430f89faf2af23f701c2c6909bcaad",
extensionParameters2:
"86a8abb36f0cb30b5946dec0c761d042",
```

and extracts the following plaintext parameters:

```
version: "WebGL 1.0 (OpenGL ES 2.0 Chromium)",
vendor: "WebKit",
vendorUnmasked: "Google Inc. (Apple)",
renderer: "WebKit WebGL",
rendererUnmasked: "ANGLE (Apple, ANGLE Metal
Renderer: Apple M1 Ultra, Unspecified Version)",
shadingLanguageVersion: "WebGL GLSL ES 1.0 (OpenGL ES
GLSL ES 1.0 Chromium)",
```

2. **Comparison to open-source FingerprintJS**

3. **Parameter weights**

Type definitions:

- Parameter weights: $\{(\text{name}, \text{stability}, \text{uniqueness})\}$
  $\text{stability}, \text{uniqueness} \in [0, 1]$
- Fingerprint: $\{(\text{name}, \text{value})\}$
- $\text{overlap} \in [0, 1]$
  score for each db entry that describes how accurately said entry matches a fingerprint

Given:

- Database of weights for parameters P
- Fingerprint f
- Collection of existing fingerprints C
- $c_p = \langle c \in C \mid c.\text{name} = p \rangle$
- $c_{\text{p+v}} = \langle c \in C \mid \text{c.name} = \text{p} \land \text{c.value} = \text{v} \rangle$
- $C_p = \{c \in c_p\}$
- $C_{\text{p+v}} = \{c \in c_{\text{p+v}}\}$

$$\forall p \in P \mid p_{\text{stability}} = \mathbb{E}\left[C_p\right] = \sum_{\text{v} \in C_p} \left(\frac{\left|c_{\text{p+v}}\right|}{\left|c_p\right|}\right)^2$$

$$\forall p \in P \mid p_{\text{uniqueness}} = \mathbb{E}\left[I_{C_p}\right]$$

We generate a fingerprint:

define match function $m(a, b) := \begin{cases} 0 \text{ if a} \neq b \\ 1 \text{ otherwise} \end{cases}$

for each $c \in C$ we calculate the parameter match set:
$M_{f,c} = \{(f_n, m(f_v, c_v)) \mid (f_n, f_v) \in f, (c_n, c_v) \in c, f_n = c_n\}$

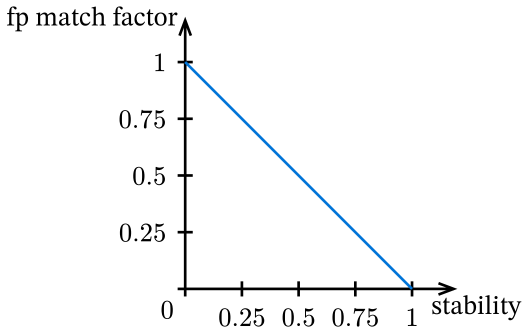
$$\text{overlap}\ \left(M_{f,c}\right) = \frac{\prod_{p \in M_{f,c}} \text{algorithm}(p)}{2}$$

$$\text{stability(s, x)} = (1 - s) \cdot (x + 1) + 2 \cdot s \cdot x$$

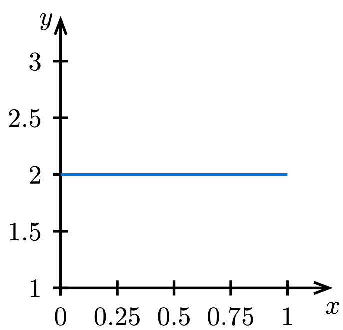
$$\text{uniqueness(u, x)} = (1 - x) + (1 + u) \cdot x$$

$$\text{algorithm}(s, u, x) = \text{stability(s, x)} \cdot \text{uniqueness(u, x)}$$

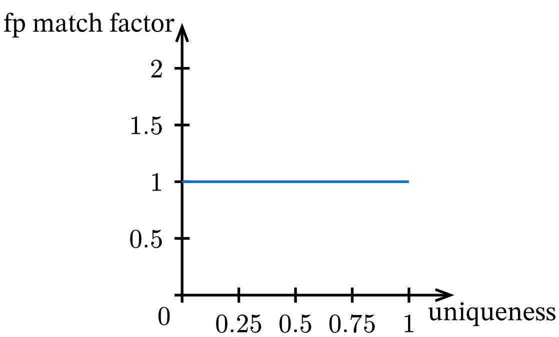
Values don't match



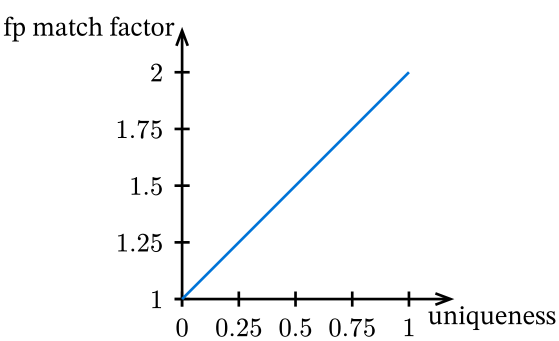
Values match



Values don't match



Values match



5. Discussion

6. Conclusion

7. References

[1] Z. S. Umar Iqbal Steven Englehardt, "Fingerprinting the Fingerprinters: Learning to Detect Browser Fingerprinting Behaviors." [Online]. Available: https://arxiv.org/pdf/2008.04480

[2] [Online]. Available: https://fingerprint.com/

[3] P. Eckersley, "How Unique Is Your Web Browser?," in *Privacy Enhancing Technologies*, M. J. Atallah and N. J. Hopper, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 1–18.

[4] [Online]. Available: https://github.com/aappleby/smhasher

[5] N. N. J. P. Konstantinos Solomos Panagiotis Ilia, "Escaping the Confines of Time: Continuous Browser Extension Fingerprinting Through Ephemeral Modifications." [Online]. Available: https://www.cs.uic.edu/~polakis/papers/solomos-ccs22.pdf

[6] A. K. Junhua Su, "Automatic Discovery of Emerging Browser Fingerprinting Techniques." [Online]. Available: https://www.kapravelos.com/publications/fptechniques-www23.pdf

[7] [Online]. Available: https://github.com/fingerprintjs/fingerprintjs

8. ATTACHMENTS

1. WebGL Attributes

```
  contextAttributes: [
alpha=true
antialias=true
depth=true
desynchronized=false
failIfMajorPerformanceCaveat=false
powerPreference=default
premultipliedAlpha=true
preserveDrawingBuffer=false
stencil=false
xrCompatible=false
]
parameters: [
ACTIVE_ATTRIBUTES=35721
ACTIVE_TEXTURE=34016=33984
ACTIVE_UNIFORMS=35718
ALIASED_LINE_WIDTH_RANGE=33902=11
ALIASED_POINT_SIZE_RANGE=33901=1511
ALPHA=6406
ALPHA_BITS=3413=8
ALWAYS=519
ARRAY_BUFFER=34962
ARRAY_BUFFER_BINDING=34964
ATTACHED_SHADERS=35717
BACK=1029
BLEND=3042=false
BLEND_COLOR=32773=0000
BLEND_DST_ALPHA=32970=0
BLEND_DST_RGB=32968=0
BLEND_EQUATION=32777=32774
BLEND_EQUATION_ALPHA=34877=32774
BLEND_EQUATION_RGB=32777=32774
BLEND_SRC_ALPHA=32971=1
BLEND_SRC_RGB=32969=1
BLUE_BITS=3412=8
BOOL=35670
BOOL_VEC2=35671
BOOL_VEC3=35672
BOOL_VEC4=35673
BROWSER_DEFAULT_WEBGL=37444
BUFFER_SIZE=34660
BUFFER_USAGE=34661
BYTE=5120
CCW=2305
CLAMP_TO_EDGE=33071
COLOR_ATTACHMENT0=36064
COLOR_BUFFER_BIT=16384
COLOR_CLEAR_VALUE=3106=0000
COLOR_WRITEMASK=3107=truetruetruetrue
COMPILE_STATUS=35713
COMPRESSED_TEXTURE_FORMATS=34467=
CONSTANT_ALPHA=32771
CONSTANT_COLOR=32769
CONTEXT_LOST_WEBGL=37442
CULL_FACE=2884=false
CULL_FACE_MODE=2885=1029
CURRENT_PROGRAM=35725
CURRENT_VERTEX_ATTRIB=34342
CW=2304
DECR=7683
DECR_WRAP=34056
DELETE_STATUS=35712
DEPTH_ATTACHMENT=36096
DEPTH_BITS=3414=24
DEPTH_BUFFER_BIT=256
DEPTH_CLEAR_VALUE=2931=1
DEPTH_COMPONENT16=33189
DEPTH_COMPONENT=6402
DEPTH_FUNC=2932=513
DEPTH_RANGE=2928=01
DEPTH_STENCIL=34041
DEPTH_STENCIL_ATTACHMENT=33306
DEPTH_TEST=2929=false
DEPTH_WRITEMASK=2930=true
DITHER=3024=true
DONT_CARE=4352
DST_ALPHA=772
DST_COLOR=774
DYNAMIC_DRAW=35048
ELEMENT_ARRAY_BUFFER=34963
ELEMENT_ARRAY_BUFFER_BINDING=34965
EQUAL=514
FASTEST=4353
FLOAT=5126
FLOAT_MAT2=35674
FLOAT_MAT3=35675
FLOAT_MAT4=35676
FLOAT_VEC2=35664
FLOAT_VEC3=35665
FLOAT_VEC4=35666
FRAGMENT_SHADER=35632
FRAMEBUFFER=36160
FRAMEBUFFER_ATTACHMENT_OBJECT_NAME=36049
FRAMEBUFFER_ATTACHMENT_OBJECT_TYPE=36048
FRAMEBUFFER_ATTACHMENT_TEXTURE_CUBE_MAP_FACE=36051
FRAMEBUFFER_ATTACHMENT_TEXTURE_LEVEL=36050
FRAMEBUFFER_BINDING=36006
FRAMEBUFFER_COMPLETE=36053
FRAMEBUFFER_INCOMPLETE_ATTACHMENT=36054
FRAMEBUFFER_INCOMPLETE_DIMENSIONS=36057
FRAMEBUFFER_INCOMPLETE_MISSING_ATTACHMENT=36055
FRAMEBUFFER_UNSUPPORTED=36061
FRONT=1028
FRONT_AND_BACK=1032
FRONT_FACE=2886=2305
FUNC_ADD=32774
FUNC_REVERSE_SUBTRACT=32779
FUNC_SUBTRACT=32778
GENERATE_MIPMAP_HINT=33170=4352
GEQUAL=518
GREATER=516
GREEN_BITS=3411=8
HIGH_FLOAT=36338
HIGH_INT=36341
IMPLEMENTATION_COLOR_READ_FORMAT=35739=6408
IMPLEMENTATION_COLOR_READ_TYPE=35738=5121
INCR=7682
INCR_WRAP=34055
INT=5124
INT_VEC2=35667
INT_VEC3=35668
INT_VEC4=35669
INVALID_ENUM=1280
INVALID_FRAMEBUFFER_OPERATION=1286
INVALID_OPERATION=1282
INVALID_VALUE=1281
INVERT=5386
KEEP=7680
LEQUAL=515
LESS=513
LINEAR=9729
LINEAR_MIPMAP_LINEAR=9987
LINEAR_MIPMAP_NEAREST=9985
LINES=1
LINE_LOOP=2
LINE_STRIP=3
LINE_WIDTH=2849=1
LINK_STATUS=35714
```

LOW_FLOAT=36336
LOW_INT=36339
LUMINANCE=6409
LUMINANCE_ALPHA=6410
MAX_COMBINED_TEXTURE_IMAGE_UNITS=35661=32
MAX_CUBE_MAP_TEXTURE_SIZE=34076=16384
MAX_FRAGMENT_UNIFORM_VECTORS=36349=1024
MAX_RENDERBUFFER_SIZE=34024=16384
MAX_TEXTURE_IMAGE_UNITS=34930=16
MAX_TEXTURE_SIZE=3379=16384
MAX_VARYING_VECTORS=36348=30
MAX_VERTEX_ATTRIBS=34921=16
MAX_VERTEX_TEXTURE_IMAGE_UNITS=35660=16
MAX_VERTEX_UNIFORM_VECTORS=36347=1024
MAX_VIEWPORT_DIMS=3386=1638416384
MEDIUM_FLOAT=36337
MEDIUM_INT=36340
MIRRORED_REPEAT=33648
NEAREST=9728
NEAREST_MIPMAP_LINEAR=9986
NEAREST_MIPMAP_NEAREST=9984
NEVER=512
NICEST=4354
NONE=0
NOTEQUAL=517
NO_ERROR=0
ONE=1
ONE_MINUS_CONSTANT_ALPHA=32772
ONE_MINUS_CONSTANT_COLOR=32770
ONE_MINUS_DST_ALPHA=773
ONE_MINUS_DST_COLOR=775
ONE_MINUS_SRC_ALPHA=771
ONE_MINUS_SRC_COLOR=769
OUT_OF_MEMORY=1285
PACK_ALIGNMENT=3333=4
POINTS=0
POLYGON_OFFSET_FACTOR=32824=0
POLYGON_OFFSET_FILL=32823=false
POLYGON_OFFSET_UNITS=10752=0
RED_BITS=3410=8
RENDERBUFFER=36161
RENDERBUFFER_ALPHA_SIZE=36179
RENDERBUFFER_BINDING=36007
RENDERBUFFER_BLUE_SIZE=36178
RENDERBUFFER_DEPTH_SIZE=36180
RENDERBUFFER_GREEN_SIZE=36177
RENDERBUFFER_HEIGHT=36163
RENDERBUFFER_INTERNAL_FORMAT=36164
RENDERBUFFER_RED_SIZE=36176
RENDERBUFFER_STENCIL_SIZE=36181
RENDERBUFFER_WIDTH=36162
RENDERER=7937=WebKit WebGL
REPEAT=10497
REPLACE=7681
RGB565=36194
RGB5_A1=32855
RGB8=32849
RGB=6407
RGBA4=32854
RGBA8=32856
RGBA=6408
SAMPLER_2D=35678
SAMPLER_CUBE=35680
SAMPLES=32937=4
SAMPLE_ALPHA_TO_COVERAGE=32926
SAMPLE_BUFFERS=32936=1
SAMPLE_COVERAGE=32928
SAMPLE_COVERAGE_INVERT=32939=false
SAMPLE_COVERAGE_VALUE=32938=1
SCISSOR_BOX=3088=00300150
SCISSOR_TEST=3089=false
SHADER_TYPE=35663
SHADING_LANGUAGE_VERSION=35724=WebGL GLSL ES 1.0 (OpenGL ES GLSL ES 1.0 Chromium)
SHORT=5122
SRC_ALPHA=770
SRC_ALPHA_SATURATE=776
SRC_COLOR=768
STATIC_DRAW=35044
STENCIL_ATTACHMENT=36128
STENCIL_BACK_FAIL=34817=7680
STENCIL_BACK_FUNC=34816=519
STENCIL_BACK_PASS_DEPTH_FAIL=34818=7680
STENCIL_BACK_PASS_DEPTH_PASS=34819=7680
STENCIL_BACK_REF=36003=0
STENCIL_BACK_VALUE_MASK=36004=2147483647
STENCIL_BACK_WRITEMASK=36005=2147483647
STENCIL_BITS=3415=0
STENCIL_BUFFER_BIT=1024
STENCIL_CLEAR_VALUE=2961=0
STENCIL_FAIL=2964=7680
STENCIL_FUNC=2962=519
STENCIL_INDEX8=36168
STENCIL_PASS_DEPTH_FAIL=2965=7680
STENCIL_PASS_DEPTH_PASS=2966=7680
STENCIL_REF=2967=0
STENCIL_TEST=2960=false
STENCIL_VALUE_MASK=2963=2147483647
STENCIL_WRITEMASK=2968=2147483647
STREAM_DRAW=35040
SUBPIXEL_BITS=3408=4
TEXTURE0=33984
TEXTURE10=33994
TEXTURE11=33995
TEXTURE12=33996
TEXTURE13=33997
TEXTURE14=33998
TEXTURE15=33999
TEXTURE16=34000
TEXTURE17=34001
TEXTURE18=34002
TEXTURE19=34003
TEXTURE1=33985
TEXTURE20=34004
TEXTURE21=34005
TEXTURE22=34006
TEXTURE23=34007
TEXTURE24=34008
TEXTURE25=34009
TEXTURE26=34010
TEXTURE27=34011
TEXTURE28=34012
TEXTURE29=34013
TEXTURE2=33986
TEXTURE30=34014
TEXTURE31=34015
TEXTURE3=33987
TEXTURE4=33988
TEXTURE5=33989
TEXTURE6=33990
TEXTURE7=33991
TEXTURE8=33992
TEXTURE9=33993
TEXTURE=5890
TEXTURE_2D=3553
TEXTURE_BINDING_2D=32873
TEXTURE_BINDING_CUBE_MAP=34068
TEXTURE_CUBE_MAP=34067
TEXTURE_CUBE_MAP_NEGATIVE_X=34070
TEXTURE_CUBE_MAP_NEGATIVE_Y=34072
TEXTURE_CUBE_MAP_NEGATIVE_Z=34074
TEXTURE_CUBE_MAP_POSITIVE_X=34069

TEXTURE_CUBE_MAP_POSITIVE_Y=34071
TEXTURE_CUBE_MAP_POSITIVE_Z=34073
TEXTURE_MAG_FILTER=10240
TEXTURE_MIN_FILTER=10241
TEXTURE_WRAP_S=10242
TEXTURE_WRAP_T=10243
TRIANGLES=4
TRIANGLE_FAN=6
TRIANGLE_STRIP=5
UNPACK_ALIGNMENT=3317=4
UNPACK_COLORSPACE_CONVERSION_WEBGL=37443=37444
UNPACK_FLIP_Y_WEBGL=37440=false
UNPACK_PREMULTIPLY_ALPHA_WEBGL=37441=false
UNSIGNED_BYTE=5121
UNSIGNED_INT=5125
UNSIGNED_SHORT=5123
UNSIGNED_SHORT_4_4_4_4=32819
UNSIGNED_SHORT_5_5_5_1=32820
UNSIGNED_SHORT_5_6_5=33635
VALIDATE_STATUS=35715
VENDOR=7936=WebKit
VERSION=7938=WebGL 1.0 (OpenGL ES 2.0 Chromium)
VERTEX_ATTRIB_ARRAY_BUFFER_BINDING=34975
VERTEX_ATTRIB_ARRAY_ENABLED=34338
VERTEX_ATTRIB_ARRAY_NORMALIZED=34922
VERTEX_ATTRIB_ARRAY_POINTER=34373
VERTEX_ATTRIB_ARRAY_SIZE=34339
VERTEX_ATTRIB_ARRAY_STRIDE=34340
VERTEX_ATTRIB_ARRAY_TYPE=34341
VERTEX_SHADER=35633
VIEWPORT=2978=00300150
ZERO=0
]
shaderPrecisions: [
FRAGMENT_SHADER.LOW_FLOAT=12712723
FRAGMENT_SHADER.MEDIUM_FLOAT=12712723
FRAGMENT_SHADER.HIGH_FLOAT=12712723
FRAGMENT_SHADER.LOW_INT=31300
FRAGMENT_SHADER.MEDIUM_INT=31300
FRAGMENT_SHADER.HIGH_INT=31300
VERTEX_SHADER.LOW_FLOAT=12712723
VERTEX_SHADER.MEDIUM_FLOAT=12712723
VERTEX_SHADER.HIGH_FLOAT=12712723
VERTEX_SHADER.LOW_INT=31300
VERTEX_SHADER.MEDIUM_INT=31300
VERTEX_SHADER.HIGH_INT=31300
]
extensions: [
ANGLE_instanced_arrays
EXT_blend_minmax
EXT_clip_control
EXT_color_buffer_half_float
EXT_depth_clamp
EXT_disjoint_timer_query
EXT_float_blend
EXT_frag_depth
EXT_polygon_offset_clamp
EXT_shader_texture_lod
EXT_texture_compression_bptc
EXT_texture_compression_rgtc
EXT_texture_filter_anisotropic
EXT_texture_mirror_clamp_to_edge
EXT_sRGB
KHR_parallel_shader_compile
OES_element_index_uint
OES_fbo_render_mipmap
OES_standard_derivatives
OES_texture_float
OES_texture_float_linear
OES_texture_half_float
OES_texture_half_float_linear
OES_vertex_array_object
WEBGL_blend_func_extended
WEBGL_color_buffer_float
WEBGL_compressed_texture_astc
WEBGL_compressed_texture_etc
WEBGL_compressed_texture_etc1
WEBGL_compressed_texture_pvrtc
WEBGL_compressed_texture_s3tc
WEBGL_compressed_texture_s3tc_srgb
WEBGL_debug_renderer_info
WEBGL_debug_shaders
WEBGL_depth_texture
WEBGL_draw_buffers
WEBGL_lose_context
WEBGL_multi_draw
WEBGL_polygon_mode
]
extensionParameters: [
CLIP_DEPTH_MODE_EXT=37725
CLIP_ORIGIN_EXT=37724
COLOR_ATTACHMENT0_WEBGL=36064
COLOR_ATTACHMENT10_WEBGL=36074
COLOR_ATTACHMENT11_WEBGL=36075
COLOR_ATTACHMENT12_WEBGL=36076
COLOR_ATTACHMENT13_WEBGL=36077
COLOR_ATTACHMENT14_WEBGL=36078
COLOR_ATTACHMENT15_WEBGL=36079
COLOR_ATTACHMENT1_WEBGL=36065
COLOR_ATTACHMENT2_WEBGL=36066
COLOR_ATTACHMENT3_WEBGL=36067
COLOR_ATTACHMENT4_WEBGL=36068
COLOR_ATTACHMENT5_WEBGL=36069
COLOR_ATTACHMENT6_WEBGL=36070
COLOR_ATTACHMENT7_WEBGL=36071
COLOR_ATTACHMENT8_WEBGL=36072
COLOR_ATTACHMENT9_WEBGL=36073
COMPLETION_STATUS_KHR=37297
COMPRESSED_R11_EAC=37488
COMPRESSED_RED_GREEN_RGTC2_EXT=36285
COMPRESSED_RED_RGTC1_EXT=36283
COMPRESSED_RG11_EAC=37490
COMPRESSED_RGB8_ETC2=37492
COMPRESSED_RGB8_PUNCHTHROUGH_ALPHA1_ETC2=37494
COMPRESSED_RGBA8_ETC2_EAC=37496
COMPRESSED_RGBA_ASTC_10x10_KHR=37819
COMPRESSED_RGBA_ASTC_10x5_KHR=37816
COMPRESSED_RGBA_ASTC_10x6_KHR=37817
COMPRESSED_RGBA_ASTC_10x8_KHR=37818
COMPRESSED_RGBA_ASTC_12x10_KHR=37820
COMPRESSED_RGBA_ASTC_12x12_KHR=37821
COMPRESSED_RGBA_ASTC_4x4_KHR=37808
COMPRESSED_RGBA_ASTC_5x4_KHR=37809
COMPRESSED_RGBA_ASTC_5x5_KHR=37810
COMPRESSED_RGBA_ASTC_6x5_KHR=37811
COMPRESSED_RGBA_ASTC_6x6_KHR=37812
COMPRESSED_RGBA_ASTC_8x5_KHR=37813
COMPRESSED_RGBA_ASTC_8x6_KHR=37814
COMPRESSED_RGBA_ASTC_8x8_KHR=37815
COMPRESSED_RGBA_BPTC_UNORM_EXT=36492
COMPRESSED_RGBA_PVRTC_2BPPV1_IMG=35843
COMPRESSED_RGBA_PVRTC_4BPPV1_IMG=35842
COMPRESSED_RGBA_S3TC_DXT1_EXT=33777
COMPRESSED_RGBA_S3TC_DXT3_EXT=33778
COMPRESSED_RGBA_S3TC_DXT5_EXT=33779
COMPRESSED_RGB_BPTC_SIGNED_FLOAT_EXT=36494
COMPRESSED_RGB_BPTC_UNSIGNED_FLOAT_EXT=36495
COMPRESSED_RGB_ETC1_WEBGL=36196
COMPRESSED_RGB_PVRTC_2BPPV1_IMG=35841
COMPRESSED_RGB_PVRTC_4BPPV1_IMG=35840
COMPRESSED_RGB_S3TC_DXT1_EXT=33776
COMPRESSED_SIGNED_R11_EAC=37489

COMPRESSED_SIGNED_RED_GREEN_RGTC2_EXT=36286
COMPRESSED_SIGNED_RED_RGTC1_EXT=36284
COMPRESSED_SIGNED_RG11_EAC=37491
COMPRESSED_SRGB8_ALPHA8_ASTC_10x10_KHR=37851
COMPRESSED_SRGB8_ALPHA8_ASTC_10x5_KHR=37848
COMPRESSED_SRGB8_ALPHA8_ASTC_10x6_KHR=37849
COMPRESSED_SRGB8_ALPHA8_ASTC_10x8_KHR=37850
COMPRESSED_SRGB8_ALPHA8_ASTC_12x10_KHR=37852
COMPRESSED_SRGB8_ALPHA8_ASTC_12x12_KHR=37853
COMPRESSED_SRGB8_ALPHA8_ASTC_4x4_KHR=37840
COMPRESSED_SRGB8_ALPHA8_ASTC_5x4_KHR=37841
COMPRESSED_SRGB8_ALPHA8_ASTC_5x5_KHR=37842
COMPRESSED_SRGB8_ALPHA8_ASTC_6x5_KHR=37843
COMPRESSED_SRGB8_ALPHA8_ASTC_6x6_KHR=37844
COMPRESSED_SRGB8_ALPHA8_ASTC_8x5_KHR=37845
COMPRESSED_SRGB8_ALPHA8_ASTC_8x6_KHR=37846
COMPRESSED_SRGB8_ALPHA8_ASTC_8x8_KHR=37847
COMPRESSED_SRGB8_ALPHA8_ETC2_EAC=37497
COMPRESSED_SRGB8_ETC2=37493
COMPRESSED_SRGB8_PUNCHTHROUGH_ALPHA1_ETC2=37495
COMPRESSED_SRGB_ALPHA_BPTC_UNORM_EXT=36493
COMPRESSED_SRGB_ALPHA_S3TC_DXT1_EXT=35917
COMPRESSED_SRGB_ALPHA_S3TC_DXT3_EXT=35918
COMPRESSED_SRGB_ALPHA_S3TC_DXT5_EXT=35919
COMPRESSED_SRGB_S3TC_DXT1_EXT=35916
CURRENT_QUERY_EXT=34917
DEPTH_CLAMP_EXT=34383
DRAW_BUFFER0_WEBGL=34853=1029
DRAW_BUFFER10_WEBGL=34863
DRAW_BUFFER11_WEBGL=34864
DRAW_BUFFER12_WEBGL=34865
DRAW_BUFFER13_WEBGL=34866
DRAW_BUFFER14_WEBGL=34867
DRAW_BUFFER15_WEBGL=34868
DRAW_BUFFER1_WEBGL=34854=1029
DRAW_BUFFER2_WEBGL=34855
DRAW_BUFFER3_WEBGL=34856
DRAW_BUFFER4_WEBGL=34857
DRAW_BUFFER5_WEBGL=34858
DRAW_BUFFER6_WEBGL=34859
DRAW_BUFFER7_WEBGL=34860
DRAW_BUFFER8_WEBGL=34861
DRAW_BUFFER9_WEBGL=34862
FRAGMENT_SHADER_DERIVATIVE_HINT_OES=35723=4352
FRAMEBUFFER_ATTACHMENT_COLOR_ENCODING_EXT=33296
FRAMEBUFFER_ATTACHMENT_COMPONENT_TYPE_EXT=33297
FRAMEBUFFER_ATTACHMENT_COMPONENT_TYPE_EXT=33297
GPU_DISJOINT_EXT=36795=false
HALF_FLOAT_OES=36193
LOWER_LEFT_EXT=36001
MAX_COLOR_ATTACHMENTS_WEBGL=36063=8
MAX_DRAW_BUFFERS_WEBGL=34852=8
MAX_DUAL_SOURCE_DRAW_BUFFERS_WEBGL=35068
MAX_EXT=32776
MAX_TEXTURE_MAX_ANISOTROPY_EXT=34047=16
MIN_EXT=32775
MIRROR_CLAMP_TO_EDGE_EXT=34627
NEGATIVE_ONE_TO_ONE_EXT=37726
ONE_MINUS_SRC1_ALPHA_WEBGL=35067
ONE_MINUS_SRC1_COLOR_WEBGL=35066
POLYGON_OFFSET_CLAMP_EXT=36379
QUERY_COUNTER_BITS_EXT=34916
QUERY_RESULT_AVAILABLE_EXT=34919
QUERY_RESULT_EXT=34918
RGB16F_EXT=34843
RGBA16F_EXT=34842
RGBA32F_EXT=34836
SRC1_ALPHA_WEBGL=34185
SRC1_COLOR_WEBGL=35065
SRGB8_ALPHA8_EXT=35907
SRGB_ALPHA_EXT=35906
SRGB_EXT=35904
TEXTURE_MAX_ANISOTROPY_EXT=34046
TIMESTAMP_EXT=36392=0
TIME_ELAPSED_EXT=35007
UNMASKED_RENDERER_WEBGL=37446
UNMASKED_VENDOR_WEBGL=37445
UNSIGNED_INT_24_8_WEBGL=34042
UNSIGNED_NORMALIZED_EXT=35863
UNSIGNED_NORMALIZED_EXT=35863
UPPER_LEFT_EXT=36002
VERTEX_ARRAY_BINDING_OES=34229=null
VERTEX_ATTRIB_ARRAY_DIVISOR_ANGLE=35070
ZERO_TO_ONE_EXT=37727