

YES, YOUR BROWSER CAN DO THAT *(probably)*

A Look At Modern Web
APIs You Might Not Know

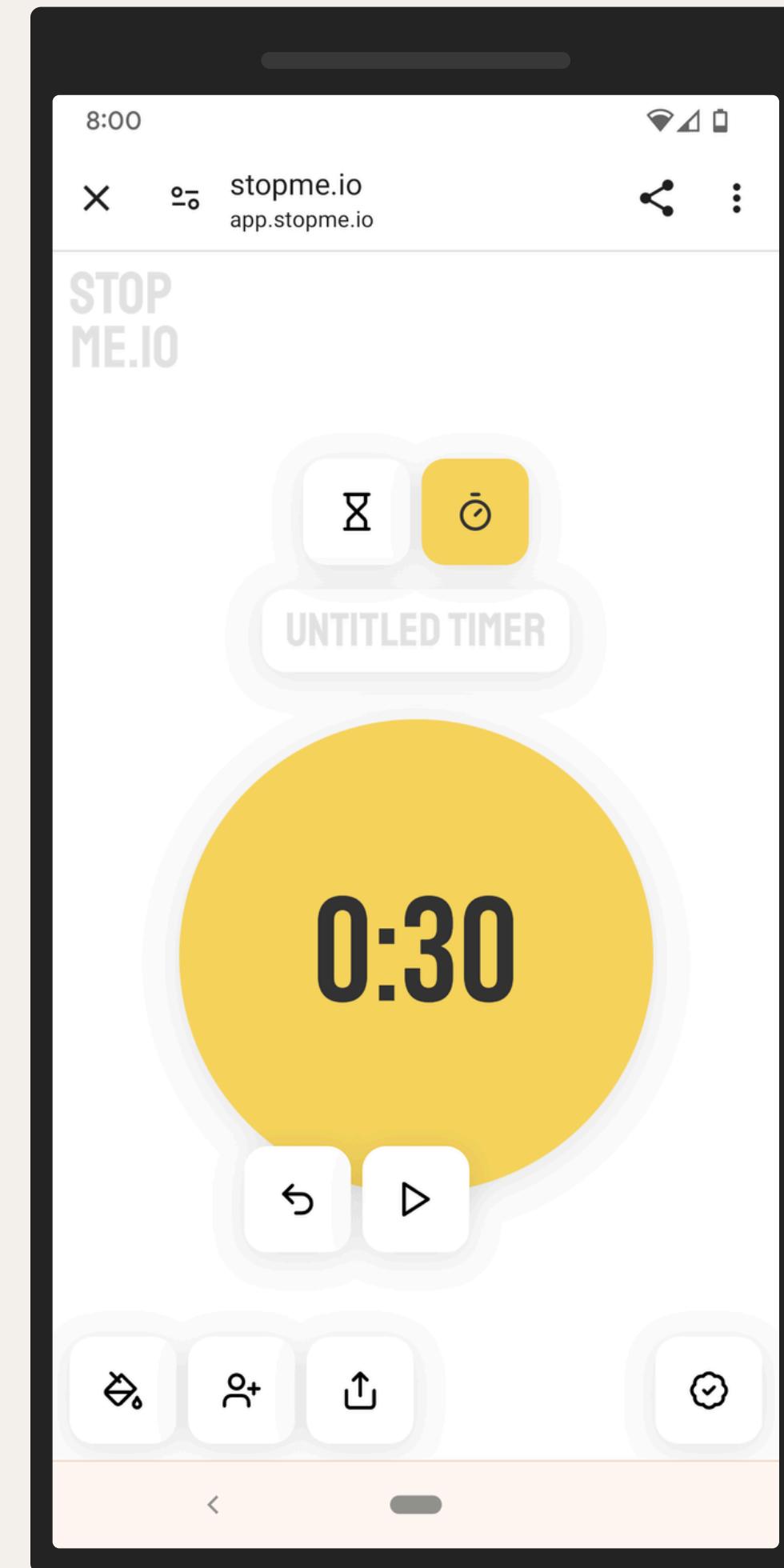
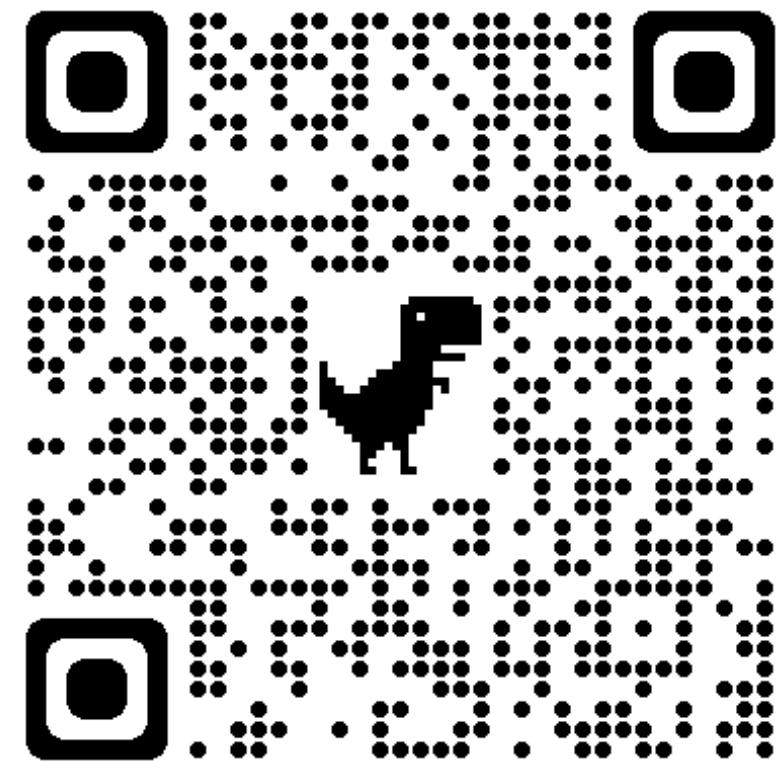
**WHY DOES IT
MATTER?**

Progressive enhancement is a design philosophy that provides a baseline of essential content and functionality to as many users as possible while delivering the best possible experience only to users of the most modern browsers

**LET'S LOOK AT SOME
BROWSER APIS**

STOPME.IO

The ultimate SaaS (Stopwatch as a Service) product for everyone

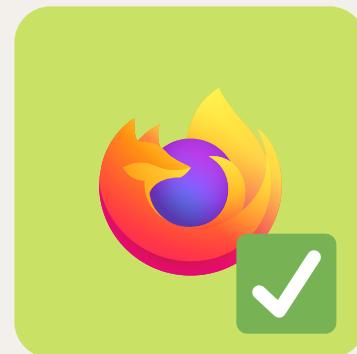


**OBSERVE
YOUR APP ...**

RESIZE & INTERSECTION OBSERVER

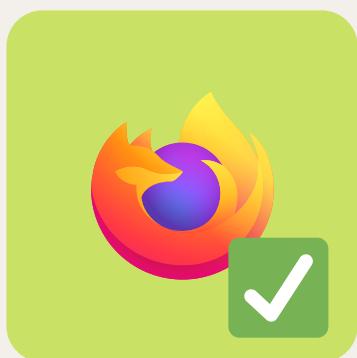


RESIZE & INTERSECTION OBSERVER



```
const callback = (entries) => {  
  entries.forEach((entry) => {  
    // *.contentBoxSize  
  });  
}  
  
const observer =  
  new ResizeObserver(callback, options);  
  
const el = document.querySelector("element");  
observer.observe(el);
```

RESIZE & INTERSECTION OBSERVER

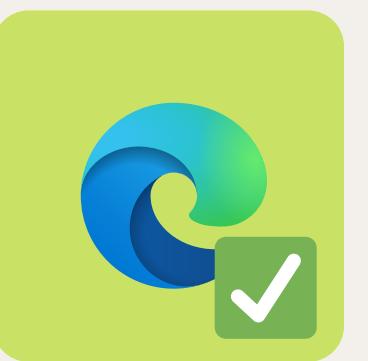


```
const callback = (entries) => {  
  entries.forEach((entry) => {  
    // *.isIntersecting  
    // *.intersectionRect  
    // *.intersectionRatio  
    // ...  
  });  
}  
  
const observer =  
  new IntersectionObserver(callback, options);  
  
const el = document.querySelector("element");  
observer.observe(el);
```

**... AND THE
USER'S DEVICE**

02

NETWORK INFORMATION API



NETWORK INFORMATION API



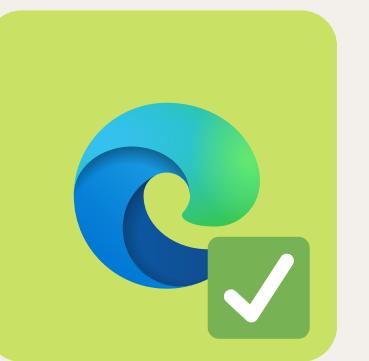
```
// Online status
let isOnline = navigator.onLine

const callback = () => {
  isOnline = navigator.onLine;
}

window.addEventListener("online", callback);
window.addEventListener("offline", callback);
```

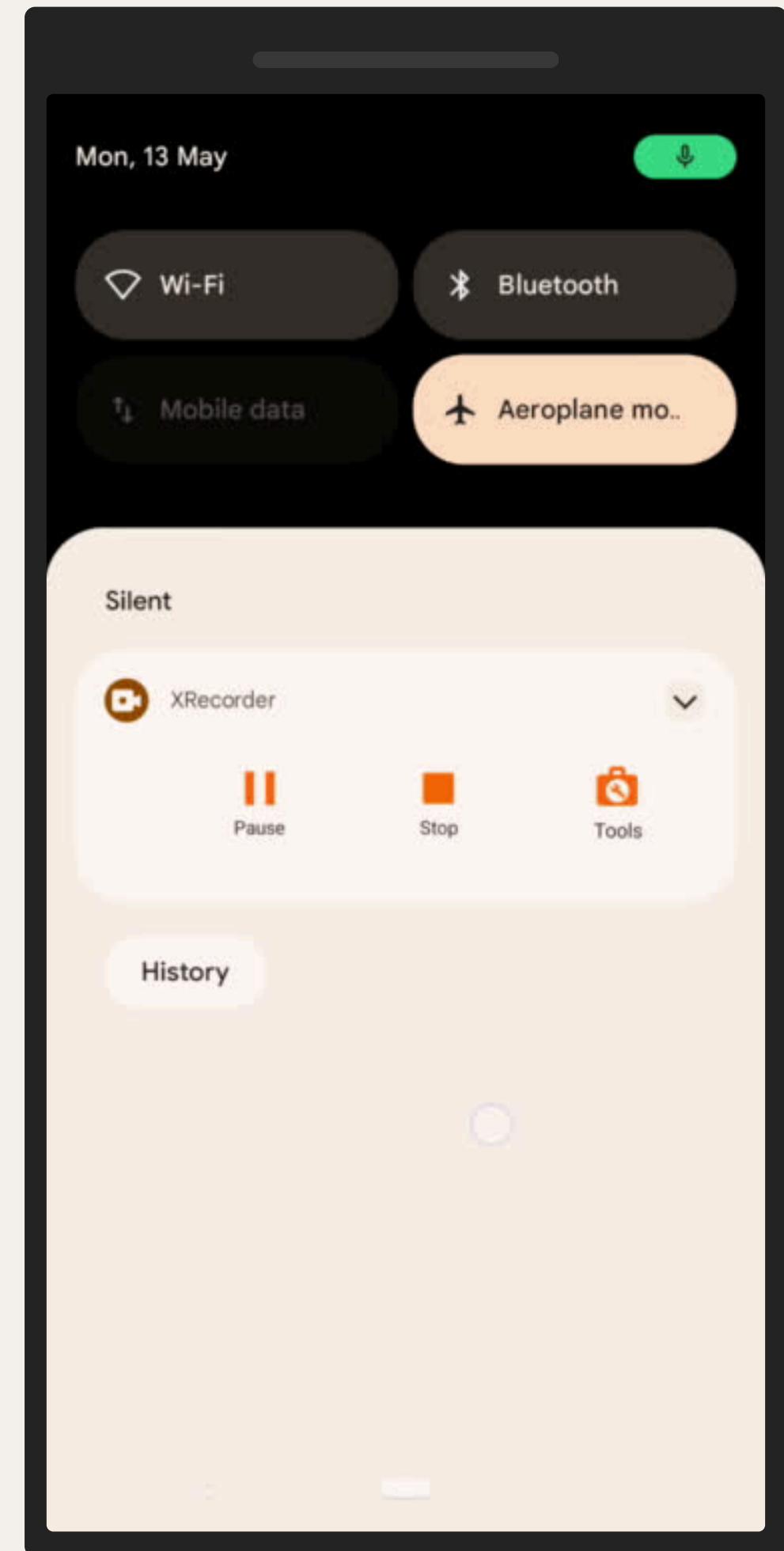
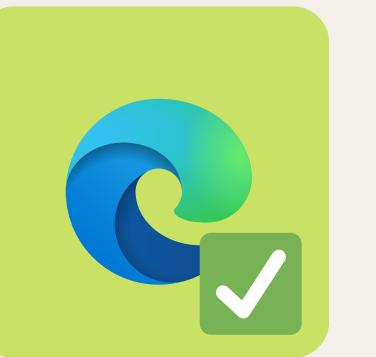
<https://developer.mozilla.org/en-US/docs/Web/API/Navigator/onLine>

NETWORK INFORMATION API



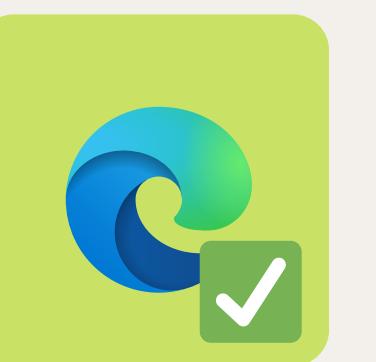
```
let userConnection = navigator.connection;  
// *.type (e.g. wifi, ...)  
// *.effectiveType (e.g. 2g, 3g, ...)  
// *.downlink  
// *.downlinkMax  
// *.rtt  
// *.saveData  
  
const callback = () => {  
  userConnection = navigator.connection;  
}  
  
navigator.connection  
  .addEventListener("change", callback);
```

NETWORK INFORMATION API

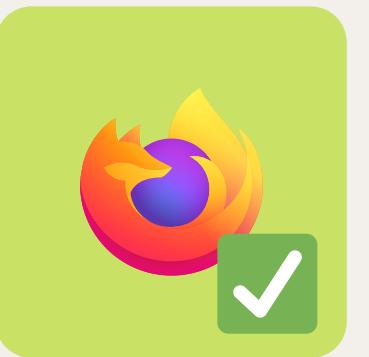


053

PAGE VISIBILITY API



PAGE VISIBILITY API



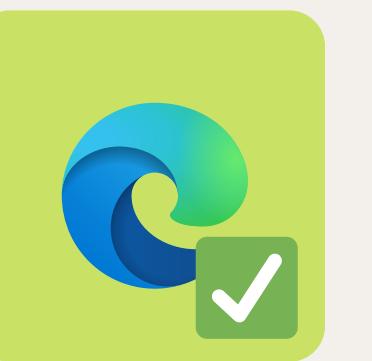
```
let isHidden = document.hidden;  
  
const callback = () => {  
  isHidden = document.hidden;  
}  
  
document.addEventListener(  
  "visibilitychange",  
  callback  
);
```

PAGE VISIBILITY API



A screenshot of a web browser window titled "0:25 – stopme.io". The address bar shows "webdirections.org/code/". The page content is the homepage for "WEB DIRECTIONS CODE 24" in Melbourne, June 20 & 21. The main title is "WEB DIRECTIONS CODE" in large white letters, with "MELBOURNE" and "JUNE 20 & 21" below it. Below the title, the text reads "The world leading conference for JavaScript Developers and Front End Engineers". At the bottom, there is a pink banner with the text "Code is for front end, JavaS Web developers, engineer managers, devops experts.".

BATTERY STATUS API



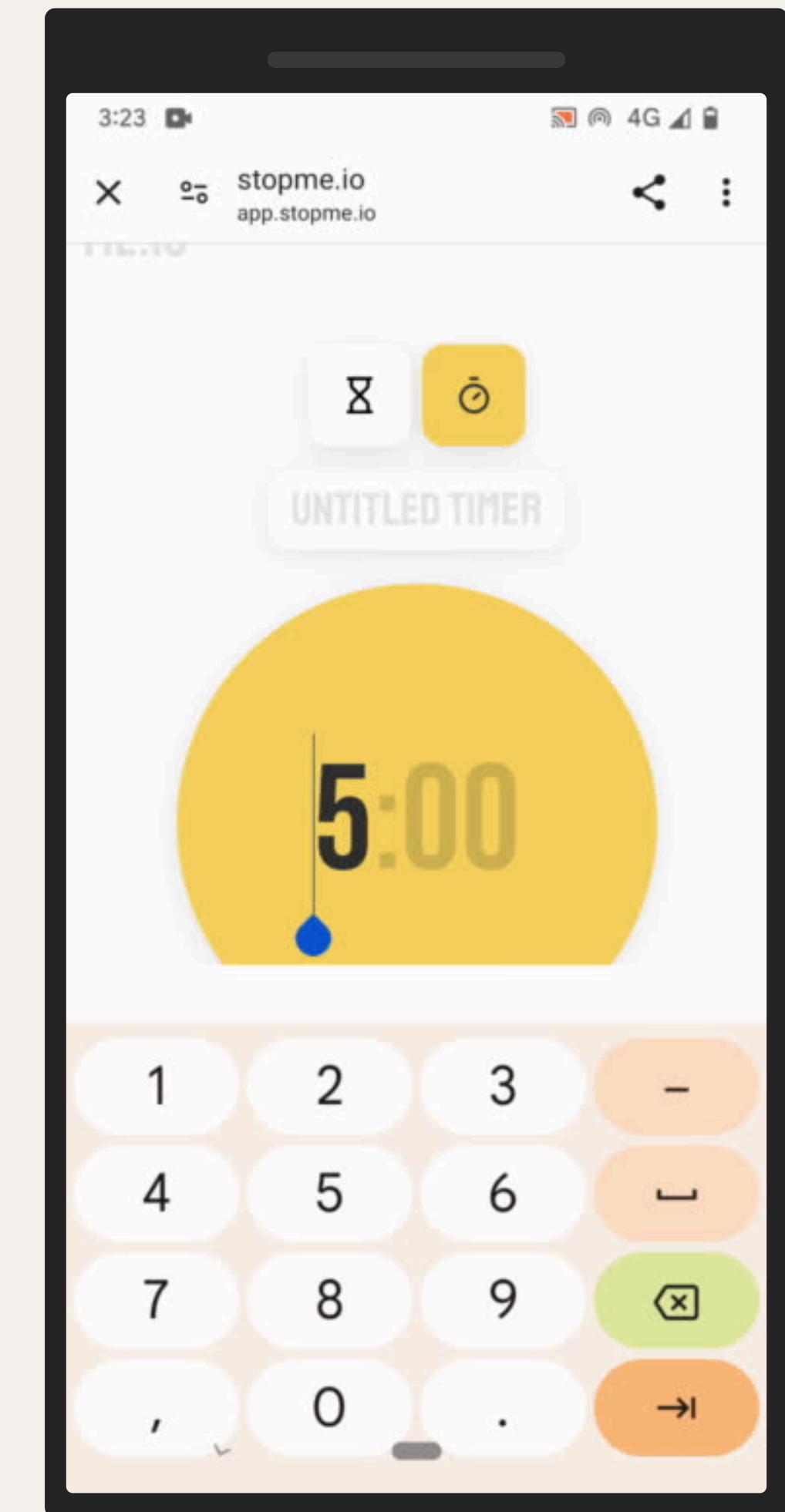
BATTERY STATUS API



```
const info = await navigator.getBattery();
// *.level
// *.charging
// *.chargingTime
// *.dischargingTime

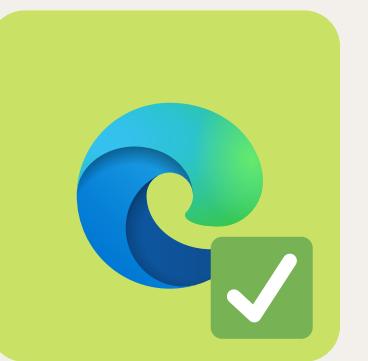
info.addEventListener("levelchange", ...);
info.addEventListener("chargingchange", ...);
// ...
```

BATTERY STATUS API



**ENHANCE YOUR
COMPONENTS**

VIBRATION API



VIBRATION API



```
navigator.vibrate(duration);

// Patterns
// e.g. vibrate for 100ms with 50ms pauses
navigator.vibrate([100, 50, 100, 50, 100]);

// Stop long vibration or pattern
navigator.vibrate(0);
```

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

VIBRATION API



```
// SOS in Morse
navigator.vibrate([
  100, 30, 100, 30, 100, 30, 200, 30, 200,
  30, 200, 30, 100, 30, 100, 30, 100
]);

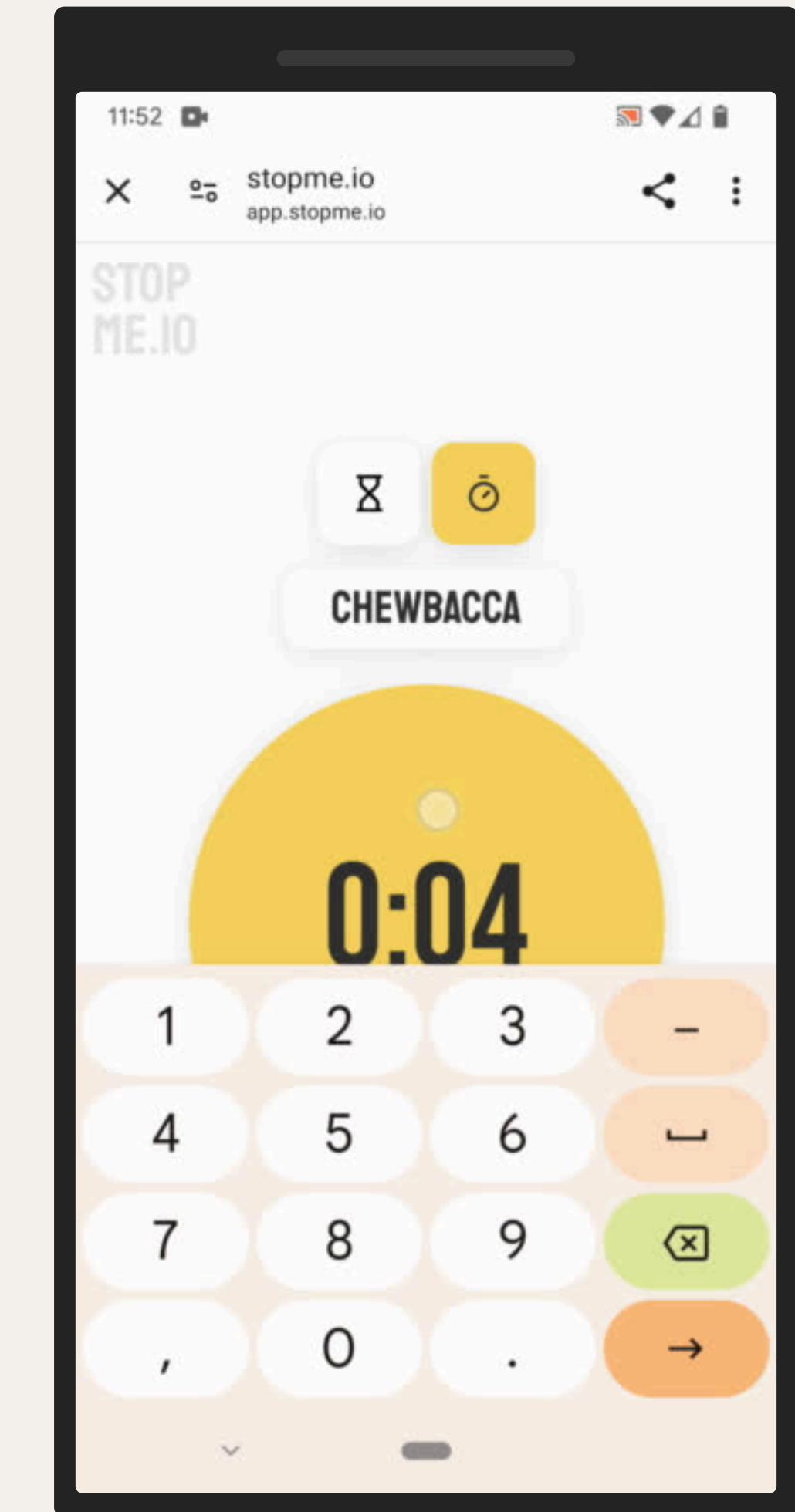
// Super Mario
navigator.vibrate([
  125, 75, 125, 275, 200, 275, 125, 75,
  125, 275, 200, 600, 200, 600
]);

// Star Wars
navigator.vibrate([
  500, 110, 500, 110, 450, 110, 200, 110,
  170, 40, 450, 110, 200, 110, 170, 40, 500
]);
```

VIBRATION API

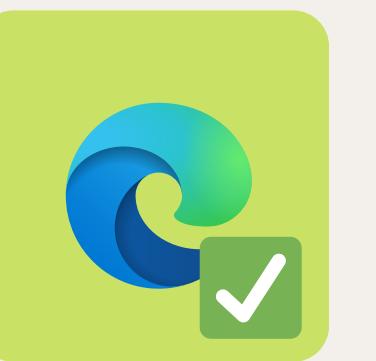


*It vibrates,
I swear!*

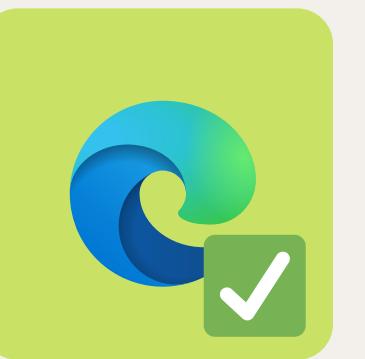


06

EYEDROPPER API



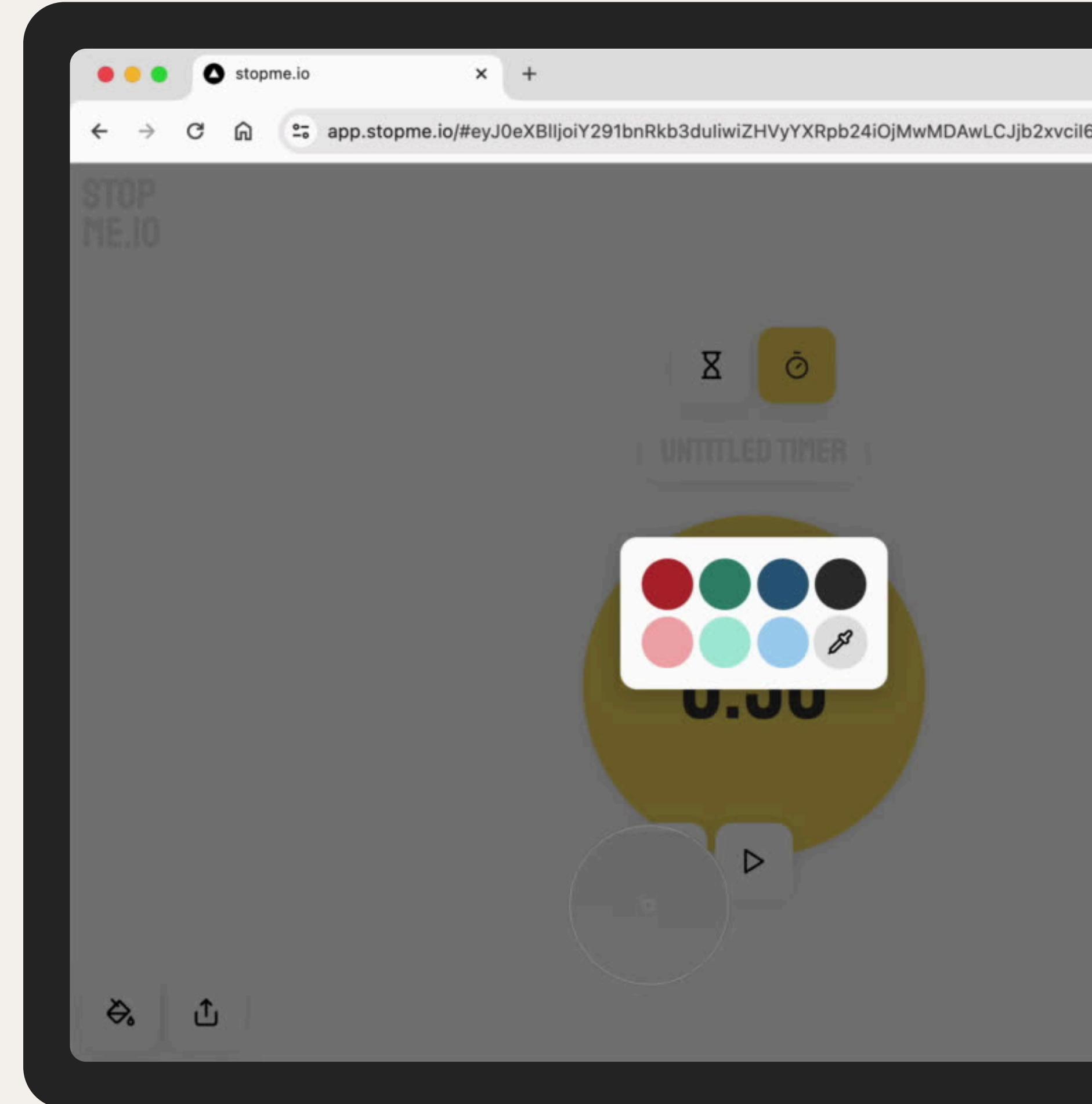
EYEDROPPER API



```
const eyeDropper = new EyeDropper();

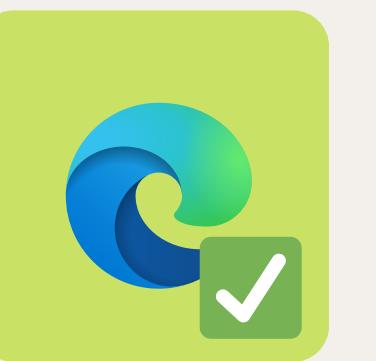
document.getElementById("btn")
  .addEventListener("click", () => {
    // Needs to be triggered by user action
    eyeDropper.open()
      .then((result) => {
        // Returns the selected color
        // *.sRGBHex
      })
      .catch((e) => {
        // Catches any errors, including
        // when the user cancels selection
      });
});
```

EYEDROPPER API



**ALMOST AS GOOD
AS NATIVE**

WEB SHARE API



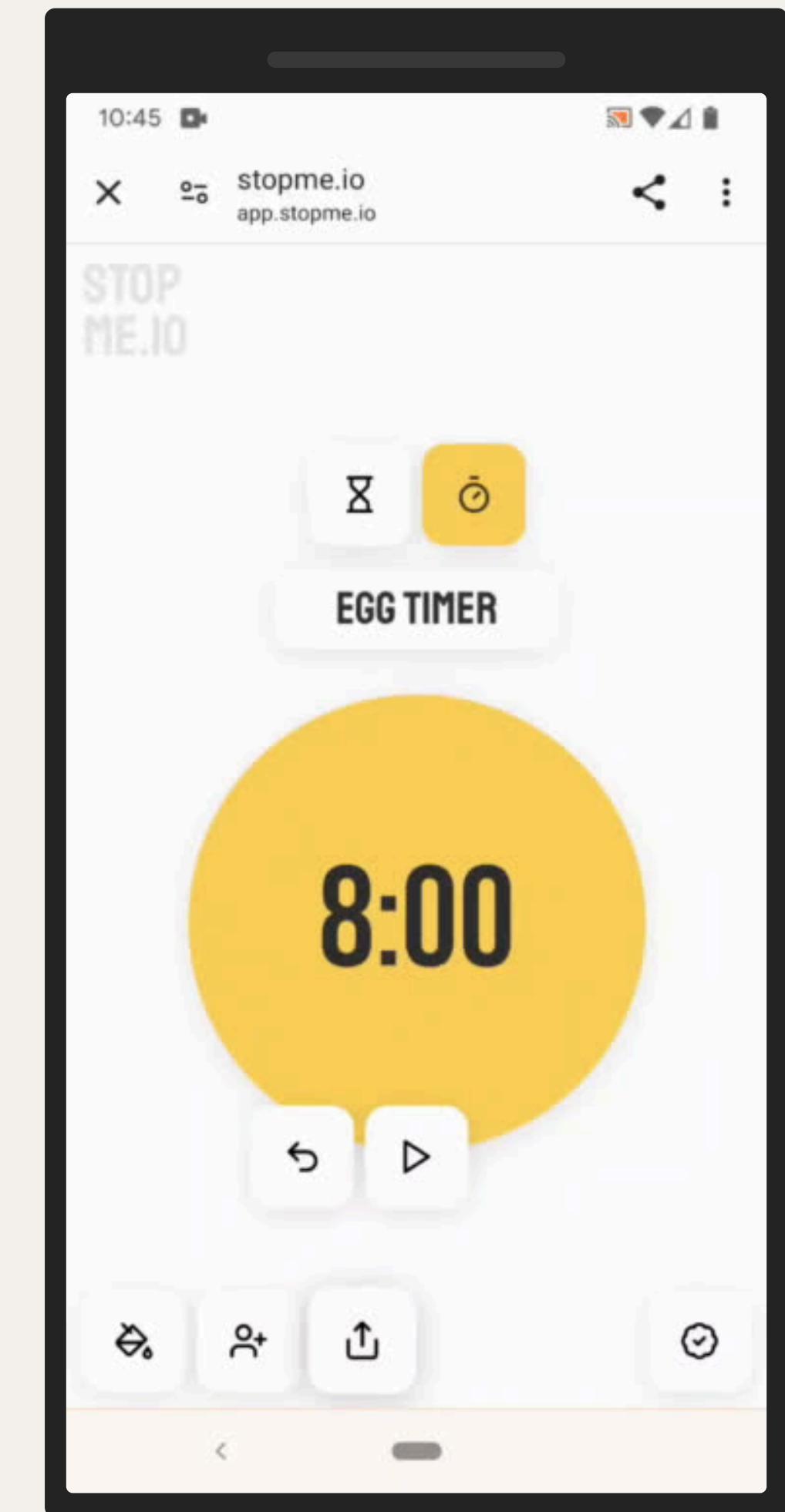
WEB SHARE API



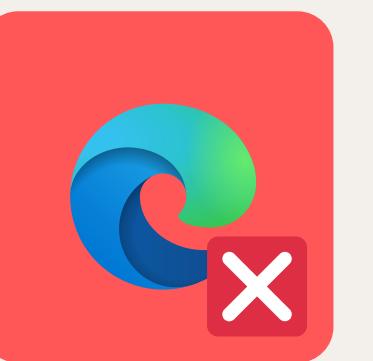
```
const shareData = {  
  title: "../NEW",  
  text: "Celebrate all things technology",  
  url: "https://slashnew.tech"  
  files: [imageFile, videoFile]  
};  
  
document.getElementById("btn")  
  .addEventListener("click", () => {  
    navigator.share(shareData)  
    .catch((e) => {  
      // Handle any errors  
    });  
});
```

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

WEB SHARE API



CONTACT PICKER API



CONTACT PICKER API

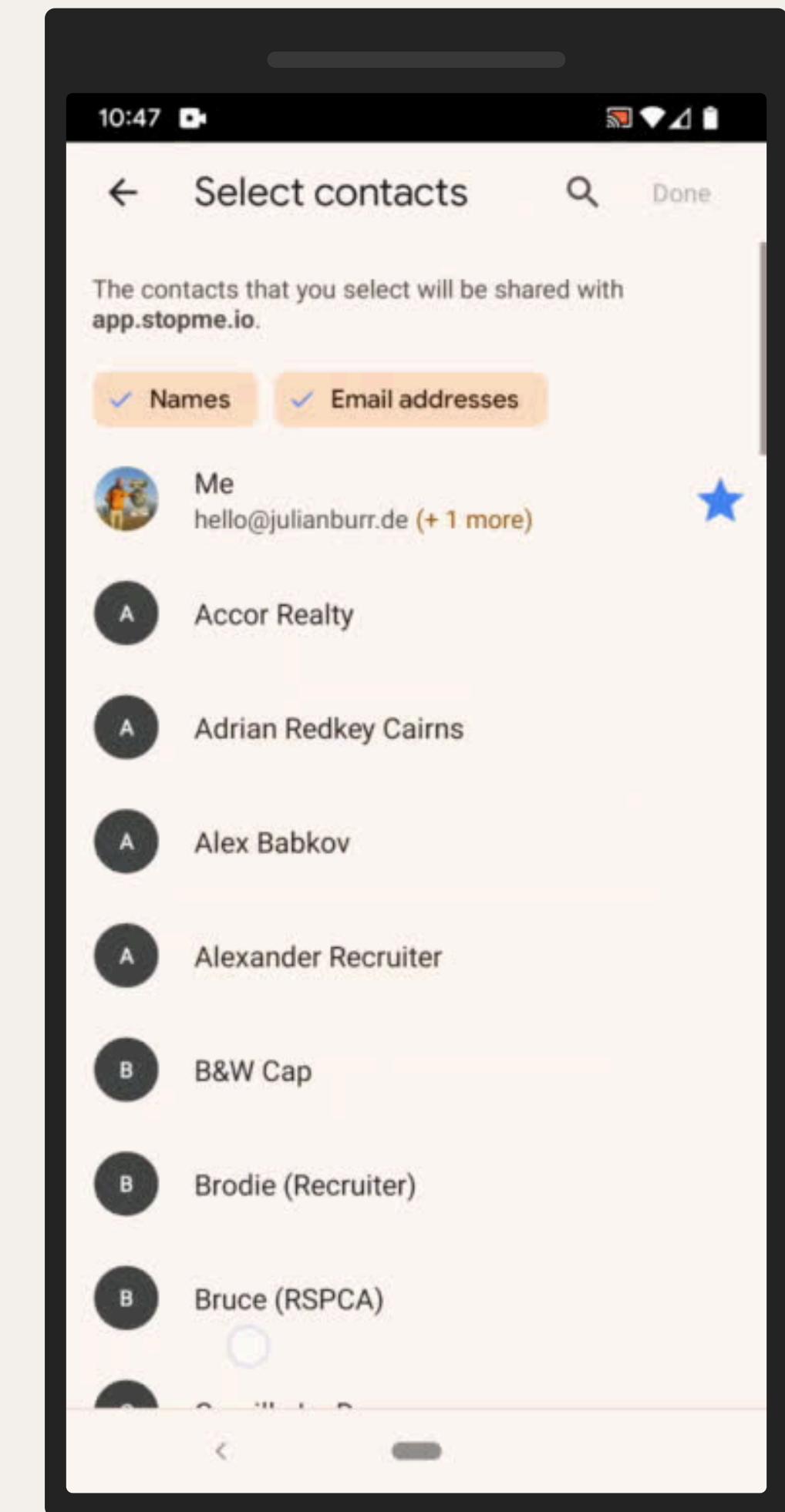


```
const props = ["name", "email", "tel", "icon"];
const opts = { multiple: true };

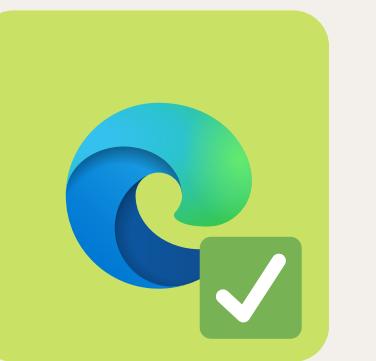
const contacts = await navigator.contacts
  .select(props, opts);
  .catch((e) => {
    // Handle any errors
 });
```

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

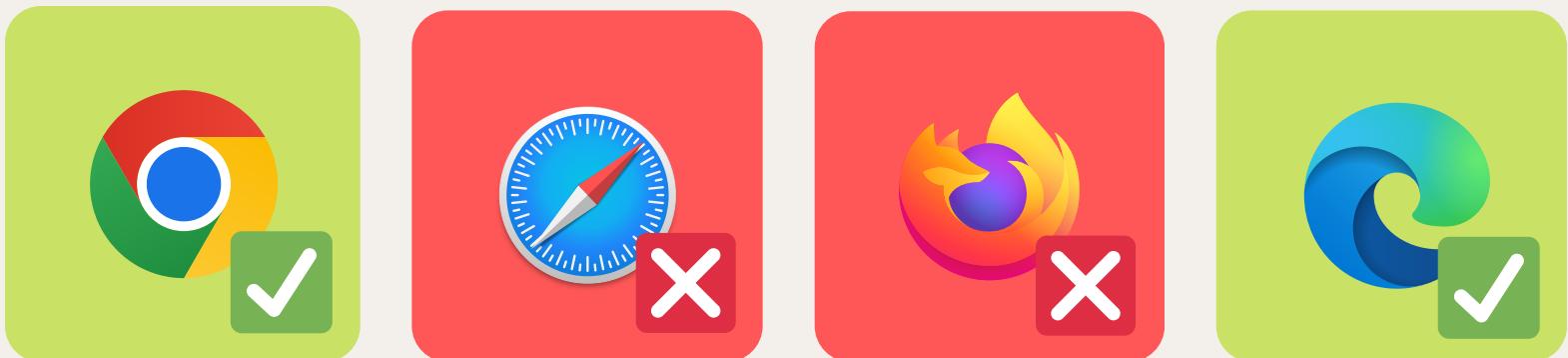
CONTACT PICKER API



WEB OTP API

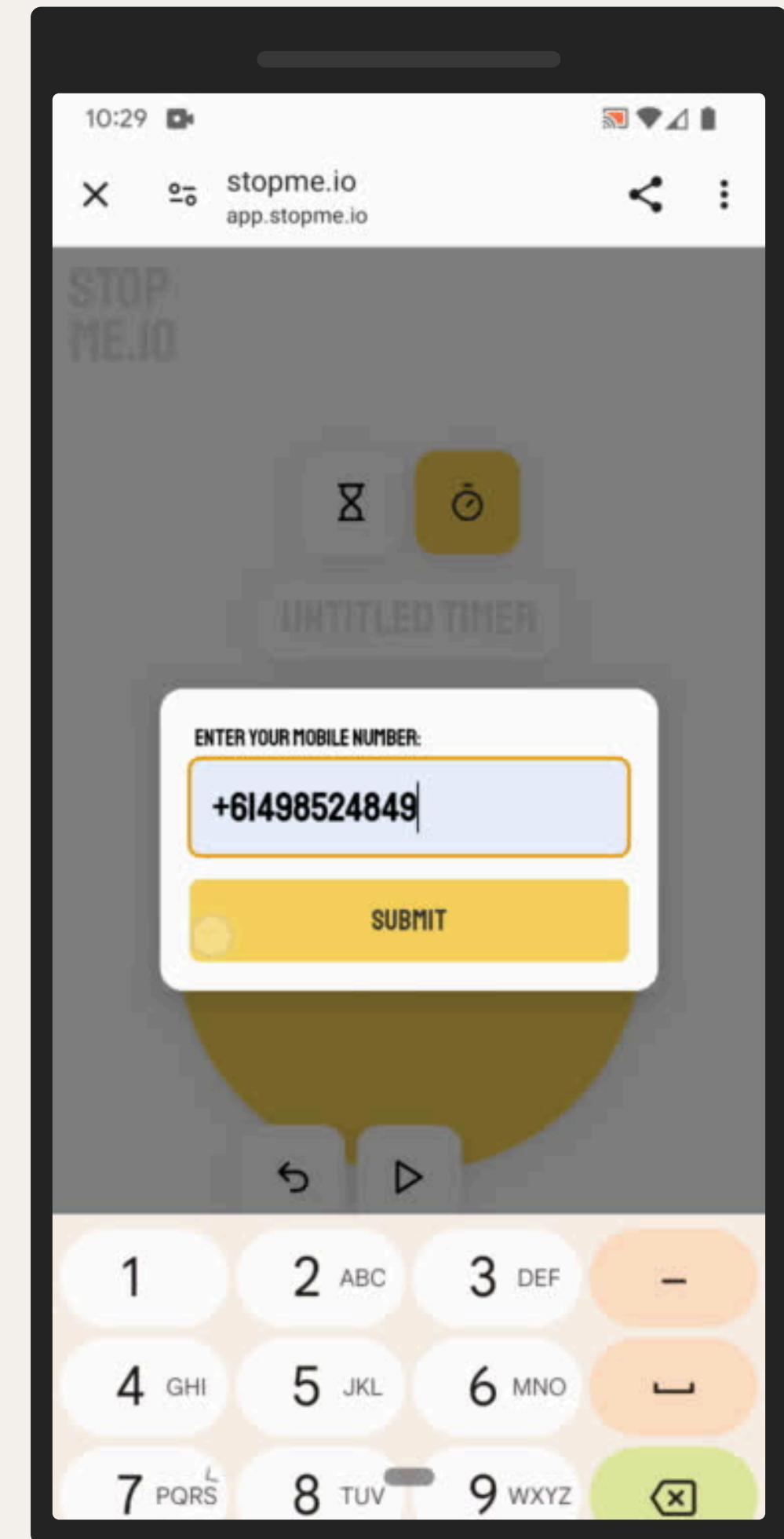
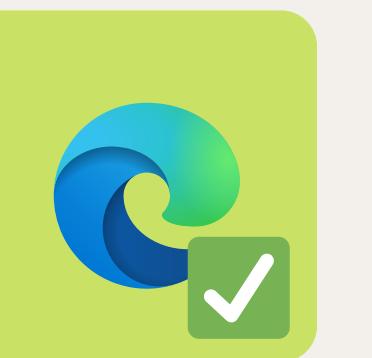


WEB OTP API

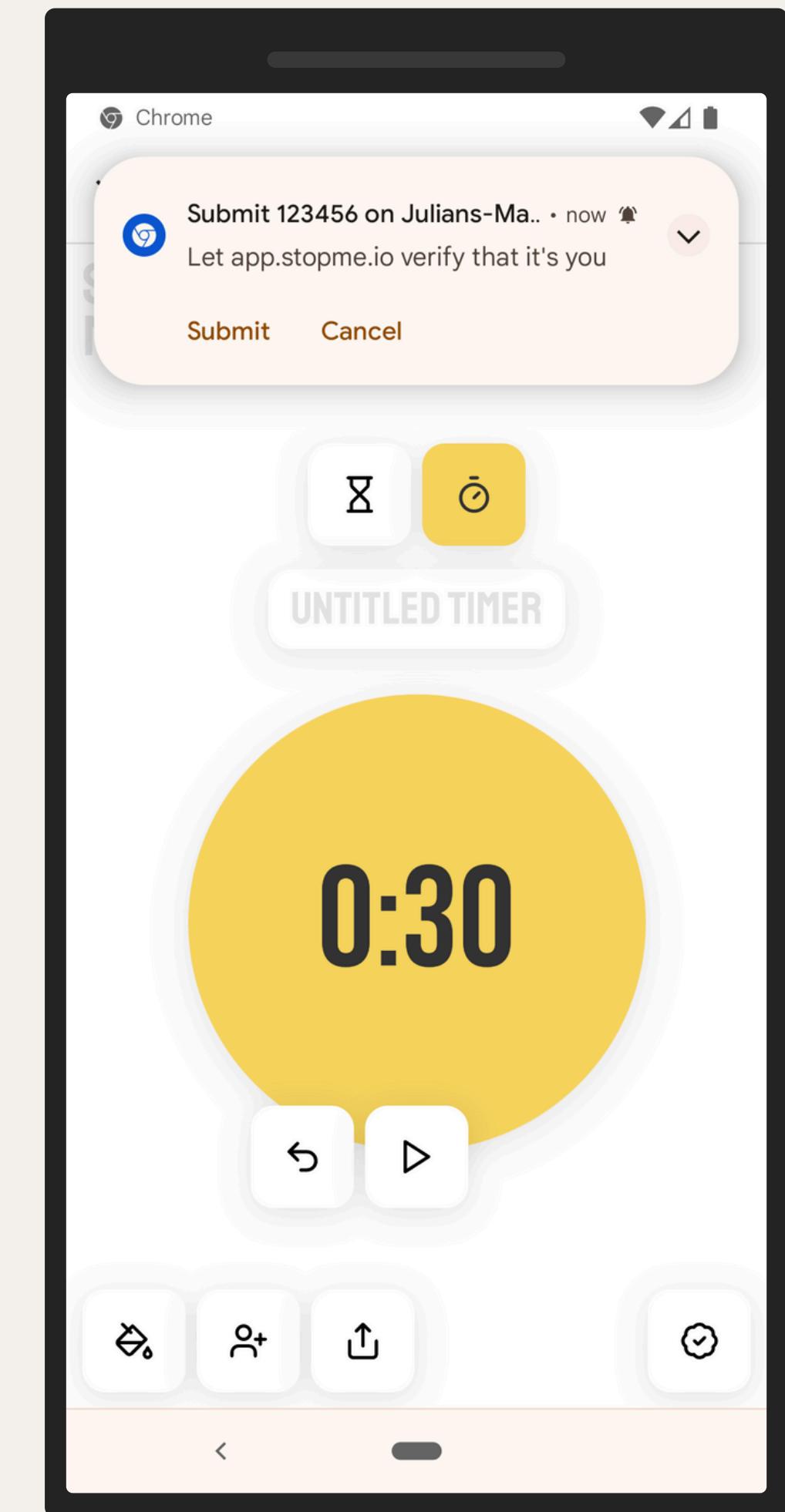


```
navigator.credentials  
  .get({  
    otp: { transport: ["sms"] },  
    signal: ac.signal,  
  })  
  .then((otp) => {  
    // *.code  
  })  
  .catch((e) => {  
    // Handle any errors  
});  
  
// Format of the SMS so it can be processed  
// Your verification code is 123456.\n\n  
// @app.stopme.io #123456
```

WEB OTP API



WEB OTP API



HONOURABLE Mentions

DEBUGGING

CONSOLE

PERFORMANCE

MEMORY

DEBUGGING

CONSOLE

PERFORMANCE

MEMORY

```
console.log(...);  
  
console.assert(...);  
console.count(...);  
console.countReset(...);  
  
console.dir(...);  
console.table(...);  
console.group(...);  
console.groupCollapsed(...);  
console.groupEnd(...);  
  
console.time(...);  
console.timeEnd(...);  
  
console.trace(...);  
  
console.clear();
```

DEBUGGING

CONSOLE

PERFORMANCE

MEMORY

```
const time = performance.now();

performance.mark("start");
performance.mark("end", { detail: { ... } });

performance.measure("login", "start", "end");

const observer =
  new PerformanceObserver((list, obj) => {
    list.getEntries().forEach((entry) => {
      // *.name
      // *.startTime
      // *.duration
      // *.detail
    });
  });
observer.observe({ type: "resource" });
```

DEBUGGING

CONSOLE

PERFORMANCE

MEMORY

```
const memory = navigator.deviceMemory;  
// Device has at least ${memory}GiB of RAM  
  
const memorySample = await performance  
    .measureUserAgentSpecificMemory();  
// *.bytes  
// *.breakdown[ ].bytes  
// *.breakdown[ ].attribution  
// *.breakdown[ ].types
```

**AUTH BUT
BETTER**

CREDENTIALS API

AUTHN API

AUTH BUT BETTER

CREDENTIALS API

AUTHN API

```
const cred = new PasswordCredential({  
  id,  
  password,  
  name  
});  
  
// Store credentials  
await navigator.credentials.store(cred);  
  
// Get stored credentials  
const user = await navigator.credentials.get();  
  
// Prevent automatic login when user signs out  
await navigator  
  .credentials  
  .preventSilentAccess();
```

AUTH BUT BETTER

CREDENTIALS API

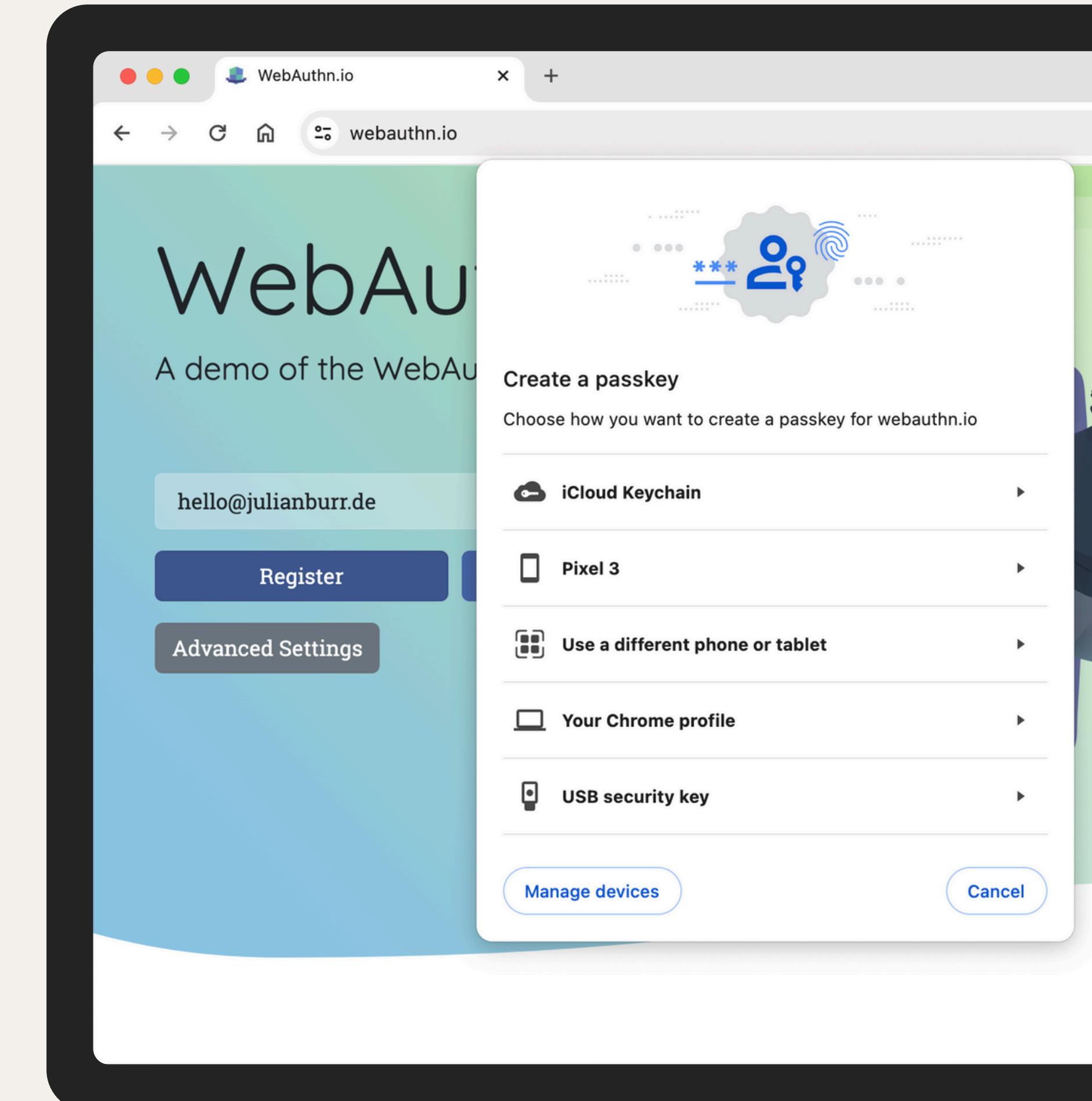
AUTHN API

```
// Create credentials object on the client  
// using challenge generated on the server  
const registerCredential =  
  await navigator.credentials.create({  
    publicKey  
});  
  
// Credentials are stored with the user  
// identity are sent back and stored on the  
// server  
  
// Authenticate again using server challenge  
const authCredential =  
  await navigator.credentials.get({  
    publicKey  
});  
  
// Use stored public key to verify validity  
// of auth credentials
```

AUTH BUT BETTER

CREDENTIALS API

AUTHN API



THE NEXT GENERATION

INTL API

TEMPORAL*

NAVIGATION API

THE NEXT GENERATION

INTL API

TEMPORAL*

NAVIGATION API

```
const options = {  
    dateStyle: "full",  
    timeStyle: "long",  
    timeZone: "Australia/Sydney"  
});  
  
new Intl.DateTimeFormat("en-US", options)  
    .format(date);  
// Friday, December 2, 2022 at 12:21:40 PM  
// GMT+11  
  
// Relative time  
const fmt = new Intl.RelativeTimeFormat(  
    "en",  
    { style: "narrow" }  
);  
fmt.format(3, "day"); // in 3 days  
fmt.format(-2, "year"); // 2 years ago
```

THE NEXT GENERATION

INTL API

TEMPORAL*

NAVIGATION API

```
const au = new Intl.NumberFormat("en-AU");
au.format(123_456.79);
// 123,456.79

const de = new Intl.NumberFormat("de-DE");
de.format(123_456.79);
// 123.456,79

const fmt = new Intl.NumberFormat(
  "de-DE",
  { style: "currency", currency: "EUR" }
);
fmt.format(123_456.79)
// 123.456,79 €
```

THE NEXT GENERATION

INTL API

TEMPORAL*

NAVIGATION API

```
// Get exact current system time
Temporal.Now.instant();

// Get time zone
Temporal.Now.timeZoneId();

// Useful utilities
const date = Temporal.PlainDate.from(dateStr);
// *.year
// *.inLeapYear
// *.toString()
// ...

// Manipulate time
date.add({ hours: 1 });
```

THE NEXT GENERATION

INTL API

TEMPORAL*

NAVIGATION API

```
// Promises for the win (finally!)
await navigation.reload({ info, state });

// Navigate around
await navigation.navigate(url, options);
await navigation.back(options);
await navigation.forward(options);
await navigation.traverseTo(key, options);

// Events
navigation
  .addEventListener("currententrychange");
  .addEventListener("navigate");
  .addEventListener("navigatesuccess");
  .addEventListener("navigateerror");
```

THE NEXT GENERATION

URL PATTERNS

POPOVER

VIEW TRANSITION

```
const p = new URLPattern({ pathname: "/foo" });
p.test("https://example.com/books")); // true
```

```
const p =
  new URLPattern({ pathname: "/books/:id" });
const match =
  p.exec("https://example.com/books/123");
// *.pathname.groups.id = 123
```

```
const p =
  new URLPattern(
    "/books/:id(\d+)",
    "https://example.com"
);
```

THE NEXT GENERATION

URL PATTERNS

POPOVER

VIEW TRANSITION

A dark-themed window with three colored window control buttons (red, yellow, green) at the top. Inside, there is sample code demonstrating the use of the Popover API.

```
<button popovertarget="mypopover">  
  Toggle the popover  
</button>  
  
<div id="mypopover" popover>  
  Popover content  
</div>  
  
document.addEventListener("keydown", (e) => {  
  if (e.key === "h") {  
    mypopover.togglePopover();  
  }  
});
```

THE NEXT GENERATION

URL PATTERNS

POPOVER

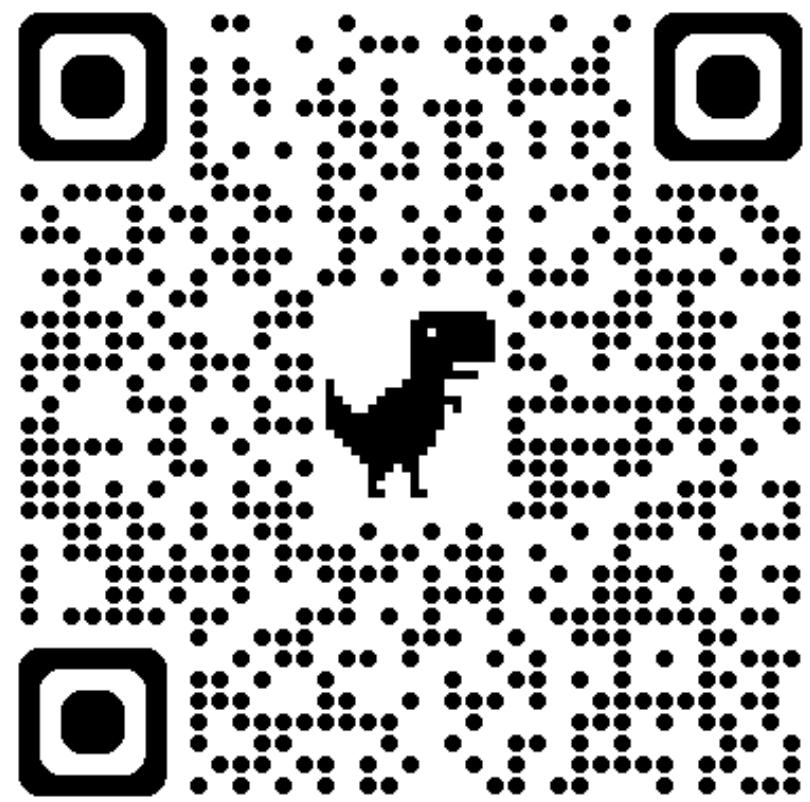
VIEW TRANSITION

A dark-themed code editor window with three colored window controls (red, yellow, green) at the top. The code inside the window is as follows:

```
function updateView(e) {
  e.preventDefault();
  const details =
    e.target.closest("details");

  if (!document.startViewTransition) {
    // Fallback for older browsers
    details.toggleAttribute("open")
  }

  document.startViewTransition(() =>
    details.toggleAttribute("open")
  );
}
```



THANKS!

@JBURR90 / JULIAN BURR

**HTTPS://WWW.JULIANBURR.DE/
WEBDIRECTIONS-CODE-2024-
SLIDES.PDF**