THE POWER OF WEBAPPS

A TOUR OF WEB CAPABILITIES

MAXIMILIANO FIRTMAN

MAXIMILIANO FIRTMAN

MOBILE+WEB DEVELOPER

HTML since 1996

JavaScript since 1998

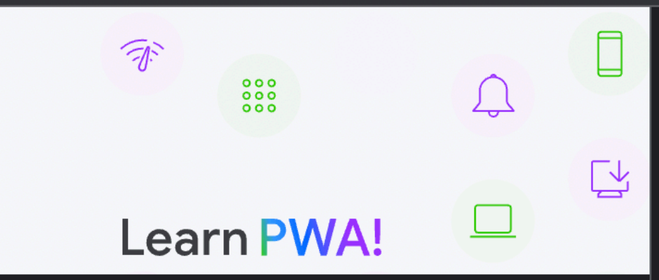
AUTHOR

Authored 13 books and +70 courses

Published +150 webapps

@FIRT · FIRT.DEV

web.dev/learn/pwa



Search

000

Learn PWA

A course that breaks down every aspect of modern progressive web app development.

On this page

Welcome to Learn Progressive Web Apps!

Welcome to Learn Progressive Web Apps!

This course covers the fundamentals of Progressive Web App development into easy-to-understand pieces. Over the following modules, you'll learn what a Progressive Web App is, how to create one or upgrade your existing web content, and how to add all the pieces for an offline, installable app. Use the menu pane by the "Learn PWA" logo to navigate the modules.

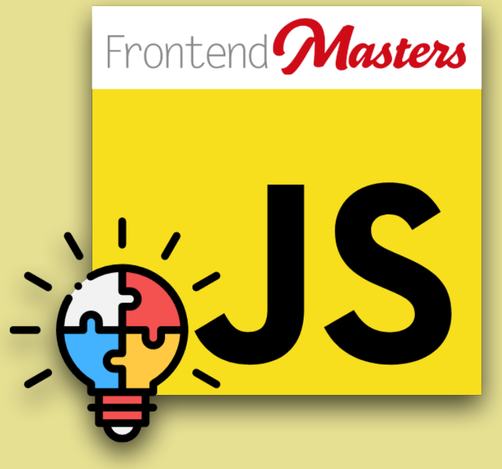
What we'll cover

- Web Capabilities
- Permissions and Security
- Compatibility
- Sensors, Location & Input
- Speech, Voice and Camera
- Hardware & Devices
- Integration with OS
- PWA Capabilities

Pre-requisites

firtman.github.io/capabilities

Questions?



Capabilities

IMPORTANT

We know what a web app is, but we need to define, for today's tour, what's a capability

DEFINITION

Web Capability

Ability of a web browser or web application to perform a specific task or function using typically a built-in API.

WARNING

Have in mind it's the web platform. Compatibility may be an issue with capabilities.

👉**Progressive Enhancement**

Capabilities by maturity

✅ **Green**: APIs available on every browser on every platform, when technically possible. Mature, use them with confidence

✳️ Light-green: Available only on some browsers. They matured within the supported subgroup of browsers so that you can use the capability confidently on them.

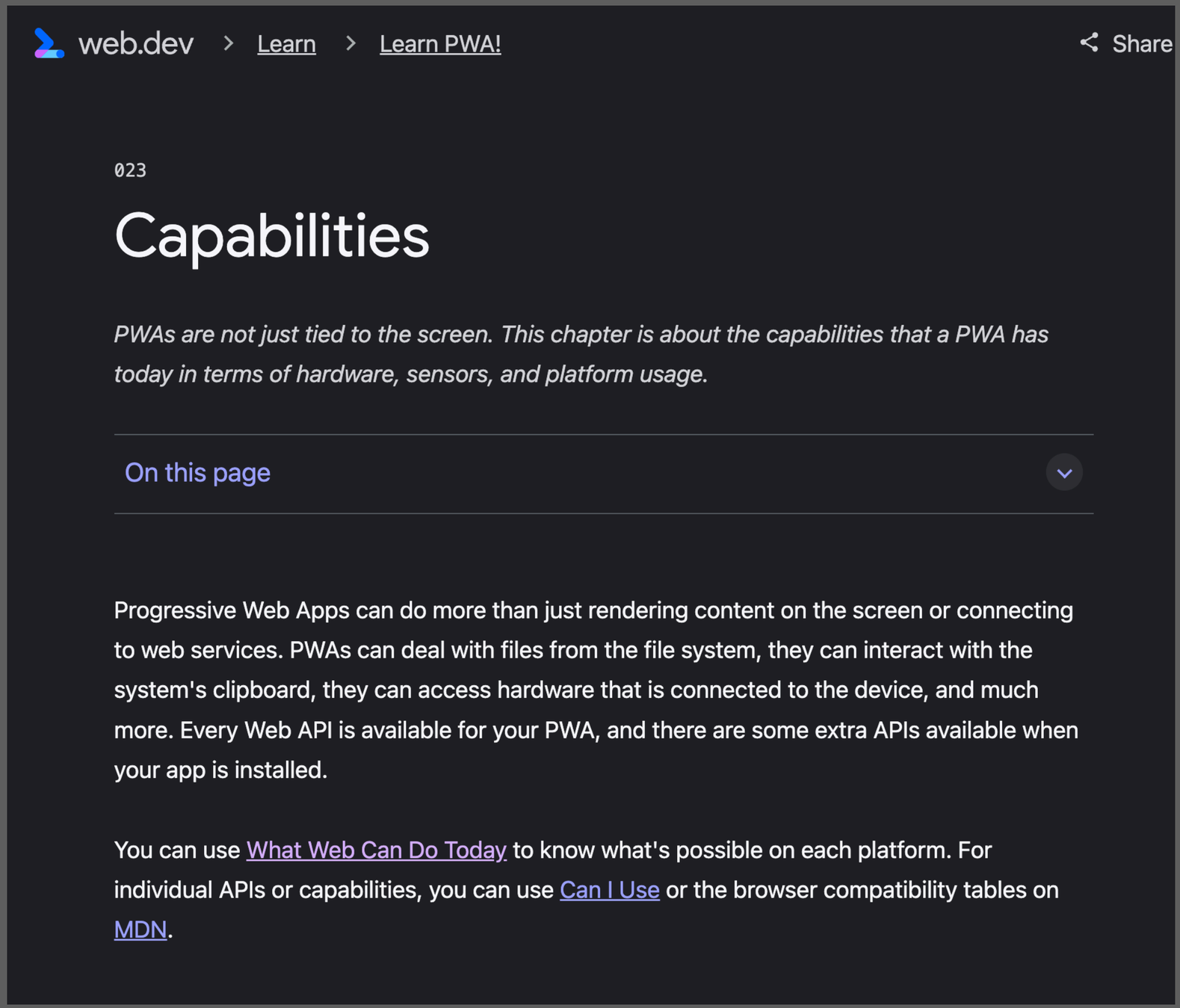
⚠️Yellow: not yet mature; they are only available on some browsers, and within tests or trials.

⛔Red: You can't use them, and plans to add them are still long term.

Meet your new friends

- developer.mozilla.org
- caniuse.com
- web.dev
- webkit.org/blog
- chromestatus.com

web.dev/learn/pwa/capabilities



Basic Capabilities
(that we won't cover today)

Core Capabilities

- Fetch
- Web Workers
- WebAssembly
- WebSocket
- WebRTC
- Web Performance APIs
- Network Information
- Device Memory
- WebOTP
- Web Crypto

Storage Capabilities

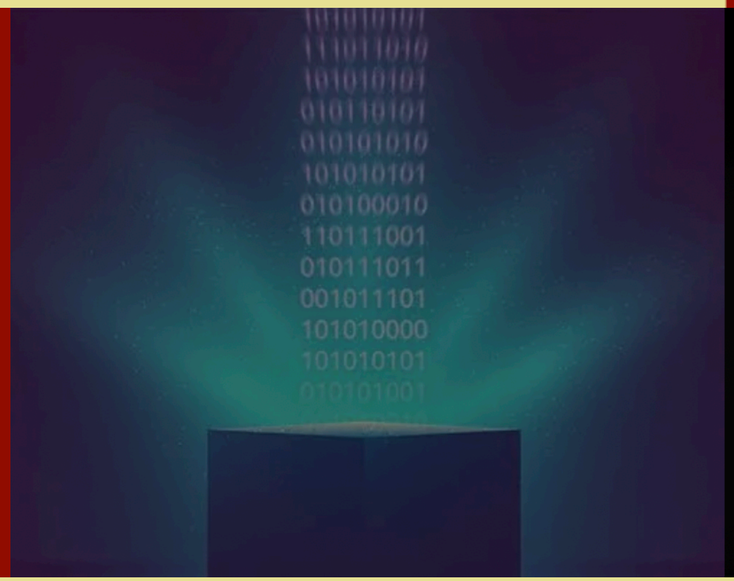
Web Storage

IndexedDB

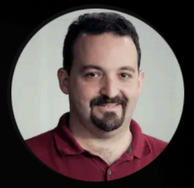
Cache Storage

FileSystem Access

Frontend Masters



Web Storage APIs



Maximiliano Firtman

Independent Consultant

Go beyond localStorage to use newer and more performant APIs like IndexedDB for storing JSON, CacheStorage for caching requests, and the FileSystem API for accessing the local file system.

4 hours, 8 minutes CC

UI Capabilities

- Web Components
- CSS
- 2D Canvas
- WebGL
- Pointer Lock
- Screen Capture

Basic PWA Capabilities

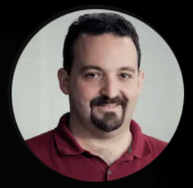
Service Workers

PWA (Web App Manifest)

Icon Installation

Frontend Masters

Build Progressive Web Apps (PWAs) from Scratch



Maximiliano Firtman
Independent Consultant

Build offline-capable Progressive Web Apps with HTML, CSS, and JavaScript: Service workers give you access to the cache storage while App Manifests allow you to be distributed on the Google Play store and Apple App Store.

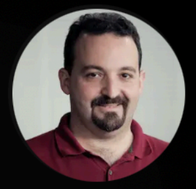
3 hours, 48 minutes CC

Background Capabilities

- Page Visibility
- Background Sync
- Background Fetch
- Web Push and Notifications
- Media Session

Frontend Masters

JavaScript in the Background



Maximiliano Firtman
Independent Consultant

Explore new web app capabilities to detect when your web app is in the background to execute code later, even when your web app isn't in focus.

4 hours, 59 minutes CC

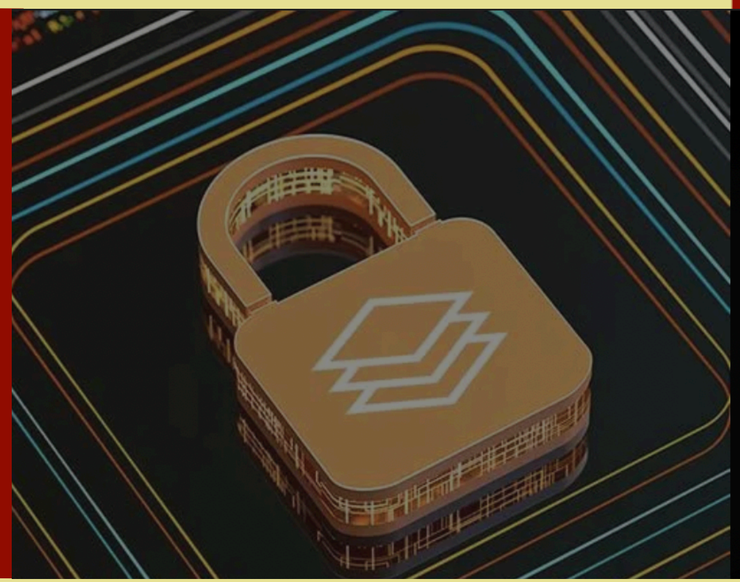
Security Capabilities

Web Authentication

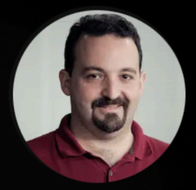
Passkeys

Credential Management

Frontend Masters



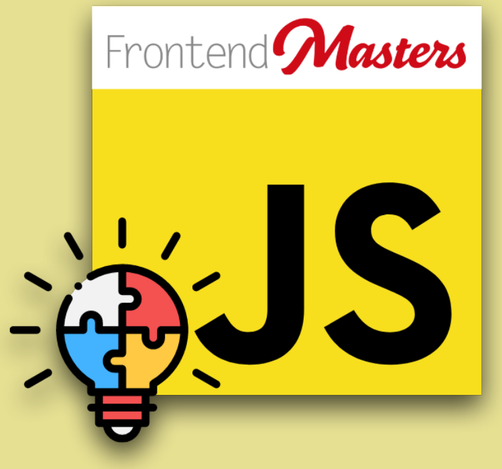
Web Authentication APIs



Maximiliano Firtman
Independent Consultant

Learn basic password logins to using Google to log in. Then, use WebAuthn API and Passkeys to use FaceID and biometrics to make passwordless web apps!

5 hours, 4 minutes CC



Permissions and Security

Permissions and Security

- Some permissions:
  - Are harmless
  - Have no cost
- Some other permissions:
  - Open a privacy risk
  - Involves a cost
- When there are costs or risks, some browsers decide to put a limit on it:
  - User engagement requirements
  - Permission dialog to the user

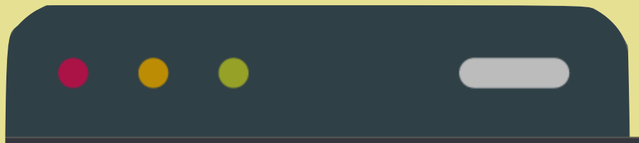
Permissions and Security

- Most capabilities will require HTTPS
- Some capabilities will need a user interaction to be enabled (aka you can't trigger it on page load)
- Permissions are granted on an origin base
- If user denies a permission, the API won't be able to ask again, manual re-enable will be mandatory
- If user granted a permission, it may be with no time limit, for a couple of days, for the session or just for one usage

Permissions and Security

- The API is enabled by default to the main navigation and sometimes off for iframes
- What if you want to turn a capability on or off?
  - Permissions Policy spec
  - It's an HTTP header: `Permissions-Policy`
  - For iframes it's an HTML attribute

Permissions Policy



To enable some permissions you define a name and a list of origins or *, the list can include the special value self

```
Permissions-Policy: geolocation=(self "https://www.mydomain.com"
"https://app.mydomain.com")
```

Permissions Policy

To disable some permissions you define a name and an empty list

```
Permissions-Policy: geolocation=()
```

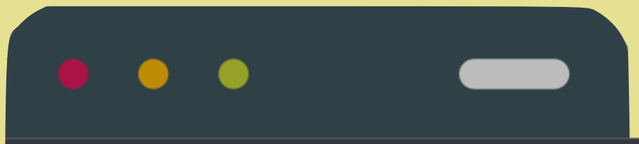
Permissions Policy

You can define multiple policies at once of with multiple HTTP headers

```
Permissions-Policy: geolocation=(self "https://www.mydomain.com"
"https://app.mydomain.com") camera=* picture-in-picture=()

Permissions-Policy: bluetooth=()
```

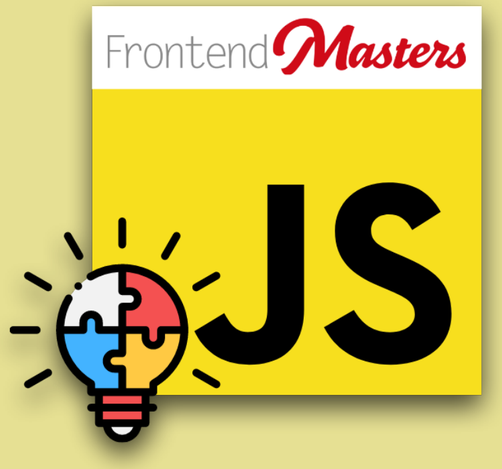
Permissions API

Permissions API



It let us query about current permission state.
It's CHAOS.

```
const response = await navigator.permissions.query(
   {name: "geolocation"}
);

const state = response.state;

// it can be "granted", "prompt", "denied"
// if it throws exception, it's an unsupported permission
```



Sensors, Geolocation and Input Devices

Sensors

Sensors

- There are different sensors on devices, mostly on mobile devices:
  - Accelerometer (3 axis)
  - Gyroscope
  - Magnetometer
  - Proximity
  - Light Sensor
- We have two ways to consume them:
  - Old APIs (global DOM APIs)
  - Sensor API

Sensors API

We use a constructor (different per sensor and ability) and then we query for data changes. ✳️ Not available on iPhone and iPad

```
let magSensor = new Magnetometer({ frequency: 60 });

magSensor.addEventListener("reading", (e) => {
  console.log(`Magnetic field along the X-axis ${magSensor.x}`);
  console.log(`Magnetic field along the Y-axis ${magSensor.y}`);
  console.log(`Magnetic field along the Z-axis ${magSensor.z}`);
});
magSensor.addEventListener("error", (event) => {
  console.log(event.error.name, event.error.message);
});
magSensor.start();
```

Sensors

Sensor	Permission Policy Name
`AbsoluteOrientationSensor`	`'accelerometer'`, `'gyroscope'`, and `'magnetometer'`
`Accelerometer`	`'accelerometer'`
`AmbientLightSensor`	`'ambient-light-sensor'`
`GravitySensor`	`'accelerometer'`
`Gyroscope`	`'gyroscope'`
`LinearAccelerationSensor`	`'accelerometer'`
`Magnetometer`	`'magnetometer'`
`RelativeOrientationSensor`	`'accelerometer'`, and `'gyroscope'`

Classic Sensors API

Still available on all mobile devices; including iOS and iPadOS. It's a set of events on the window object: devicemotion and device orientation

```
addEventListener("devicemotion", (event) ⇒ {
  const x = event.accelerationIncludingGravity.x;
  const y = event.accelerationIncludingGravity.y;
  const z = event.accelerationIncludingGravity.z;
});

addEventListener("deviceorientation", (event) ⇒ {
   const rotateDegrees = event.alpha; // alpha: rotation around z-axis
   const leftToRight = event.gamma; // gamma: left to right
   const frontToBack = event.beta; // beta: front back motion
});
```

Classic Sensors API

On iOS and iPadOS from 13.0 you must request permission with a non-standard API on a user gesture.

```
if ('requestPermission' in DeviceMotionEvent) {
   const response = await DeviceMotionEvent.requestPermission();
   // "granted", "denied"
}
```

Geolocation API

Geolocation API

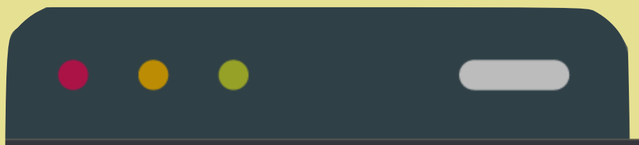
- How does it work?
- Provider-agnostic
- It needs user's permission
- It relies on the browser or operating system service
- Works only in the foreground
- Not suitable for geofencing or beacon-based location

Geolocation API

It's the oldest cool web capability.
That's why it's still an old API

```
const locationAcquiredCB = (location) ⇒ {
    // location has a timestamp and a coords property with
    // latitude, longitude, accuracy, altitude, verticalAccuracy
}
const errorCB = (error) ⇒ {}

// get user's position once
navigator.geolocation.getCurrentPosition(locationAcquiredCB, errorCB);
```

Geolocation API



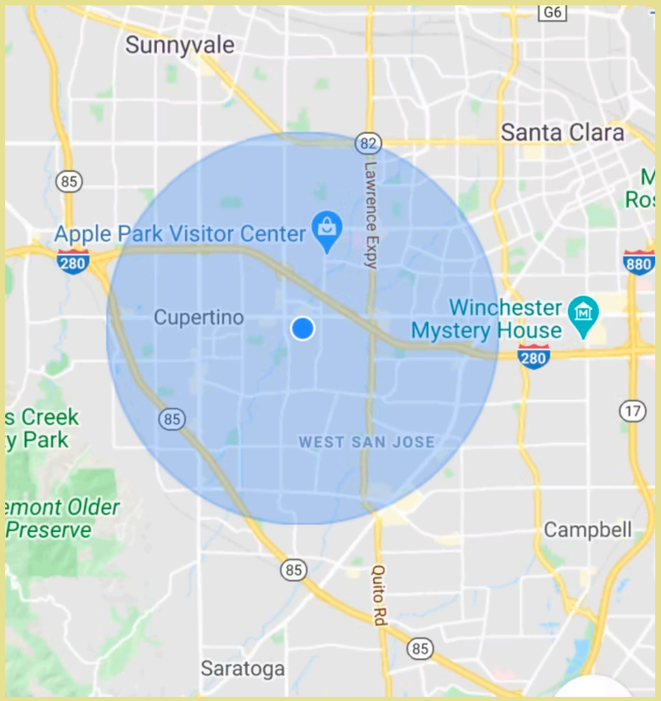
It's the oldest cool web capability.
That's why it's still an old API

```
const locationAcquiredCB = (location) ⇒ {
    // location has a timestamp and a coords property with
    // latitude, longitude, accuracy, altitude, verticalAccuracy
}
const errorCB = (error) ⇒ {}

// listen for changes in user's position
navigator.geolocation.watchPosition(locationAcquiredCB, errorCB);
```

Geolocation API Options

- The third argument has:
  - **maximumAge**
    (in milliseconds, 0 by default)
  - **timeout**
    (Infinity by default)
  - **enableHighAccuracy**
    (false by default)

Accuracy on location



- Since iOS and iPadOS 14, users can decide if they want to share with apps precise or imprecise location
- User grant that location to Safari to all web apps at the same time
- There is no API to actually know in which mode the user is
- For imprecise, you will get an accuracy between 3000 to 9000 meters (~2 to 6 miles)
- Android 12+ has a similar issue but not affecting web apps a lot.

https://firt.dev/ios-14/#geolocation-changes

Screen Orientation

Screen Orientation

It let us know the current orientation (green) and lock it to one specific (light green).

```
screen.addEventListener("orientationchange", () ⇒ {
  console.log(`The orientation of the screen is: ${screen.orientation}`);
});

const currentOrientation = screen.orientation;
screen.orientation.lock();

// it can also accept an argument with: any, natural. portrait-primary,
portrait-secondary, landscape-primary, landscape-secondary, portrait, and
landscape
```

Touch Events

Touch Events

- They work only with touch screens
  - Basic: `touchstart`, `touchend`, `touchcancel`
  - Drag: `touchmove`
- The event is fired with multiple coordinates (for multi-touch) including optional force if available
- iOS also supports a non-standard Gesture event spec

Pointer Events

Pointer Events

- They work with: mouse, trackpad, touch, pen, stylus and more. You receive one call per interaction, even if they are at the same time.
  - Press: `pointerdown`, `pointerup`, `pointercancel`
  - Hover: `pointerover`, `pointerout`
  - Drag: `pointermove`, `pointerenter`, `pointerleave`
- The event receives coordinates, a pointer id, the pointer type and optional pressure, tilt, twist data (useful with a stylus or on some touch screens)

Virtual Keyboard

Virtual Keyboard API

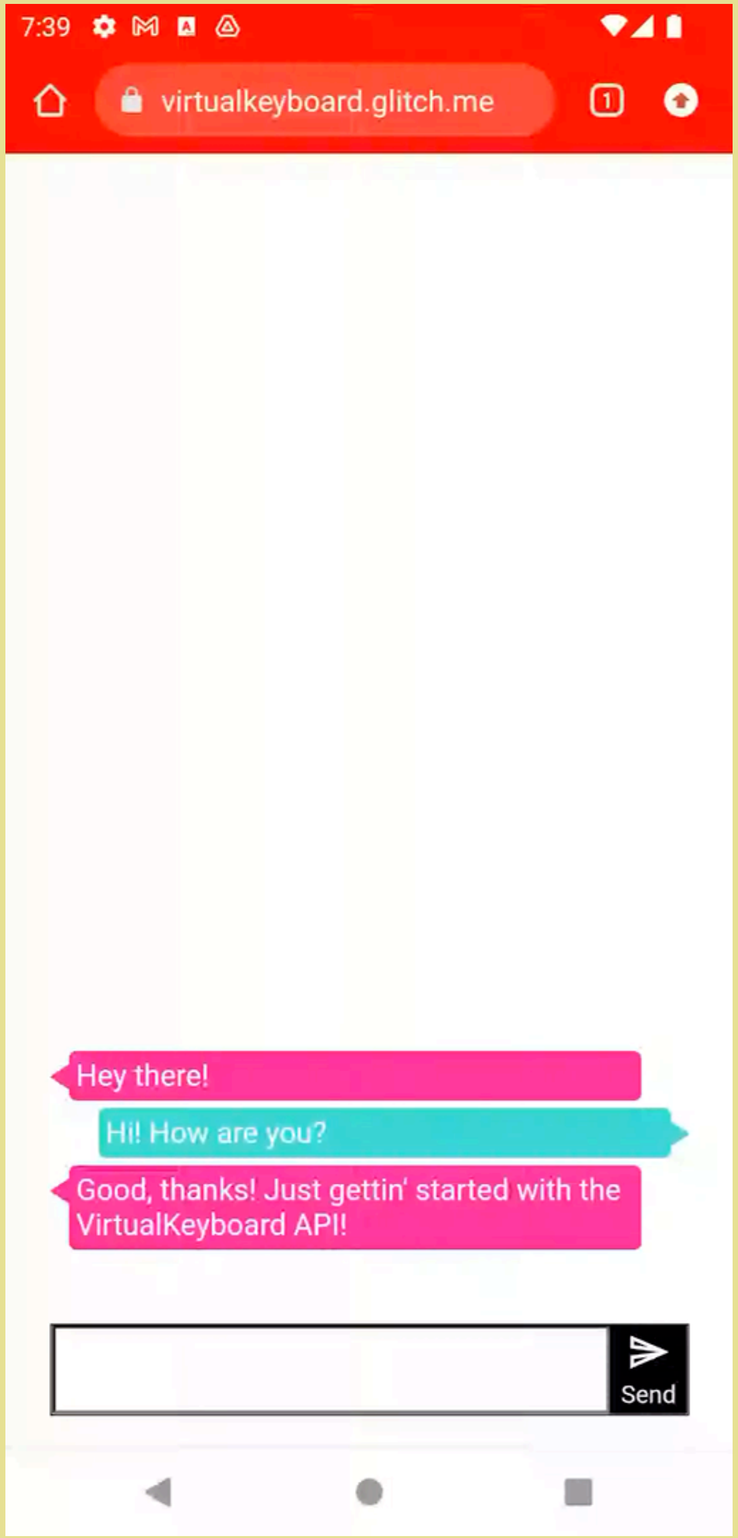
To use with touch devices

```
navigator.virtualKeyboard.overlaysContent = true;
navigator.virtualKeyboard.show();
navigator.virtualKeyboard.hide();
navigator.virtualKeyboard.addEventListener('geometrychanged',
                                  e ⇒ {})
```

Virtual Keyboard API for CSS

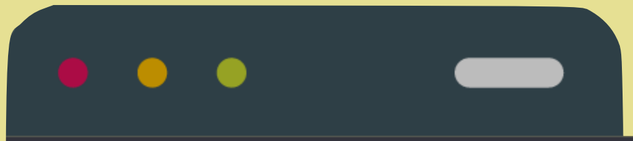
There are also new environmental variables

```
/*
  keyboard-inset-top
   keyboard-inset-right
   keyboard-inset-bottom
   keyboard-inset-left
   keyboard-inset-width
   keyboard-inset-height
  */

  margin-block-end: env(keyboard-inset-height, 100px);
```


7:39
virtualkeyboard.glitch.me
Hey there!
Hi! How are you?
Good, thanks! Just gettin' started with the VirtualKeyboard API!
Send

Gamepad API

Gamepad API



You need to first connect your Gamepad to the device (USB or Bluetooth) and query it in a requestAnimationFrame loop

```
window.addEventListener('gamepadconnected', event ⇒ {
   const buttons = e.gamepad.buttons;
});
```

Web HID

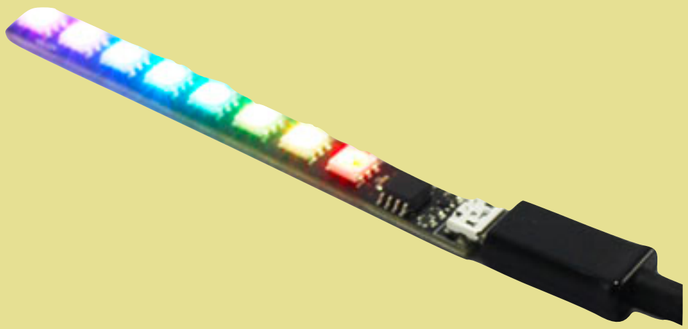
Web HID

(Human Interface Device)

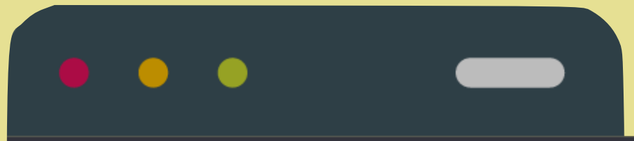
- Let us access devices connected to a user's computer via USB or Bluetooth.
- Built on top of the widely adopted HID protocol.
- Provides users with control over which devices web applications can access.
- Users must grant permission through a browser dialog box before a web application can access a device.
- Useful for non-standard input devices
- **It's a low-level API**


STREAM DECK


X
Y
A
B



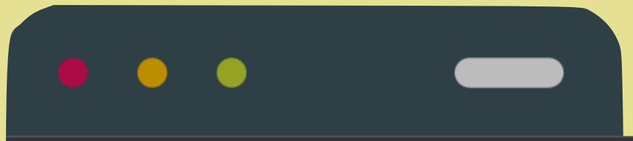
WebHID API



You need to know the device communication API

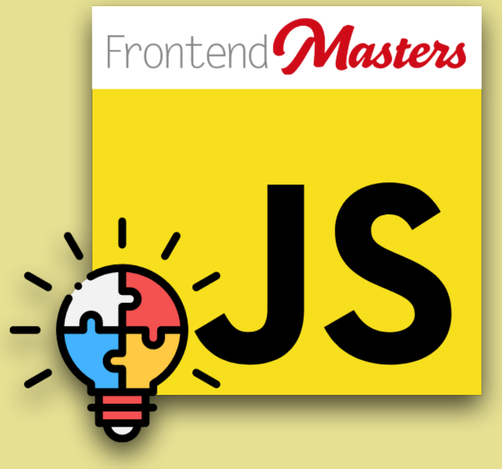
```
let devices = await navigator.hid.getDevices();

devices.forEach((device) => {
  console.log(`HID: ${device}`);
});
```

Gamepad API



You need to first connect your Gamepad to the device (USB or Bluetooth) and query it in a requestAnimationFrame loop

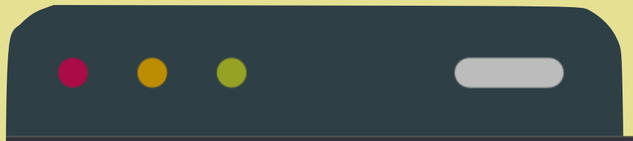
```
window.addEventListener('gamepadconnected', event ⇒ {
   const buttons = e.gamepad.buttons;
});
```



Speech, Voice and Camera

Speech Recognition

Speech Recognition API



It allows us to listen for user's voice and get a text transcript.
It works only after a user gesture and microphone permission

```
const recognition = new SpeechRecognition();
recognition.continuous = false;
recognition.lang = "fr-CA";

recognition.addEventListener("result", event ⇒ {
    if (event.type=="result") {
       event.results.forEach( result ⇒ {
              const confidence = result.confidence*100;
              const text = result.transcript;
       });
    }
})
```

11:03

Web Speech API

Speak now.

English

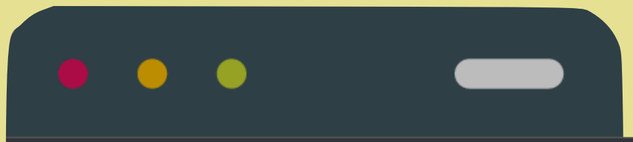
United Kingdom

AA

google.co

Speech Synthesis

Speech Synthesis API



We can make the web app speak in the standard speaker based on a text transcript. It uses synthesis voices from the OS.

```
var utterance = new SpeechSynthesisUtterance();
utterance.text = "¡Hola! Bienvenidos a mi curso";
utterance.lang = "es-AR";
utterance.rate = 1;
utterance.volume = 1;
utterance.pitch = 1;

speechSynthesis.speak(utterance);
```

Web Speech Synthesis Demo

Your browser **supports** speech synthesis.

Hello! This is a great conference

Voice

Volume

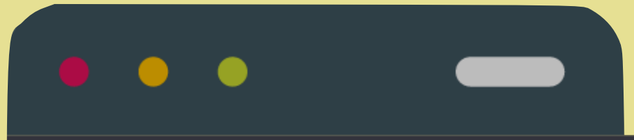
Rate

Pitch

Speak

Shape Detection

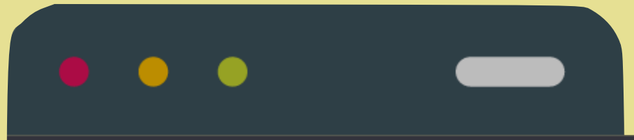
Barcode Detector



This is an experimental API working on Chrome for Android only

```
const detector = new BarcodeDetector();
const objects = await detector.detect(image);
objects.forEach(object ⇒ console.log(object));
```

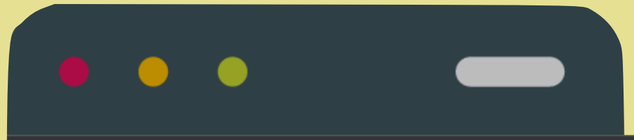
Face Detector



This is an experimental API working on Chrome for Android only

```
const detector = new FaceDetector();
const objects = await detector.detect(image);
objects.forEach(object ⇒ console.log(object));
```

Text Detector (OCR)



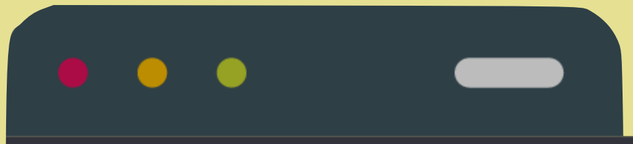
This is an experimental API working on Chrome for Android only

```
const detector = new TextDetector();
const objects = await detector.detect(image);
objects.forEach(object ⇒ console.log(object));
```

Read Text

Media Devices

Media Devices



It lets you open the camera and microphone to get the stream.
For years, it was known as getUserMedia

```
const stream = await navigator.mediaDevices
  .getUserMedia({
      audio: true,
      video: true,
  });
const video = document.querySelector("video");
video.srcObject = stream;
video.onloadedmetadata = () => {
  video.play();
};
```

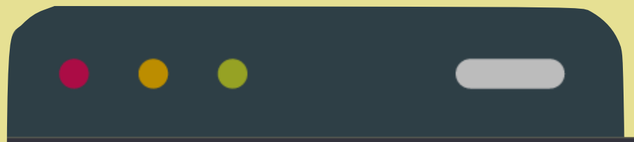
Advanced Control Camera

Augmented Reality

Augmented Reality (AR, XR)

- We can use WebXR to use VR/AR devices and then render content in WebGL/WebGPU
- We can use a high-level solution for augmented-reality objects on the screen
- For WebXR, different APIs are available for
  - Devices available
  - 3D Space detection
  - Pose
  - Depth sensing
  - Create vectors representing the movements of input controls
  - Lightning estimation
  - Hit testing

WebXR



It works with AR and VR, using devices or just a magic window.

```
await navigator.xr.requestSession('immersive-vr');

await navigator.xr.requestSession('immersive-ar');

await navigator.xr.requestSession('inline');
```

AR Quick Look

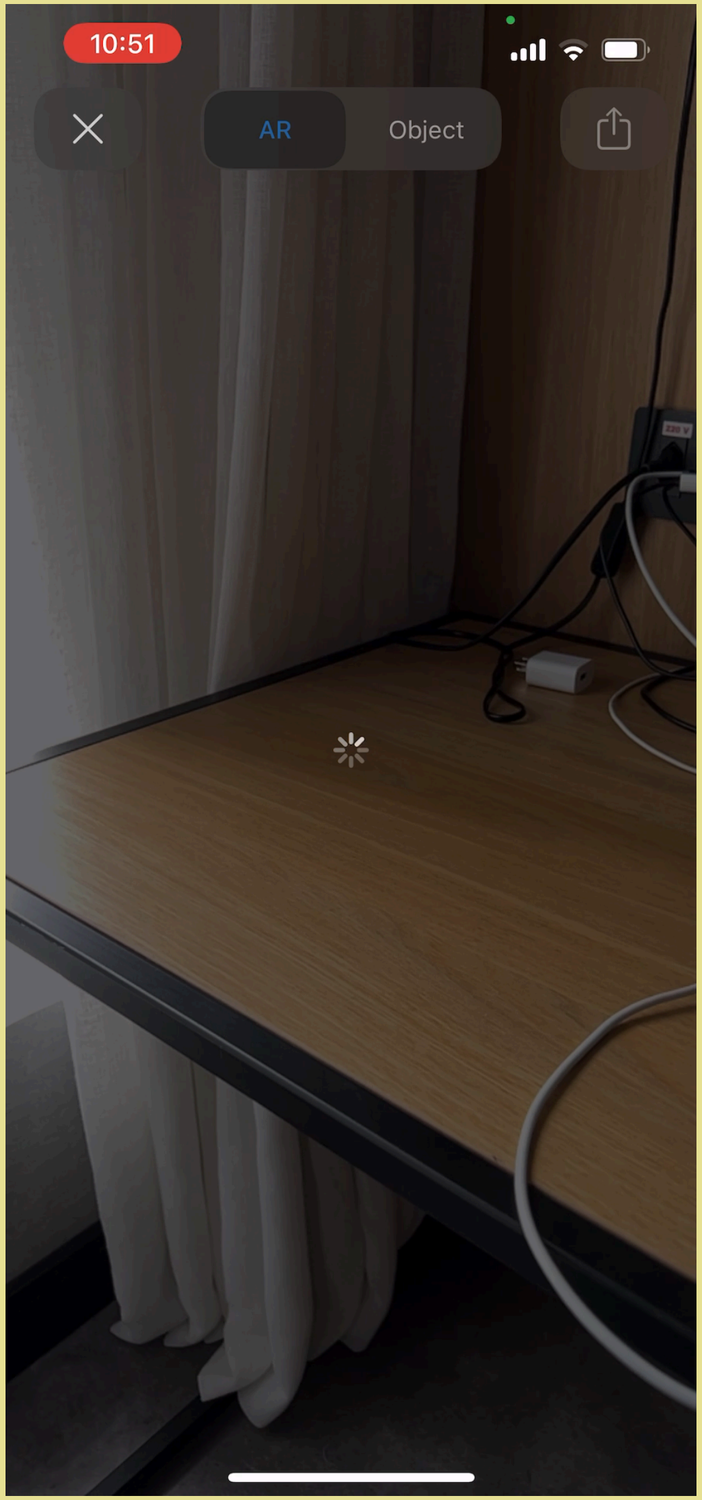
It works only on iOS and iPadOS

```
<a rel="ar" href="model.usdz">
   <img src="model-thumbnail.jpg">
</a>
```

Model Viewer

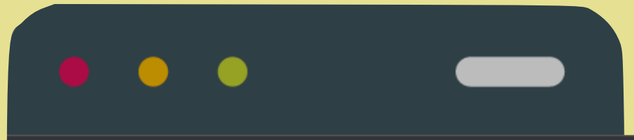
Check modelviewer.dev

```
<script type="module" src="https://
ajax.googleapis.com/ajax/libs/model-viewer/3.1.1/
model-viewer.min.js"></script>

<model-viewer alt="" src="model.glb" ar environment-
image="environment.hdr" poster="poster.jpg" shadow-
intensity="1" camera-controls touch-action="pan-y">
</model-viewer>
```


10:51
AR
Object

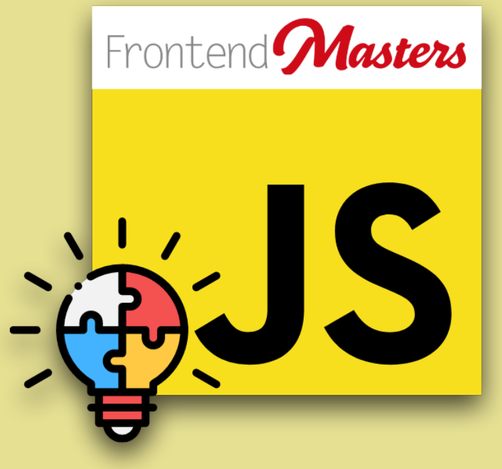
Screen Wake Lock API

Screen Wake Lock



It can make our screen visible all the time, disabling lock from OS

```
wakeLock = await navigator.wakeLock.request();

wakeLock.addEventListener('release', () => {
  // it was released
});

wakeLock.release();
```



External Hardware and Devices

Web Bluetooth

Web Bluetooth

With this API we can:

1- Scan BLE devices

2- Scan services available

3- Connect to these services

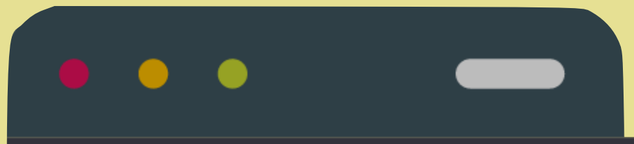
4- Send and receive data

It's a low-level API

We need to manipulate binary data

We need to understand the device's communication protocol.

Bluetooth API



We can connect to devices that we never connected before on that participar device

```
const device = await navigator.bluetooth.requestDevice({
   filters: [{ services: ['heart_rate'] }]
});

const server = await device.gatt.connect();
const service = await server.getPrimaryService('heart_rate');
const characteristic = await service.
                        getCharacteristic('heart_rate_control_measurement');

await characteristic.startNotifications();
characteristic.addEventListener('characteristicvaluechanged', event ⇒ {
  const value = event.target.value;
});
```

Web Audio

Web Audio

We can use the Web Audio API for

1- Generate dynamic audio

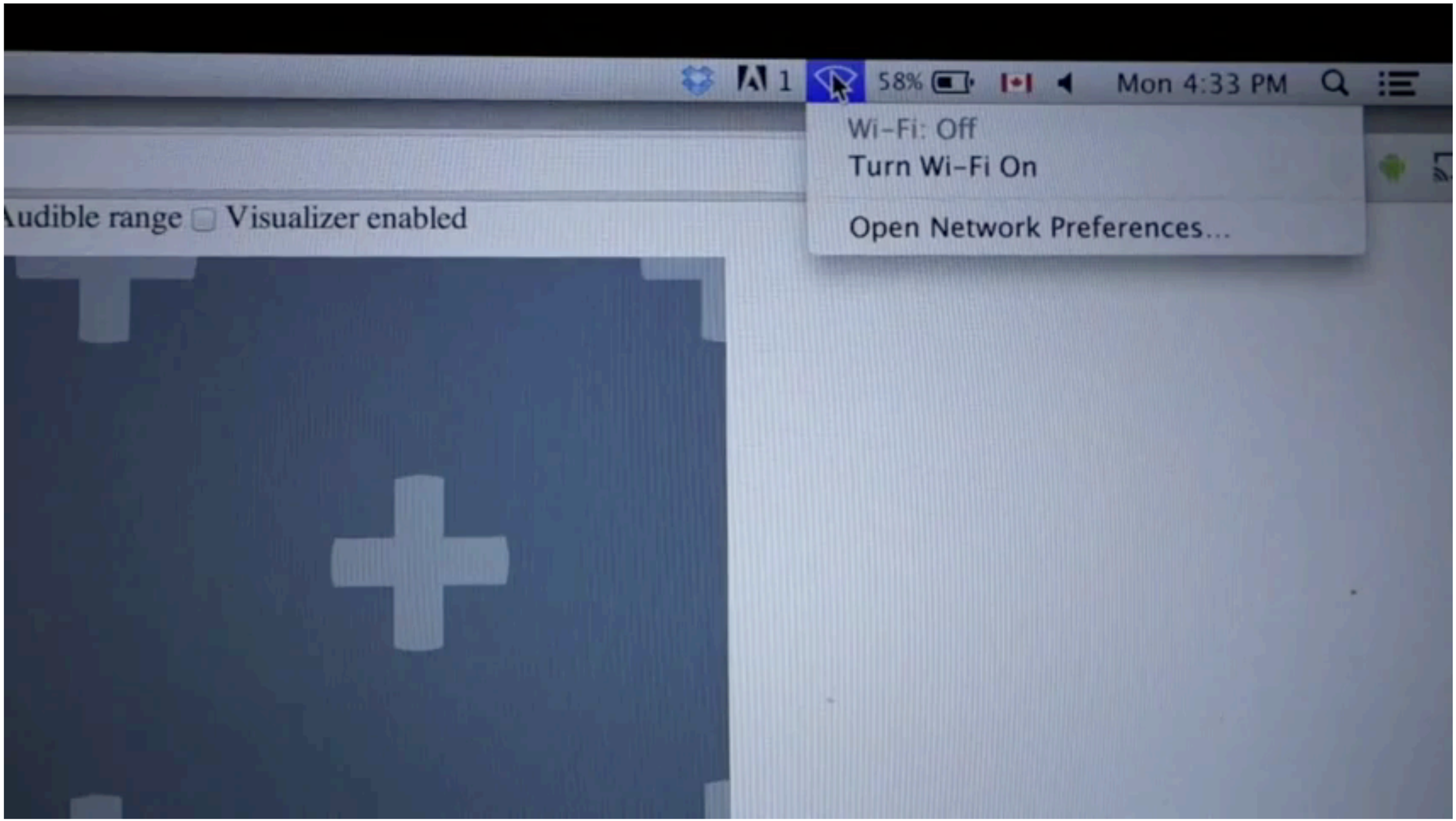
2- 3D Audio

3- Useful for ultrasound communication with devices!

Using Web Audio for Ultrasound networking

We are using here the Sonicnet library

```
// Sending
ssocket = new SonicSocket({alphabet: '0123456789'});
ssocket.send('31415');

// Receiving
sserver = new SonicServer({alphabet: '0123456789'});
sserver.on('message', function(message) {
  // Expect message to be '31415'.
  console.log(message);
});
sserver.start()
```


1
58%
Mon 4:33 PM
Wi-Fi: Off
Turn Wi-Fi On
Open Network Preferences...
udible range
Visualizer enabled

Web MIDI

Web MIDI

- It's a low-level API
- Connect to music devices, such as synthesizers, keyboards, guitars, drum machines and also lightning systems
- Send and receive MIDI messages

Web MIDI

```
navigator.requestMIDIAccess({sysex: false})
         .then(onMIDISuccess, onMIDIFailure);

function onMIDISuccess(midi) {
    Const inputs = midi.inputs.values();
    for (var input=…) {
       input.value.addEventListener(“midimessage”,
             function(event) {
                 // event.data has the bytes
       });
    }
}
```

Web Serial

Web Serial

- It's a low-level API
- Communicate with serial devices connected to a user's computer via USB, Bluetooth, or other serial connections.
- Built on top of the existing serial protocol, which is widely supported by devices and operating systems.
- It requires user permission before allowing web applications to access serial devices.
- Sending and receiving data, controlling settings, and detecting device disconnection.

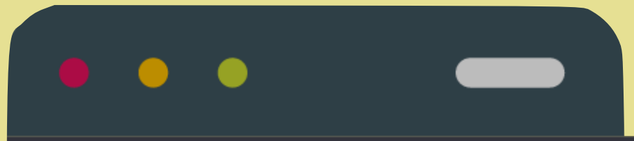
Web USB

Web USB

- It's a low-level API; targeted to device vendors mainly
- The device should not be registered by the OS yet with a driver
- Communication with USB devices connected to a user's computer.
- Read and write data directly to and from USB devices, without the need for custom drivers or proprietary software.
- It requires user permission before allowing web applications to access USB devices.
- Configuring device settings, updating firmware, and accessing device data.

Vibration

Vibration API



It doesn't work on many browsers :(

```
window.navigator.vibrate(300);

window.navigator.vibrate([200, 100, 200]);
```

Battery Status

Battery Status API

It doesn't work on many browsers :(

```
const battery = await navigator.getBattery();

console.log(battery.level*100);


battery.addEventListener("chargingchange", () ⇒ {}
battery.addEventListener("levelchange", () ⇒ {}
battery.addEventListener("chargingtimechange", () ⇒ {}
battery.addEventListener("dischargingtimechange", () ⇒ {}
```

Idle Detection

Idle Detection

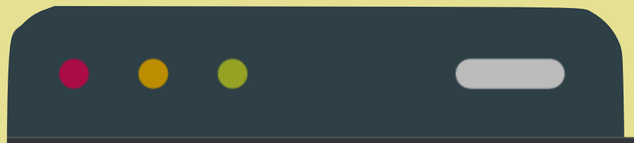
```
if (await idleDetector.requestPermission()=="granted") {
    const idleDetector = new IdleDetector();

    idleDetector.addEventListener("change", () ⇒ {
      const userState = idleDetector.userState;
      const screenState = idleDetector.screenState;
    });

    await idleDetector.start();
}
```

Web NFC

Web NFC

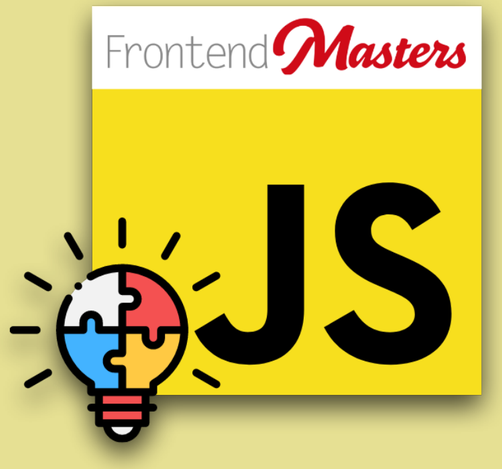
- Near Field Communication
- Access to Tap to Pay, Tap to Share, etc.
- We can access tags within (5-10cm, 2-4 inches)
- The current scope is limited to NDEF (NFC Data Exchange Format): a lightweight binary message format.
  - We can read tags
  - We can write to tags on compatible browsers and tags
- Low-level I/O operations and Host-based Card Emulation (HCE) are not supported.

Web NFC



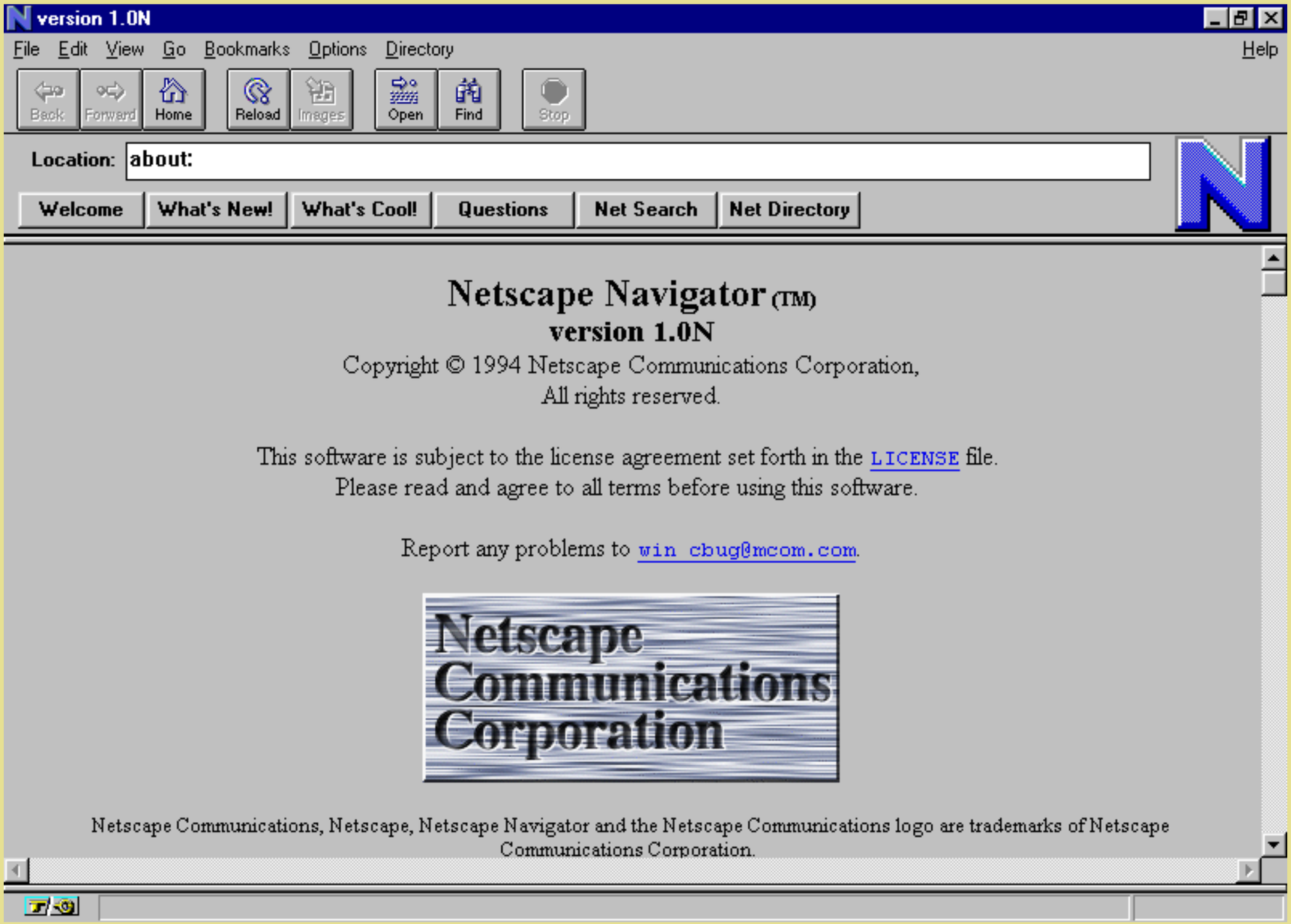
We will need user's permission and then we will start scanning for NFC tags.

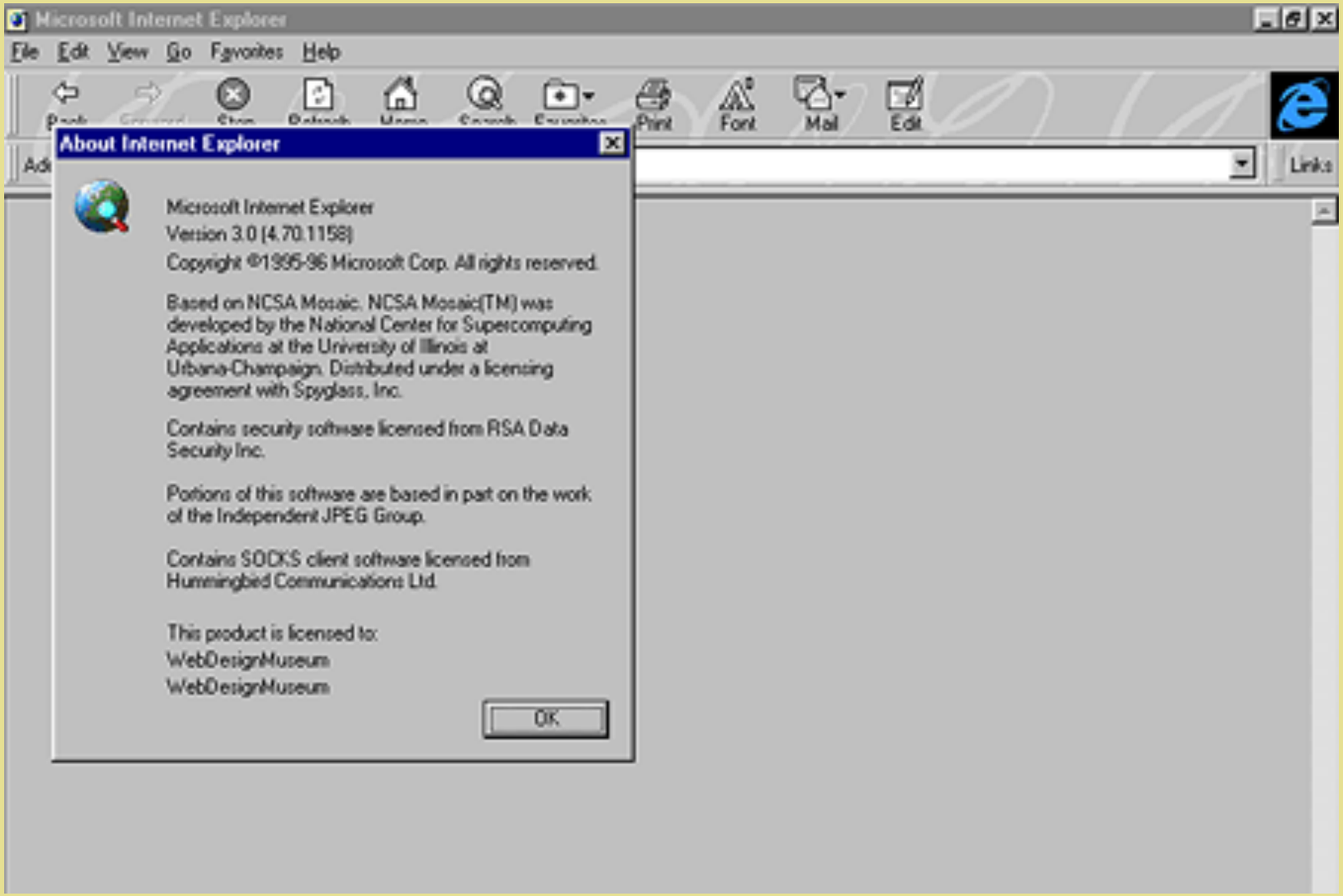
```
const ndef = new NDEFReader();
await ndef.scan();

ndef.addEventListener("readingerror", () ⇒ {
   log("Error reading this tag");
});

ndef.addEventListener("reading", ({ message, serialNumber }) ⇒ {
   log(`Serial Number: ${serialNumber}`);
   log(`Records: (${message.records.length})`);
});
```



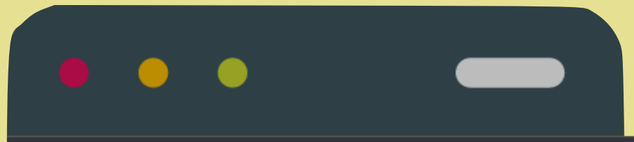
OS Integration

Managing Windows


version 1.0N
File Edit View Go Bookmarks Options Directory Help
Back Forward Home Reload Images Open Find Stop
Location: about:
Welcome What's New! What's Cool! Questions Net Search Net Directory
Netscape Navigator (TM)
version 1.0N
Copyright © 1994 Netscape Communications Corporation,
All rights reserved.
This software is subject to the license agreement set forth in the LICENSE file.
Please read and agree to all terms before using this software.
Report any problems to win_cbug@mcom.com.
Netscape Communications Corporation
Netscape Communications, Netscape, Netscape Navigator and the Netscape Communications logo are trademarks of Netscape Communications Corporation.


Microsoft Internet Explorer
File Edit View Go Favorites Help
Print
Font
Mail
Edit
Links
About Internet Explorer
Microsoft Internet Explorer
Version 3.0 (4.70.1158)
Copyright ©1995-96 Microsoft Corp. All rights reserved.
Based on NCSA Mosaic. NCSA Mosaic(TM) was developed by the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign. Distributed under a licensing agreement with Spyglass, Inc.
Contains security software licensed from RSA Data Security Inc.
Portions of this software are based in part on the work of the Independent JPEG Group.
Contains SOCKS client software licensed from Hummingbird Communications Ltd.
This product is licensed to:
WebDesignMuseum
WebDesignMuseum
OK

Classic APIs from the 90's



We can move windows again!

```
window.moveTo(16, 16);
window.resizeTo(800, 600);
window.open("", "_blank", "");
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help

Sat Apr 30 11:20

PWA WindowsDemo

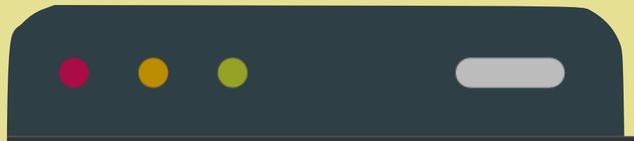
mlearn-pwa-windows-basic.glitch.me

Desktop - Maximili...Book Pro

PWA Windows Demo

Install this PWA in your device and open it from the icon to test all the features

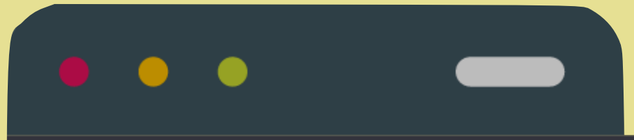
Multi-Screen Windows Placement API



We can move windows again!

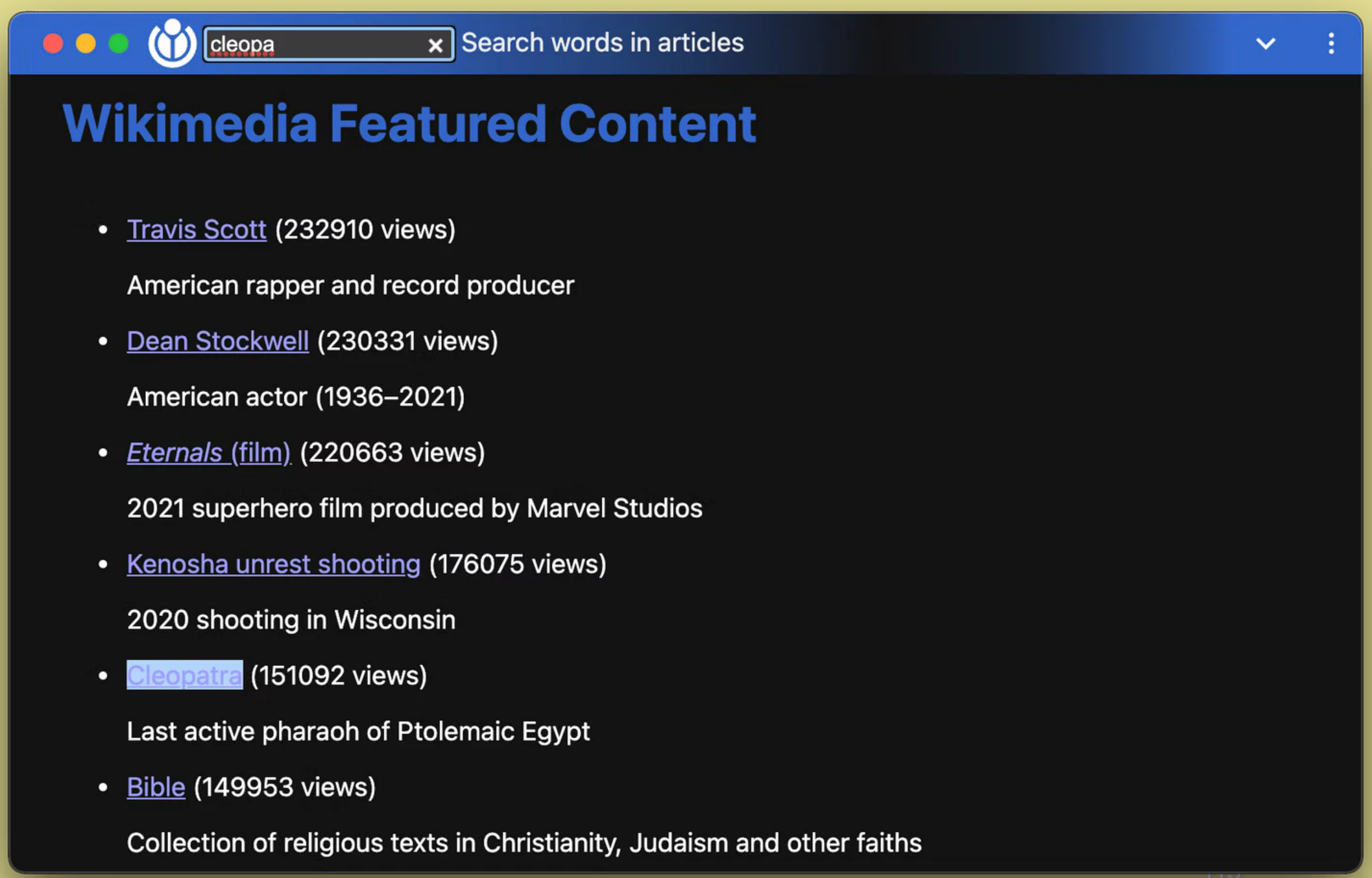
```
const screenDetails = await window.getScreenDetails();


screenDetails.addEventListener('screenschange', e()⇒{

}
```

Windows Controls Overlay



In the Web App Manifest we can express a new display mode

```
"display_override": ["window-controls-overlay"]
```


cleopa
Search words in articles
Wikimedia Featured Content
Travis Scott (232910 views)
American rapper and record producer
Dean Stockwell (230331 views)
American actor (1936–2021)
Eternals (film) (220663 views)
2021 superhero film produced by Marvel Studios
Kenosha unrest shooting (176075 views)
2020 shooting in Wisconsin
Cleopatra (151092 views)
Last active pharaoh of Ptolemaic Egypt
Bible (149953 views)
Collection of religious texts in Christianity, Judaism and other faiths

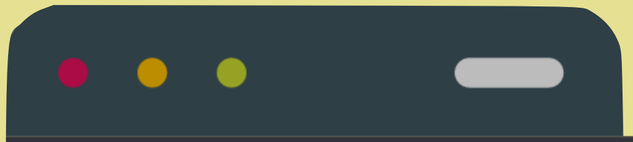
File Handler

File Handler

In the Web App Manifest we can express our intentions

```
"file_handlers": [
    {
      "action": "/readcsv",
      "accept": {
        "text/csv": [".csv"]
      },
      "icons": [
        // ...
      ],
      "launch_type": "single-client"
    }
]
```

File Handler

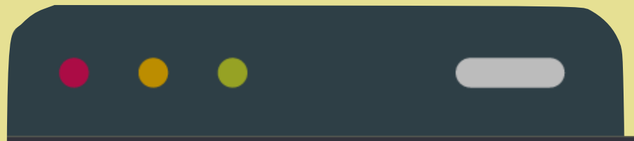


In JavaScript then we can check on the launch queue if there are files to process

```
launchQueue.setConsumer( launchParams ⇒ {
    // Nothing to do when the queue is empty
    if (!launchParams.files.length) return;

    for (const fileHandle of launchParams.files) {
      // Handle each file
    }
}
```

URL Protocol Handler

Protocol Handler



In the Web App Manifest we can express our protocol that must be prefixed with **web+**

```
{
  "protocol_handlers": [
    {
      "protocol": "web+frontendmasters",
      "url": "/watch?argument=%s"
    },
}
```

Protocol Handler

We can now use it in a standard link

```
<a href="web+frontendmasters://swift>Max's Swift Course</a>
```

Web Share

Web Share

```
navigator.share({
    title: 'firt.dev',
    text: 'Content for web devs about PWAs',
    url: 'https://firt.dev',
    // optional files array available
})
```

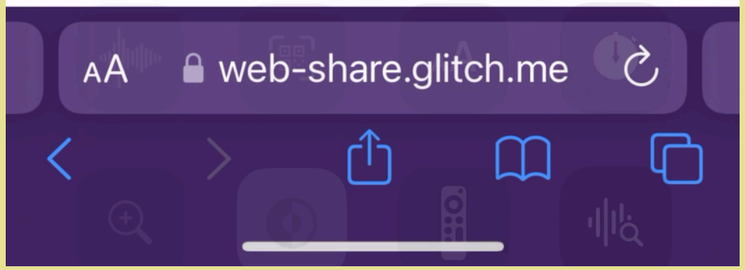
11:17

Web Share API

The Web Share API, allows sites to share links and text to other apps via the system provided share picker.

```
const shareOpts = {
  title: 'Jabberwocky',
  text: 'Check out this great poem about a Jabber
  url: 'https://en.wikipedia.org/wiki/Jabberwocky
};
navigator.share(shareOpts);
```

Share



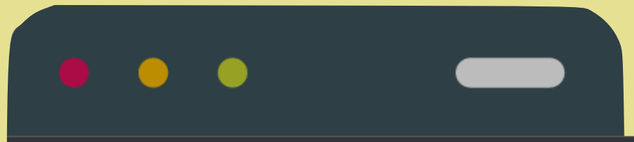
Web Share Target

We can express in the Web App Manifest our intentions

```
"share_target": {
  "action": "/receiving-share/",
  "method": "GET",
  "params": {
    "id": "id",
    "text": "contents",
 }
}
```

Contact Picker API

Contact Picker

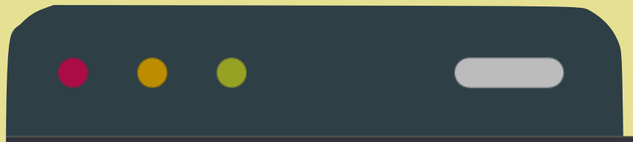


We can read contacts from user's database

```
const props = ["name", "email", "tel", "address", "icon"];
const opts = { multiple: true };

const contacts = await navigator.contacts.select(props, opts);
```

Fullscreen API

Fullscreen API



We can request one DOM element to enter fullscreen.
On iPhone it works only with <video>. Safari uses prefix :(

```
domElement.requestFullscreen();

const currentFS = document.fullscreenElement;
document.exitFullscreen();

document.addEventListener("fullscreenchange", event ⇒ {});
```

Payment Request

Payment Request

The browser is the intermediary

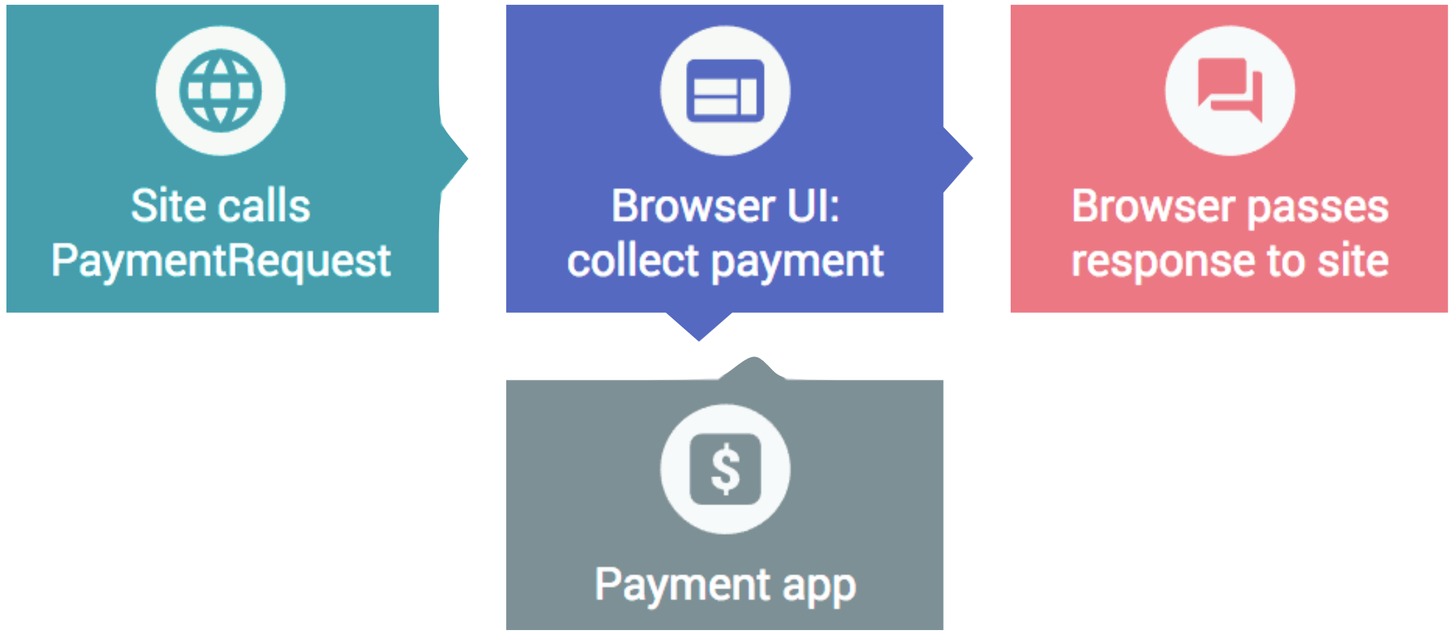
Merchant (website)

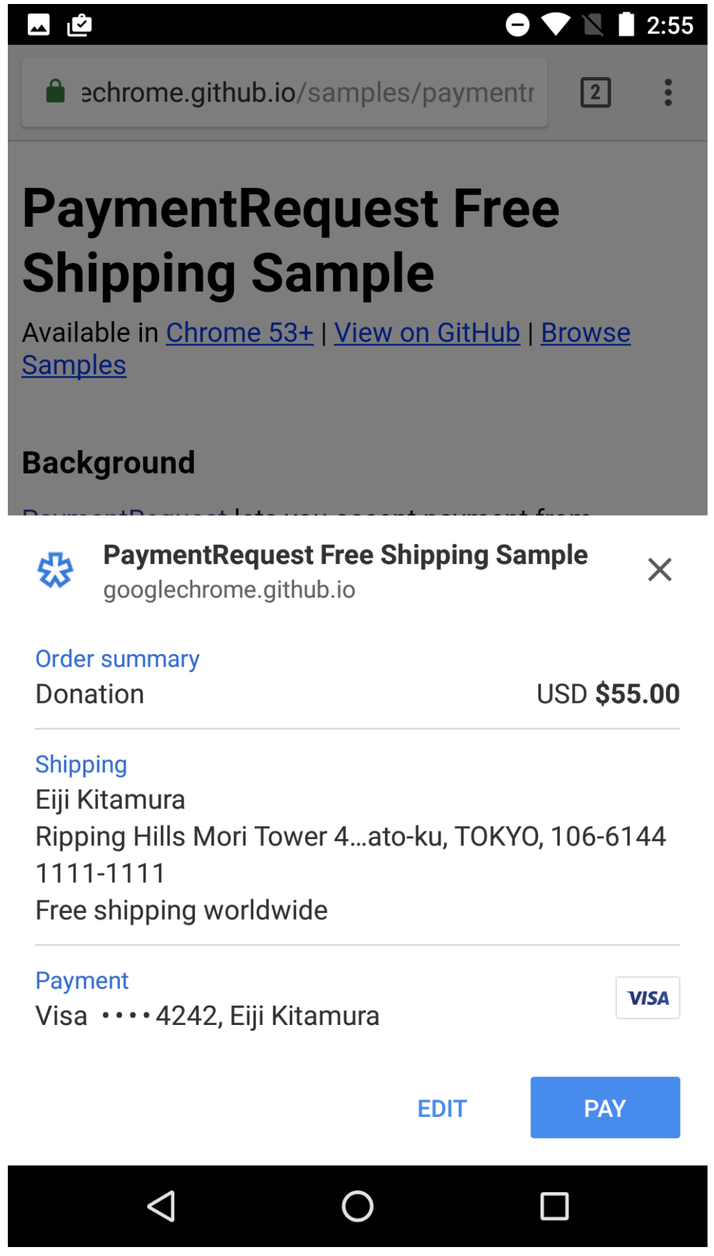
User

Payment Processor

Apple Pay JS is also available

Payment Request





From developers.google.com

Get Installed Related Apps

IMPORTANT

We need to know the native app's package ID, typically in the form of: com.domain.app-name

Get Installed Related Apps

We start with the Web App Manifest

```
"related_applications": [
    {
     "platform": "play",
     "id": "com.myapp.pwa",
     "url": "https://play.google.com/store/apps/details?
id=com.myapp.pwa",
    }
```

Get Installed Related Apps

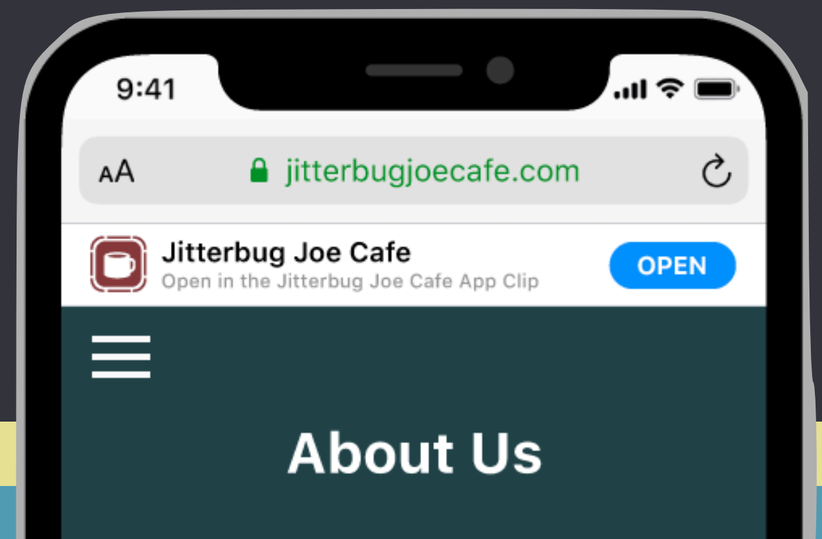
Then we can use the API

```
// we get all the installed related apps taken from Manifest
const installedApps = await navigator.getInstalledRelatedApps();
// we filter them to find the one we need by package ID
const packageId = "com.app.pwa";
const app = installedApps.find(app ⇒ app.id == packageId);
if (app) {
   // app was found
   console.log(`${app.id} version ${app.version} is installed`);
}
```

Smart App Banner

For iOS and iPadOS we only have a meta tag for AppStore apps

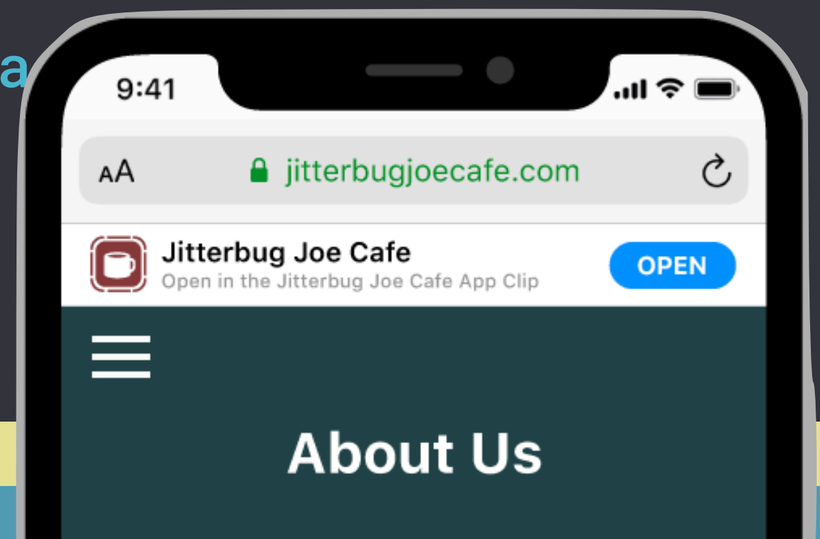
```
<meta name="apple-itunes-app"
      content="app-id=com.myapp.pwa">
```



Smart App Banner

For iOS and iPadOS we only have a meta tag for AppStore apps

```
<meta name="apple-itunes-app"
      content="app-id=com.myapp.pwa">

<meta name="apple-itunes-app"
content="app-id=com.myapp.pwa,app-argument=myapp.com/deep/link">

<meta name="apple-itunes-app"
content="app-clip-bundle-id=com.myapp.pwa
```



App Badging

App Badging

It lets us alert the user that there is new content or pending tasks in our app

Android: Chrome and Edge

Windows: Chrome and Edge

macOS: Chrome and Edge

iOS and iPadOS: Installed PWAs (only available after a Web Push notification was received)

App Badging

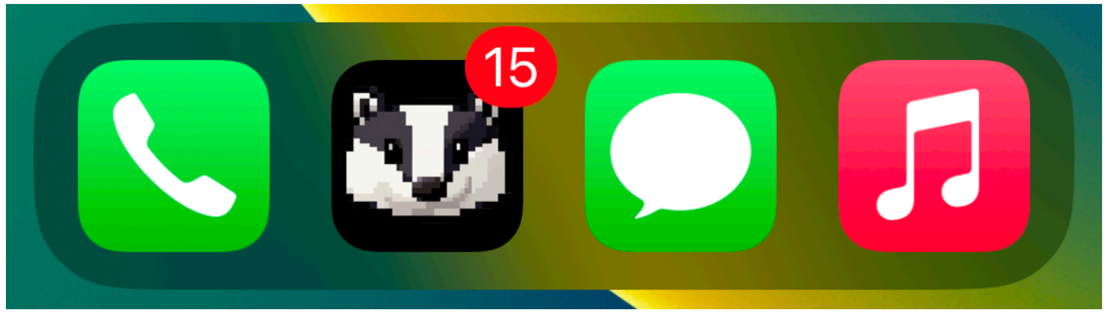
Showing the number is not guaranteed

Setting the badge as zero will clear it

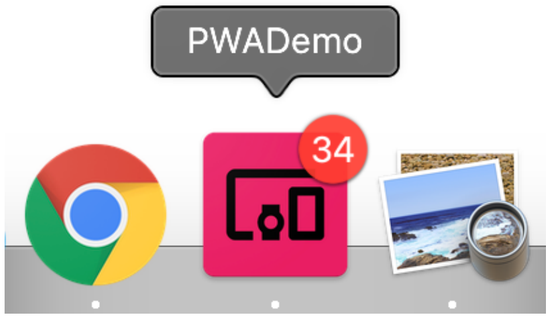
Platforms are using a best effort scenario

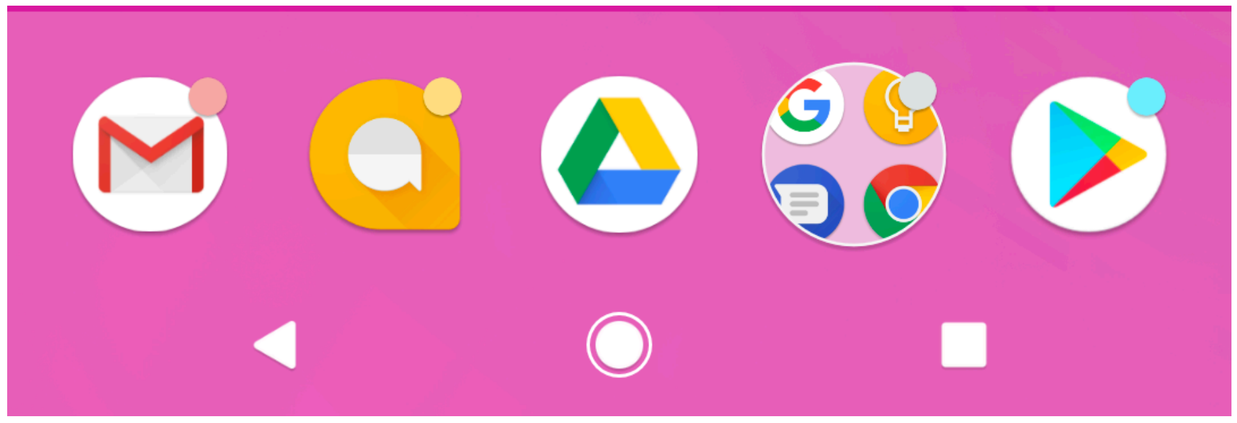
Some platforms, such as some Android devices don't support badges, so they use a notification dot instead

Other platforms might crop or display large numbers in different ways, such as showing 99+ in the badge instead of 1285


15


76


PWADemo
34



App Badging API

We can use the API from the Service Worker

```
if ('setAppBadge' in navigator) {
  navigator.setAppBadge(24)
}

if ('clearAppBadge' in navigator) {
   navigator.clearAppBadge()
}
```

App Shortcuts

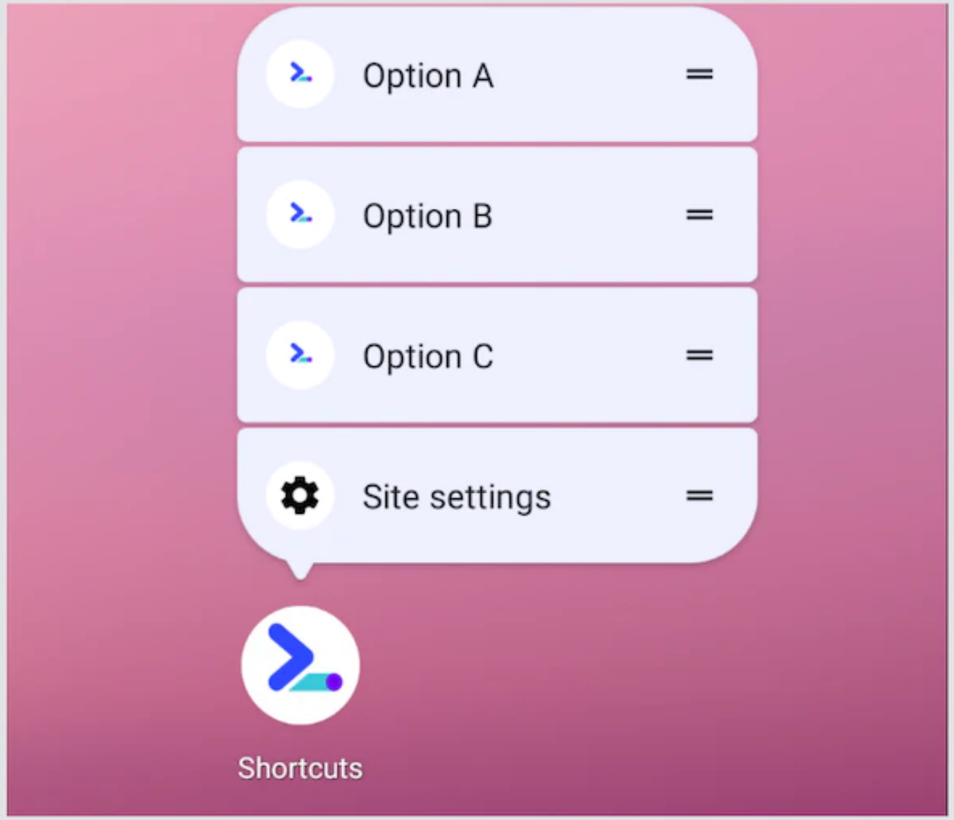
App Shortcuts

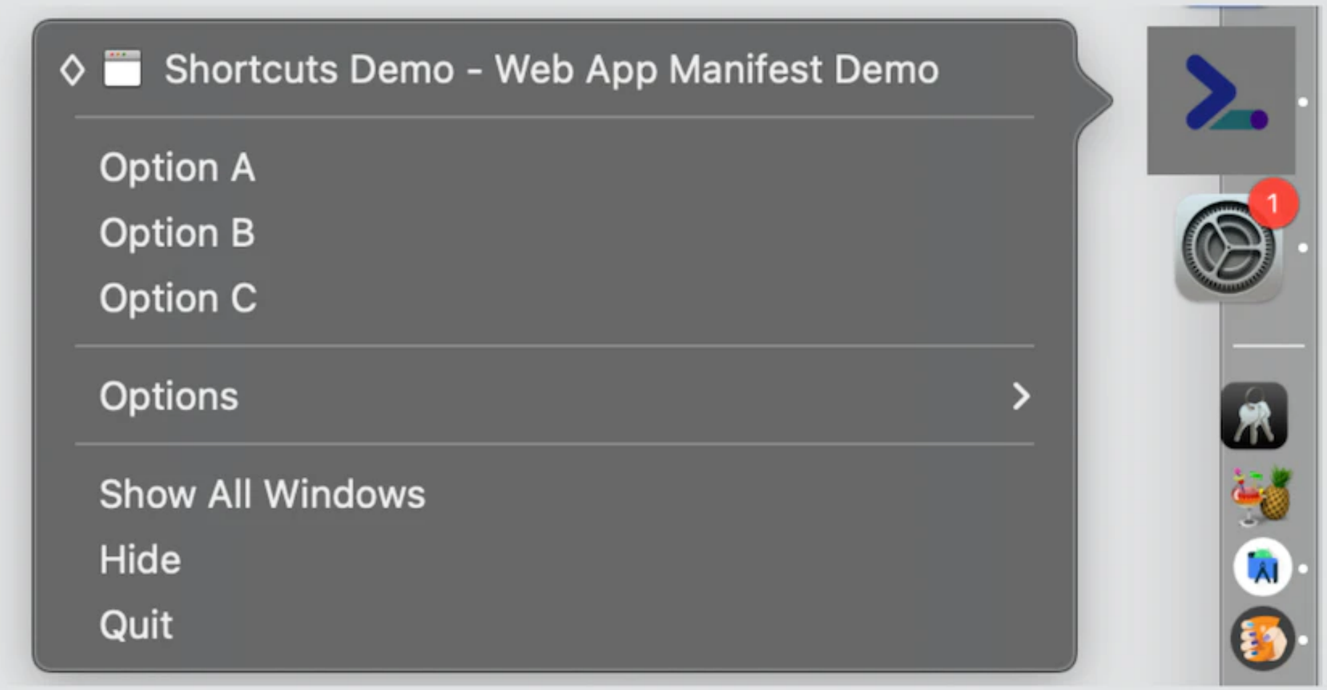
Menu items that will appear on icon launcher's
contextual menu (right click or long press)

Each menu item will open a navigation within
the PWA to a different URL

Menu items are defined statically per app

Changing them requires changing the
App Manifest


Option A
Option B
Option C
Site settings
Shortcuts


Shortcuts Demo - Web App Manifest Demo
Option A
Option B
Option C
Options
Show All Windows
Hide
Quit

App Shortcuts

New keys in Manifest; same rules as UI top-level properties

```
"shortcuts": [{
    "name": "Open a Second Action",
    "short_name": "Second",
    "description": "",
  "url": "/second-action",
  "icons": [{
    "src": "/icons/second.png",
    "sizes": "192x192",
    "type": "image/png"
  }]
}]
```

Experimental and New Capabilities

EyeDropper

Compression Streams

WebGPU

View Transitions APIs

Compute Pressure

Popover

Digital Goods API

Web Transport

Frontend Masters

What we've covered

- Web Capabilities
- Permissions and Security
- Compatibility
- Sensors, Location & Input
- Speech, Voice and Camera
- Hardware & Devices
- Integration with OS
- PWA Capabilities

THANKS! 🤗

A TOUR OF WEB CAPABILITIES

MAXIMILIANO FIRTMAN