

PLATINUM



Octopus Deploy



GOLD



SILVER



STANDARD





@JBURR90 / JULIAN BURR
DDD BRISBANE

YES, YOUR
BROWSER CAN
DO THAT!
(PROBABLY)

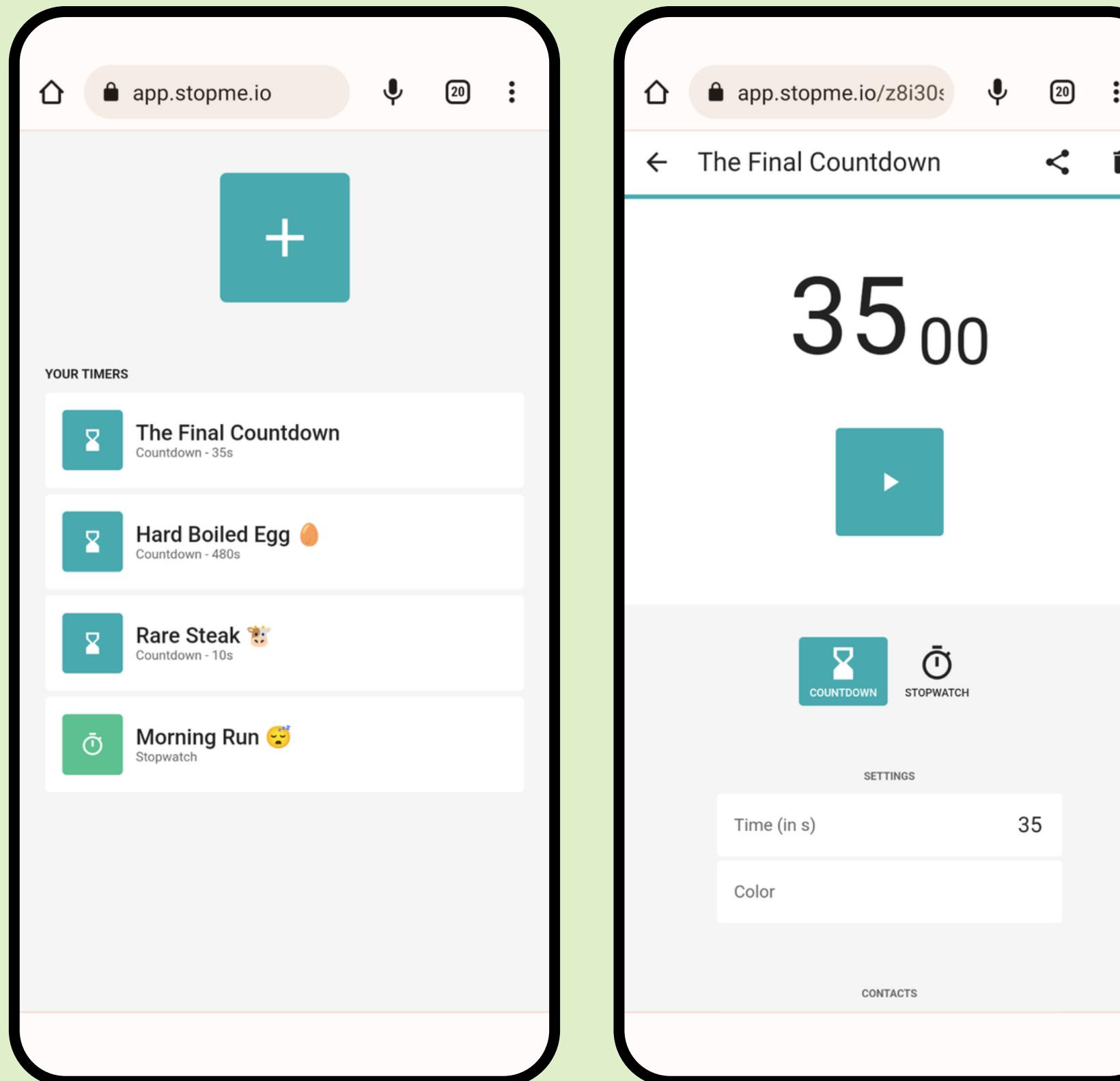
WHY DO BROWSER APIS MATTER?

A very brief history of frontend
development and the constant desire to
build native

STOPME.IO

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

STOP
ME.IO



01

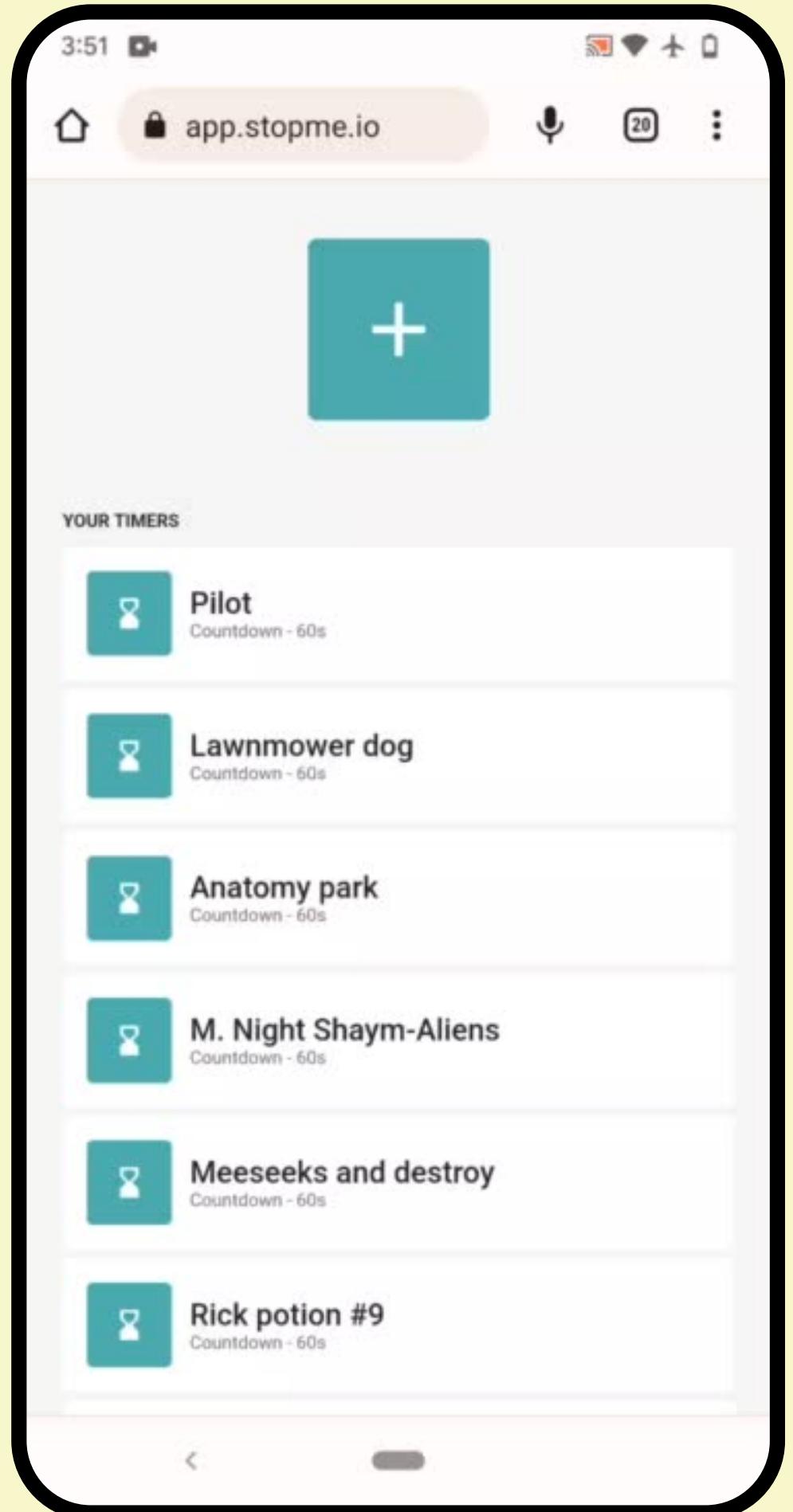
OBSERVE

INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport

INTERSECTION OBSERVER

Keep track of when DOM elements enter or leave the users viewport



INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new IntersectionObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new IntersectionObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new IntersectionObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

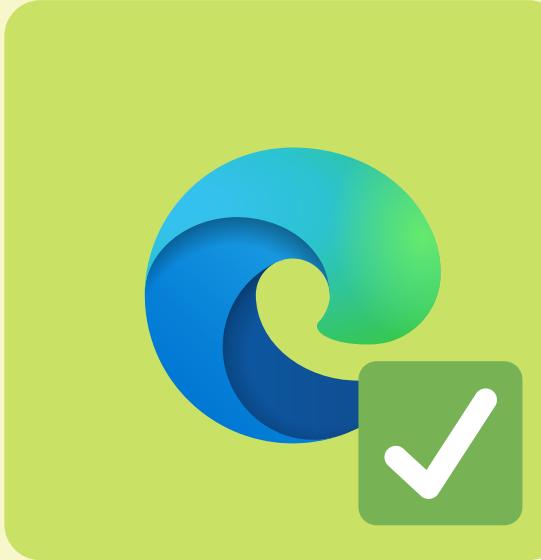
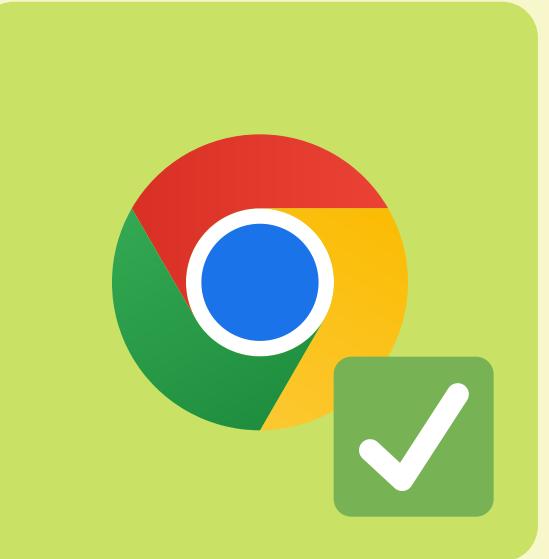
INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new IntersectionObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

INTERSECTION OBSERVER

Keep track of when DOM elements
enter or leave the users viewport



RESIZE OBSERVER

Observe DOM elements and run
callbacks whenever their content
box changes

RESIZE OBSERVER

Observe DOM elements and run callbacks whenever their content box changes

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new ResizeObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

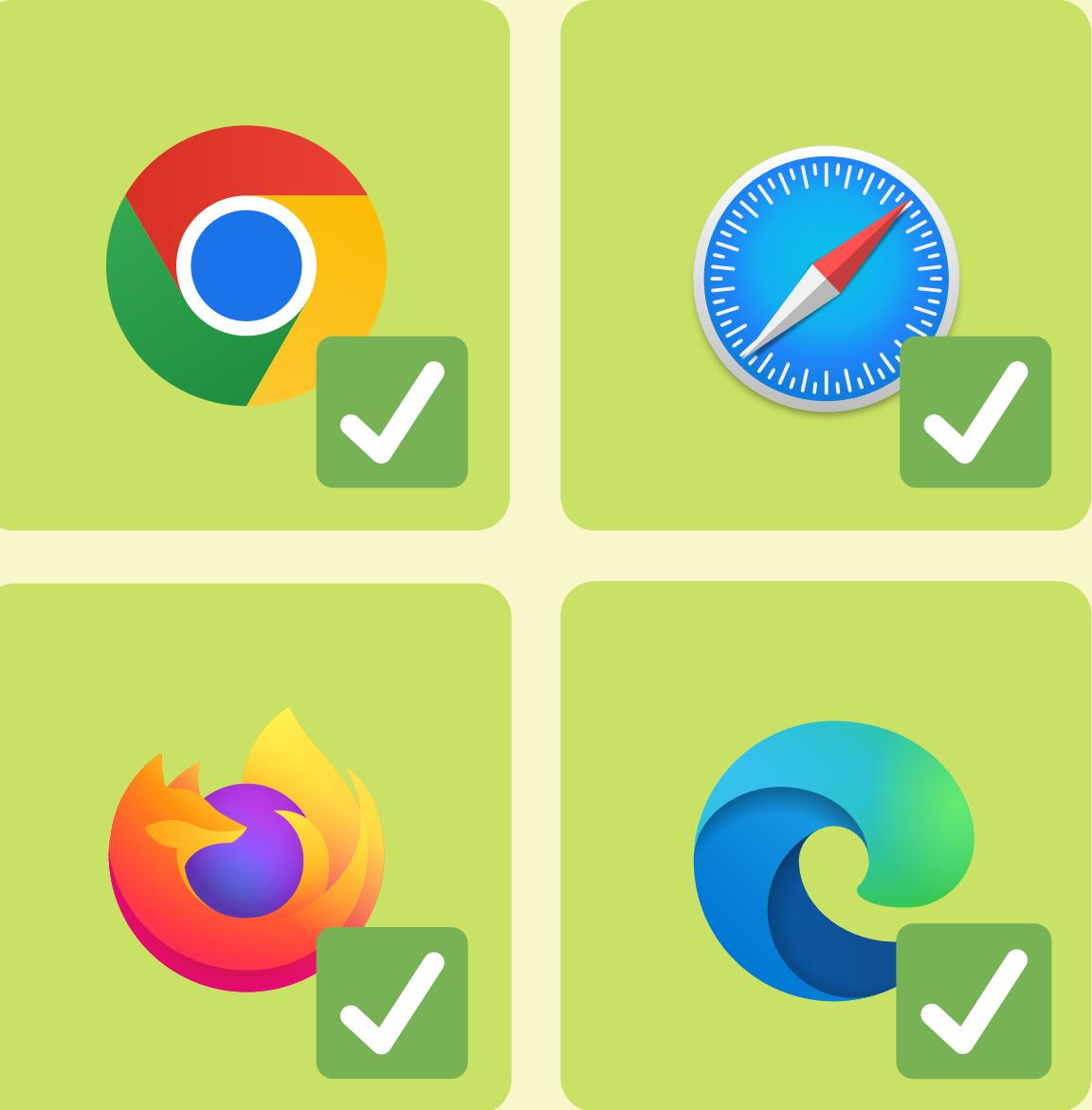
RESIZE OBSERVER

Observe DOM elements and run callbacks whenever their content box changes

```
const callback =  
  (entries, observer) => {  
    entries.forEach(entry => {  
      // ...  
    })  
  }  
  
const observer =  
  new ResizeObserver(  
    callback,  
    options  
  )  
  
observer.observe(element)  
observer.unobserve(element)
```

RESIZE OBSERVER

Observe DOM elements and run callbacks whenever their content box changes





**EVEN OBSERVE
THE USER'S
DEVICE**

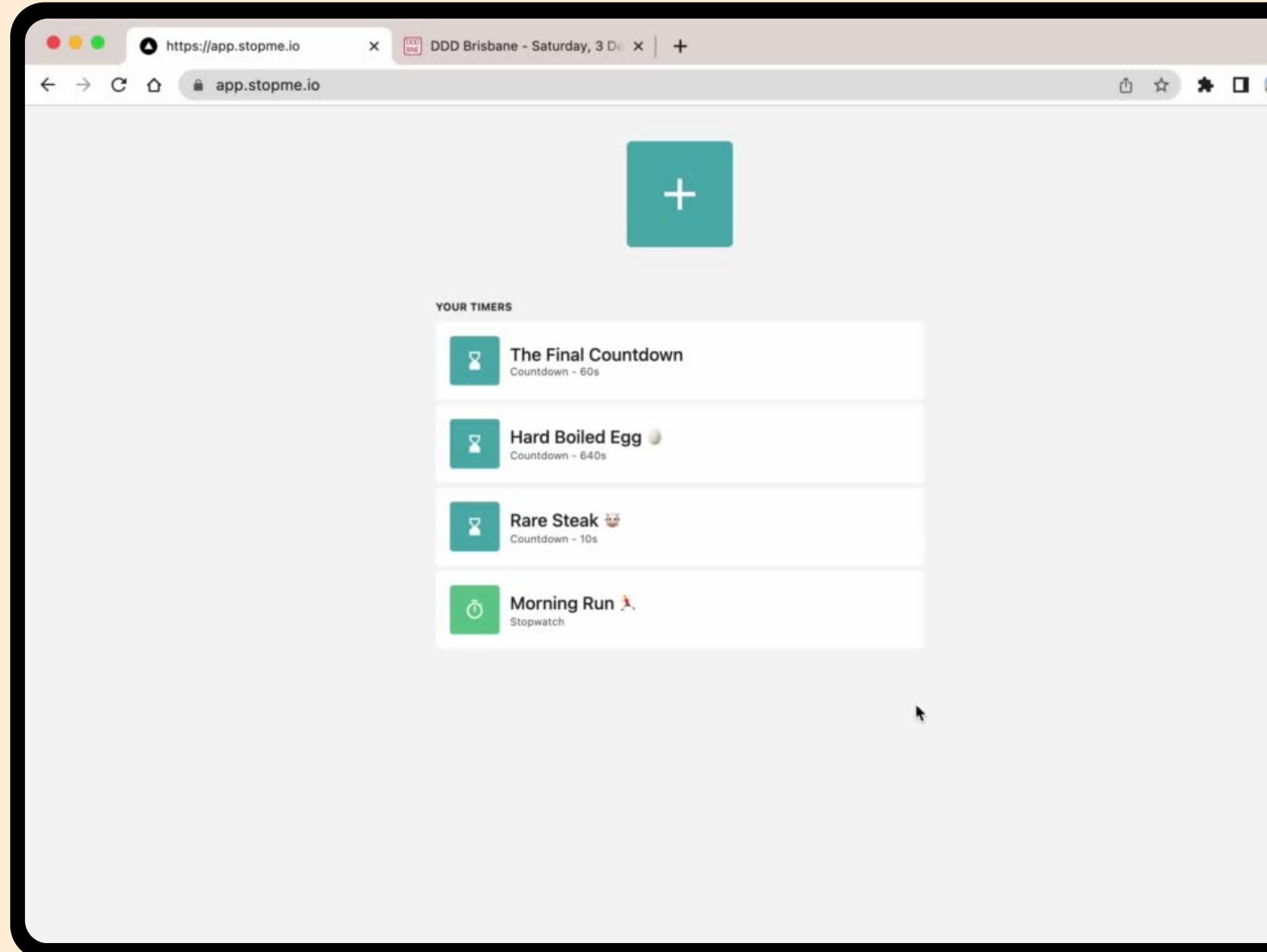
02

PAGE VISIBILITY API

Detect when the tab of the current page is active or in the background

PAGE VISIBILITY API

Detect when the tab of the current page is active or in the background



PAGE VISIBILITY API

Detect when the tab of the current page is active or in the background

```
function handleChange () {  
  if (document.visibilityState === "hidden") {  
    // do or stop something when tab is hidden  
  } else {  
    // do or stop something when it's visible  
  }  
  
  document.addEventListener(  
    "visibilitychange",  
    handleChange  
  )
```

PAGE VISIBILITY API

Detect when the tab of the current page is active or in the background



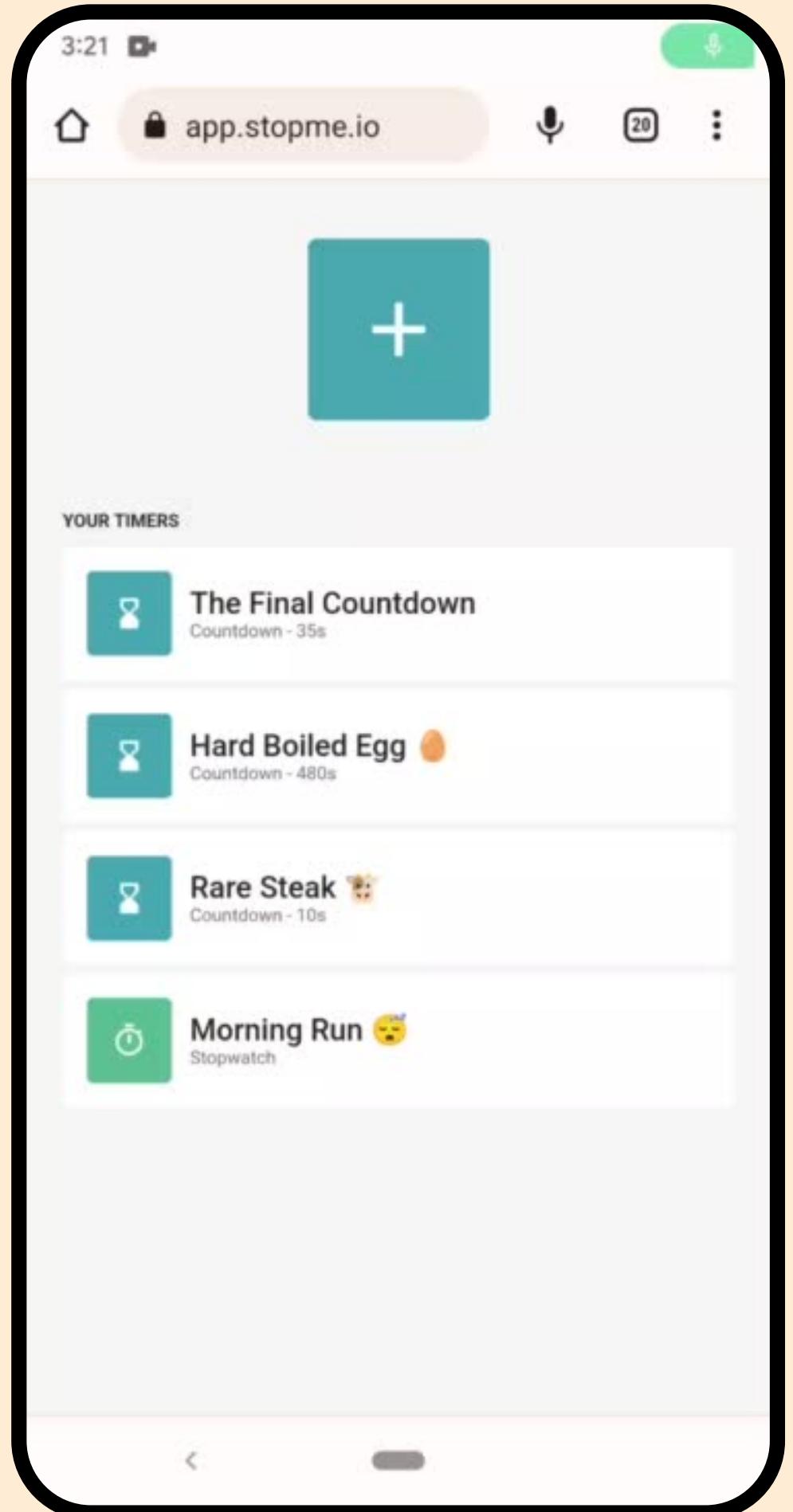
02

NETWORK INFORMATION API

Get information about the users
network connection

NETWORK INFORMATION API

Get information about the users
network connection



NETWORK INFORMATION API

Get information about the users
network connection

```
function handleChange () {  
    // navigator.connection.effectiveType  
    // *.type  
    // *.downlink / *.downlinkMax  
    // *.rtt  
    // *.saveData  
  
    if (!navigator.onLine) {  
        // user is offline  
    }  
  
    navigator.connection.addEventListener(  
        "change",  
        handleChange  
    );
```

NETWORK INFORMATION API

Get information about the users
network connection

```
function handleChange () {  
    // navigator.connection.effectiveType  
    // *.type  
    // *.downlink / *.downlinkMax  
    // *.rtt  
    // *.saveData  
  
    if (!navigator.onLine) {  
        // user is offline  
    }  
  
    navigator.connection.addEventListener(  
        "change",  
        handleChange  
    );
```

NETWORK INFORMATION API

Get information about the users
network connection

```
function handleChange () {  
    // navigator.connection.effectiveType  
    // *.type  
    // *.downlink / *.downlinkMax  
    // *.rtt  
    // *.saveData  
  
    if (!navigator.onLine) {  
        // user is offline  
    }  
  
    navigator.connection.addEventListener(  
        "change",  
        handleChange  
    );
```

NETWORK INFORMATION API

Get information about the users
network connection

```
function handleChange () {  
    // navigator.connection.effectiveType  
    // *.type  
    // *.downlink / *.downlinkMax  
    // *.rtt  
    // *.saveData  
  
    if (!navigator.onLine) {  
        // user is offline  
    }  
  
    navigator.connection.addEventListener(  
        "change",  
        handleChange  
    );
```

02

NETWORK INFORMATION API

Get information about the users
network connection



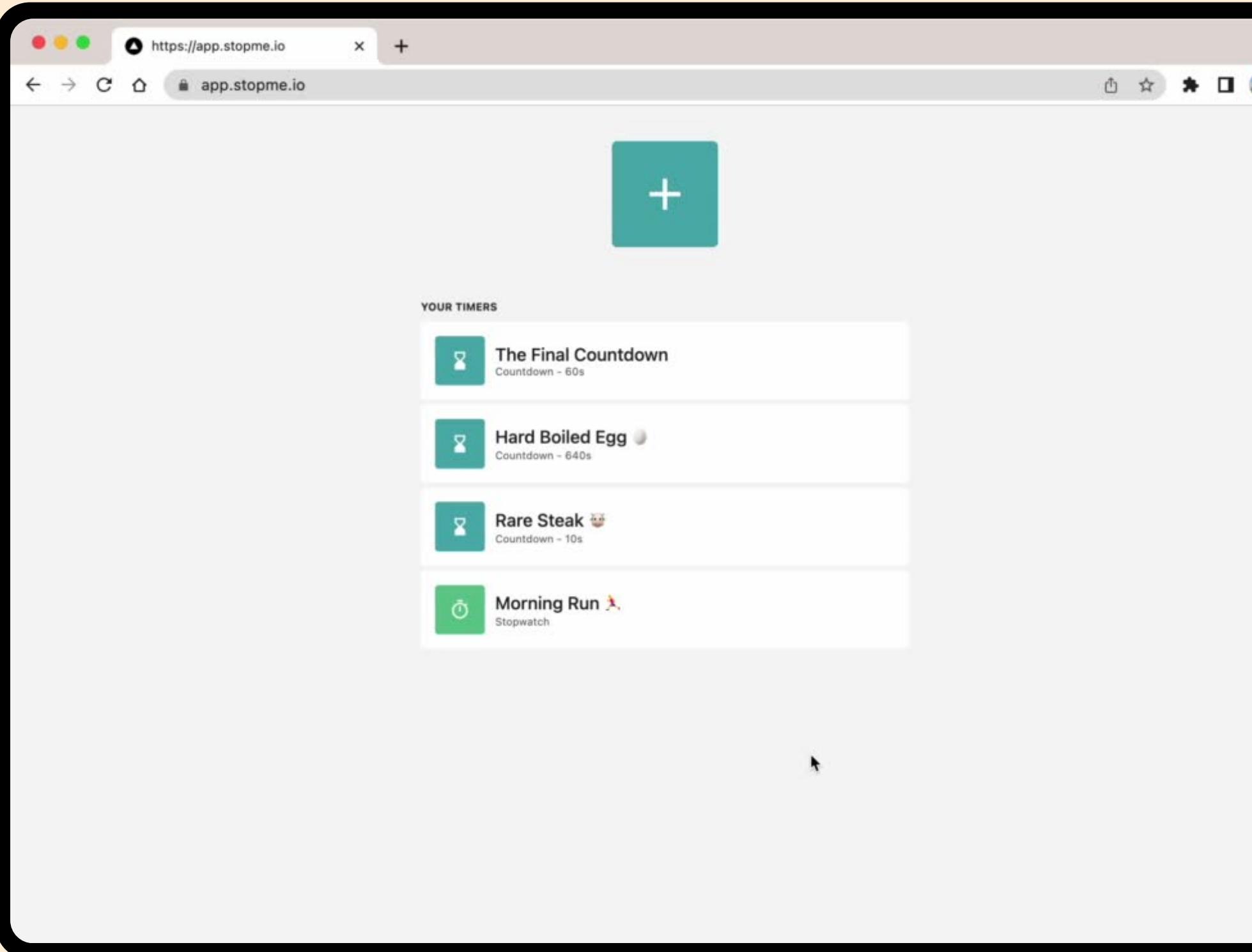
02

BATTERY STATUS API

Get details about the devices
battery status

BATTERY STATUS API

Get details about the devices
battery status



BATTERY STATUS API

Get details about the devices
battery status

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

function handleChange () {
    // battery.charging
    // *.level
    // *.chargingTime
    // *.dischargingTime
}

battery.addEventListener(
    "chargingchange"
    handleChange
)
// "levelchange"
// "chargingtimechange"
// "dischargingtimechange"
```

BATTERY STATUS API

Get details about the devices
battery status

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

function handleChange () {
    // battery.charging
    // *.level
    // *.chargingTime
    // *.dischargingTime
}

battery.addEventListener(
    "chargingchange"
    handleChange
)
// "levelchange"
// "chargingtimechange"
// "dischargingtimechange"
```

BATTERY STATUS API

Get details about the devices
battery status

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

function handleChange () {
    // battery.charging
    // *.level
    // *.chargingTime
    // *.dischargingTime
}

battery.addEventListener(
    "chargingchange"
    handleChange
)
// "levelchange"
// "chargingtimechange"
// "dischargingtimechange"
```

BATTERY STATUS API

Get details about the devices
battery status

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

function handleChange () {
    // battery.charging
    // *.level
    // *.chargingTime
    // *.dischargingTime
}

battery.addEventListener(
    "chargingchange"
    handleChange
)
// "levelchange"
// "chargingtimechange"
// "dischargingtimechange"
```

02

BATTERY STATUS API

Get details about the devices
battery status





**ENHANCE
YOUR
COMPONENTS**

053

I18N API

Internationalisation helpers and
utilities

INTL API

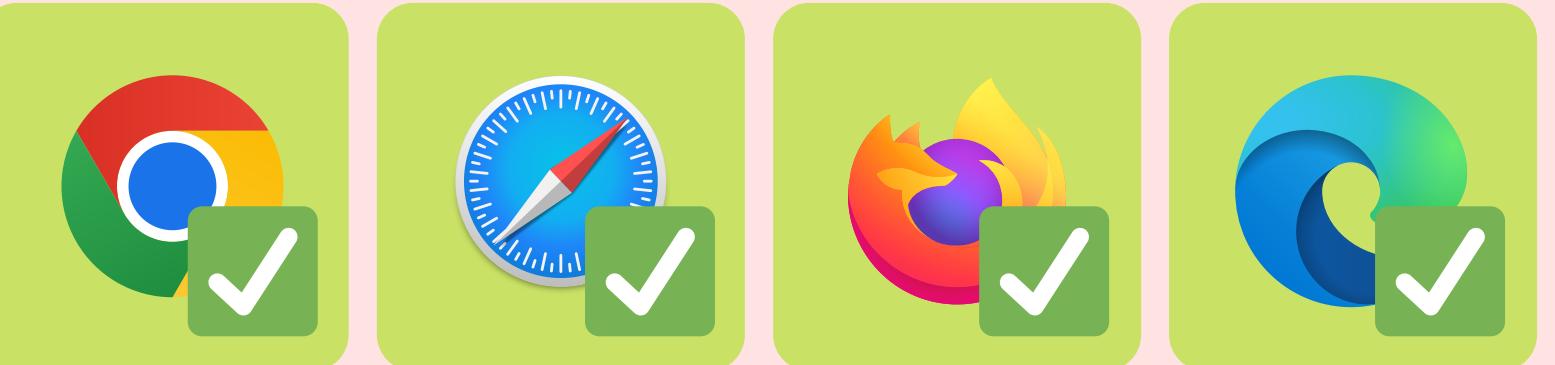
Internationalisation helpers and utilities

```
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
```

INTL

Internationalisation helpers and utilities

```
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
```



INTL API

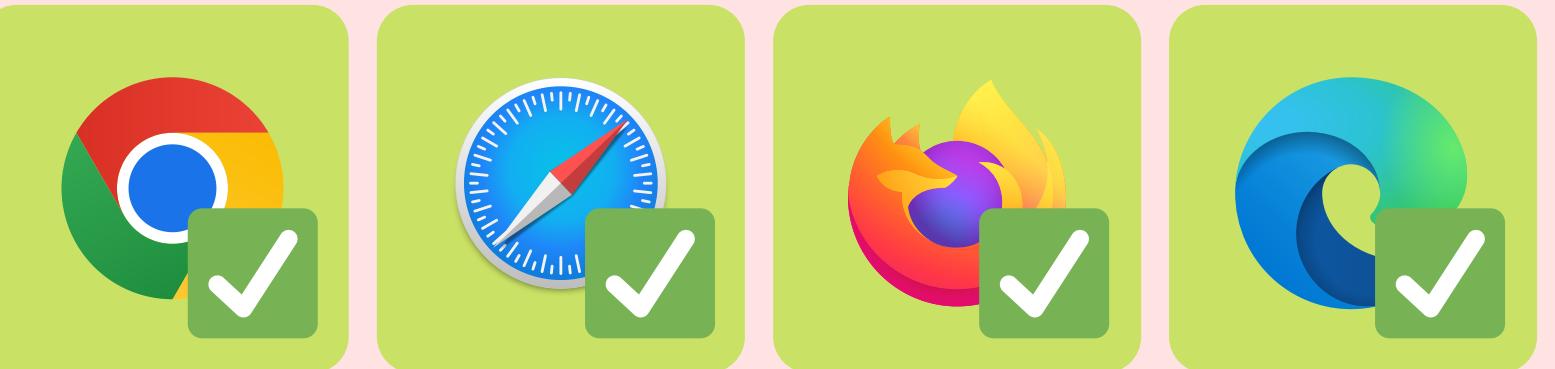
Internationalisation helpers and utilities

```
const fmt = new Intl.RelativeTimeFormat(  
  "en",  
  { style: "narrow" }  
)  
  
fmt.format(3, "day")  
// in 3 days  
  
fmt.format(-2, "year")  
// 2 years ago
```

INTL API

Internationalisation helpers and utilities

```
const formatter = new Intl.RelativeTimeFormat(  
  "en",  
  { style: "narrow" }  
)  
  
formatter.format(3, "day")  
// in 3 days  
  
formatter.format(-2, "year")  
// 2 years ago
```



INTL API

Internationalisation helpers and utilities

```
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
```

INTL API

Internationalisation helpers and utilities

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

INTL API

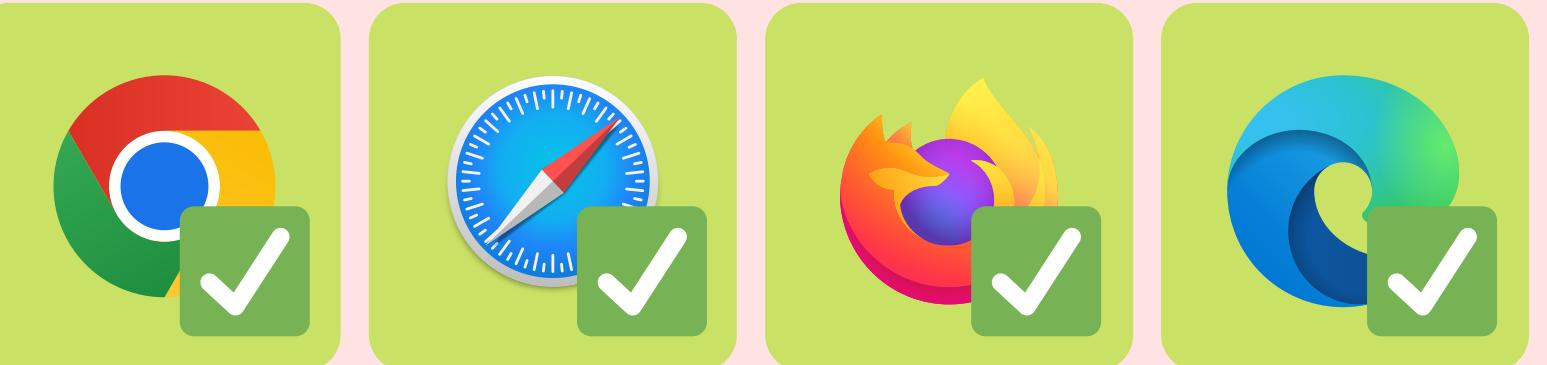
Internationalisation helpers and utilities

```
const fmt =  
  new Intl.NumberFormat("en-AU", {  
    style: "unit",  
    unit: "liter",  
    unitDisplay: "long"  
  })  
  
fmt.format(123)  
// 123 litres
```

INTL API

Internationalisation helpers and utilities

```
const fmt =  
  new Intl.NumberFormat("en-AU", {  
    style: "unit",  
    unit: "liter",  
    unitDisplay: "long"  
  })  
  
fmt.format(123)  
// 123 litres
```



053

SCREEN WAKE LOCK API

Let the users device know that you
don't want the screen to lock due to
inactivity

SCREEN WAKE LOCK API

Let the users device know that you
don't want the screen to lock due to
inactivity

```
try {  
    const wakeLock = await navigator  
        .wakeLock  
        .request("screen")  
} catch (e) {  
    // Request failed, e.g. for system  
    // related reasons like low battery  
}  
  
wakeLock.release()
```

SCREEN WAKE LOCK API

Let the users device know that you
don't want the screen to lock due to
inactivity

```
try {  
    const wakeLock = await navigator  
        .wakeLock  
        .request("screen")  
} catch (e) {  
    // Request failed, e.g. for system  
    // related reasons like low battery  
}  
  
wakeLock.release()
```

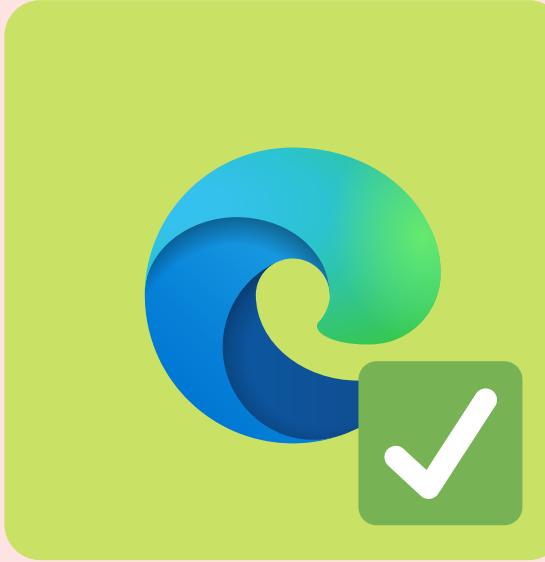
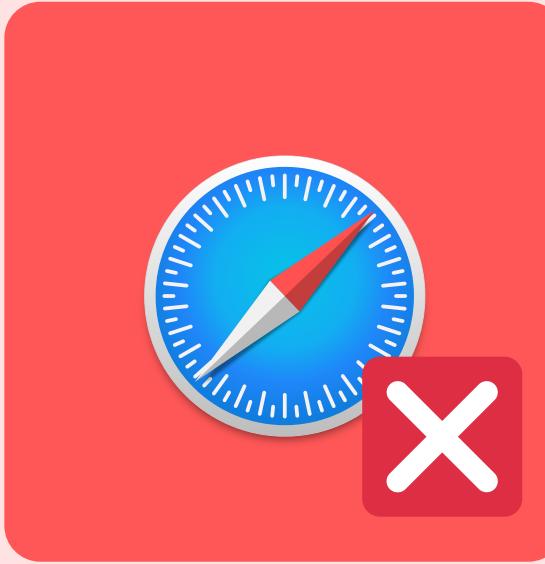
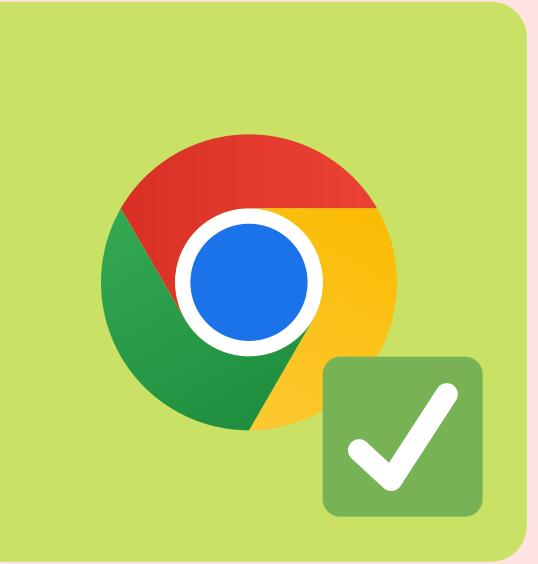
SCREEN WAKE LOCK API

Let the users device know that you
don't want the screen to lock due to
inactivity

```
try {  
    const wakeLock = await navigator  
        .wakeLock  
        .request("screen")  
} catch (e) {  
    // Request failed, e.g. for system  
    // related reasons like low battery  
}  
  
wakeLock.release()
```

SCREEN WAKE LOCK API

Let the users device know that you
don't want the screen to lock due to
inactivity



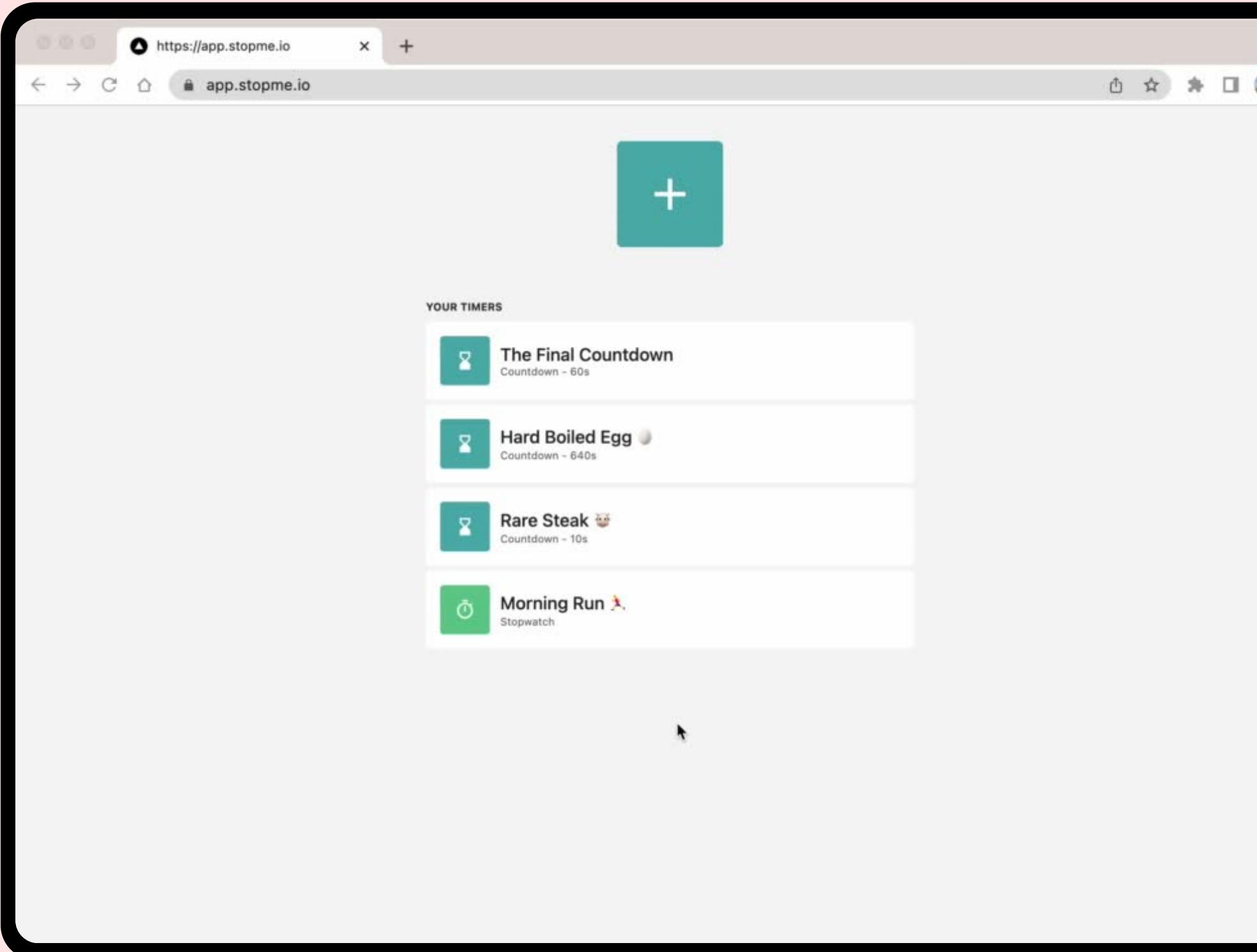
053

EYEDROPPER API

Allow your users to pick a colour
from anywhere on their screen

EYEDROPPER API

Allow your users to pick a colour
from anywhere on their screen



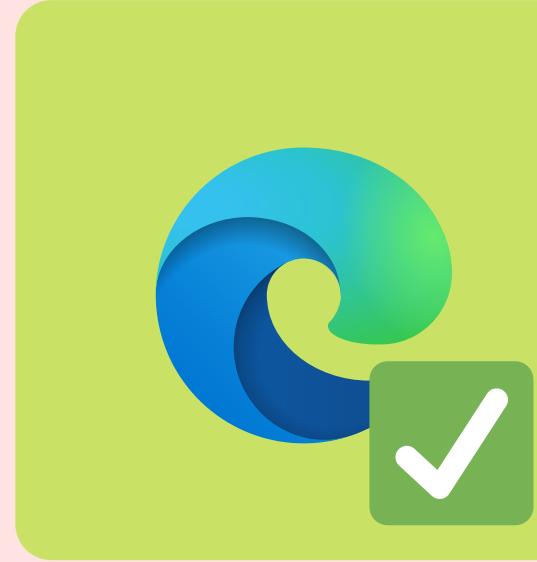
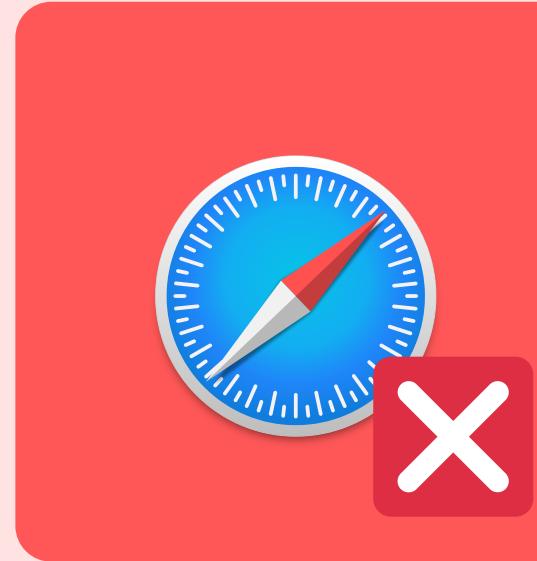
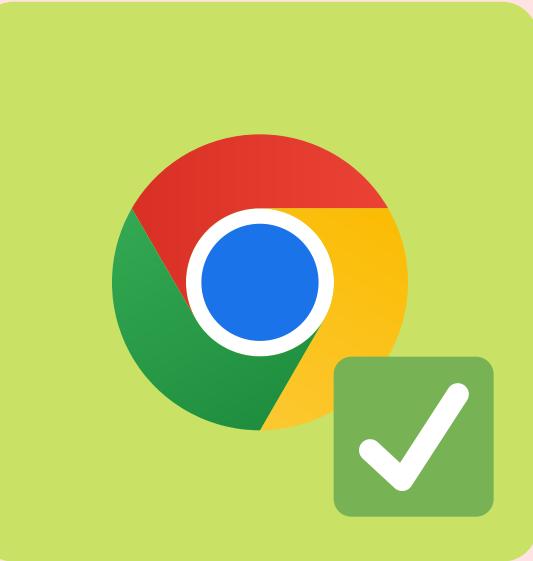
EYEDROPPER API

Allow your users to pick a colour
from anywhere on their screen

```
const eyeDropper = new EyeDropper()  
  
try {  
  const result = await eyeDropper.open()  
  // *.sRGBHex  
} catch (e) {  
  // failure, also triggered when user  
  // cancels  
}
```

EYEDROPPER API

Allow your users to pick a colour
from anywhere on their screen





**ALMOST AS
GOOD AS
NATIVE**

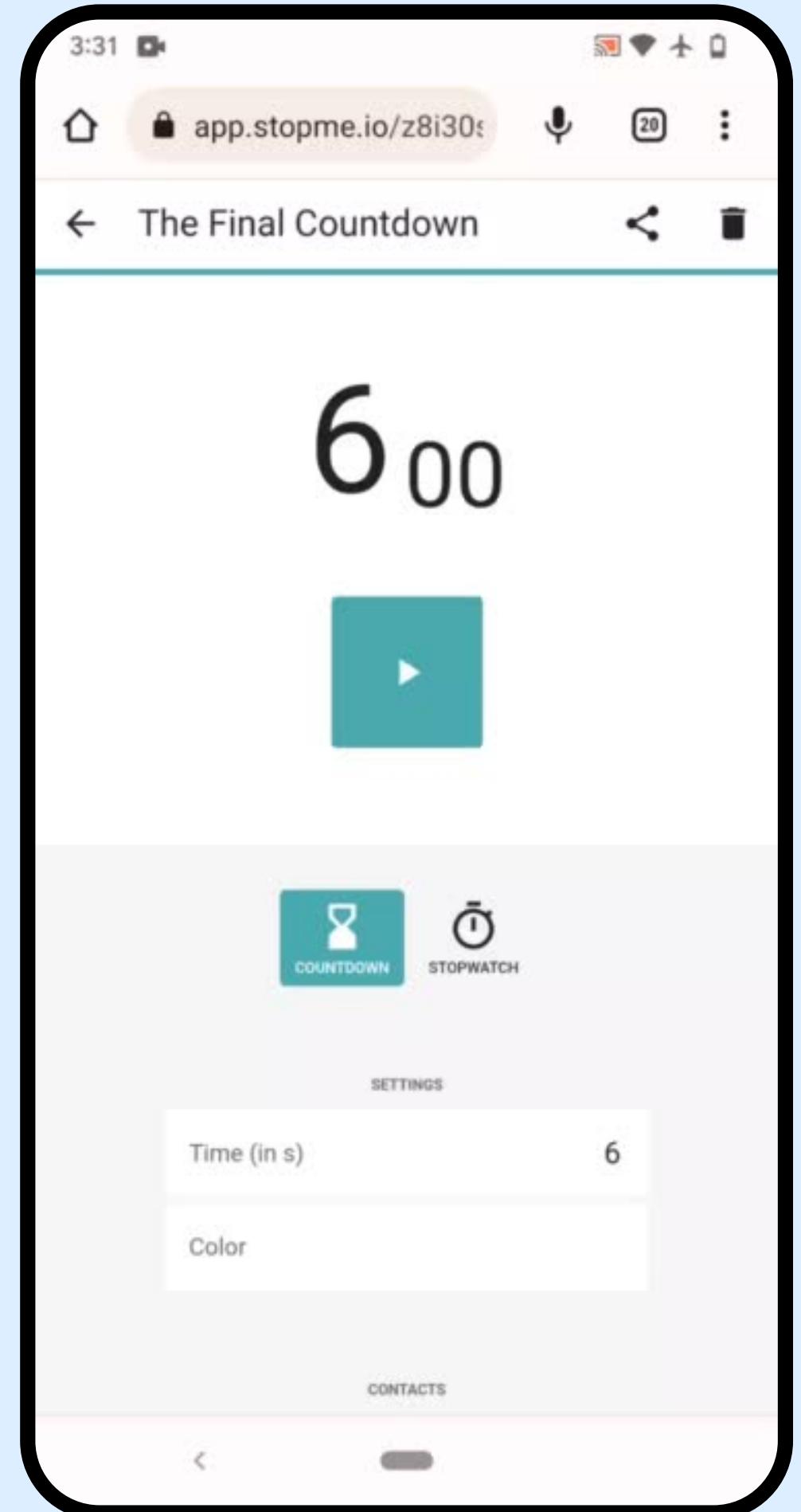
04

VIBRATION API

Control the devices vibration for
haptic feedback

VIBRATION API

Control the devices vibration for haptic feedback



VIBRATION API

Control the devices vibration for
haptic feedback

```
navigator.vibrate(200)
```

```
// Pattern
```

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

VIBRATION API

Control the devices vibration for
haptic feedback

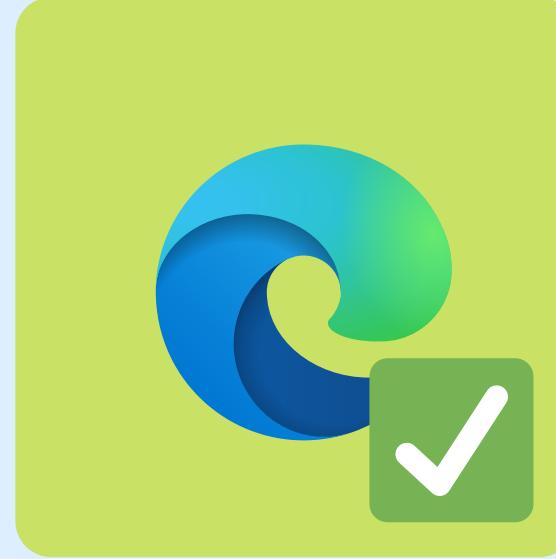
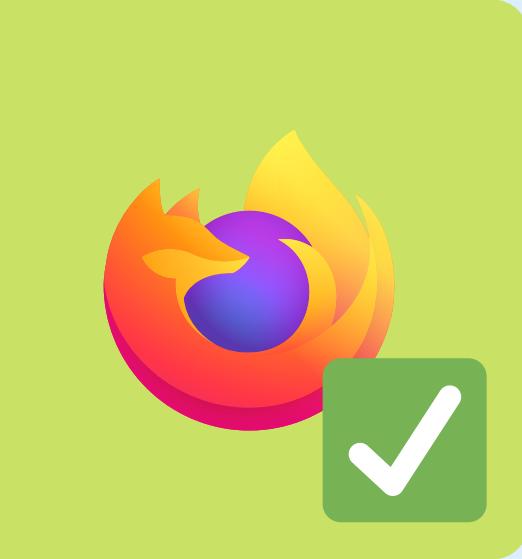
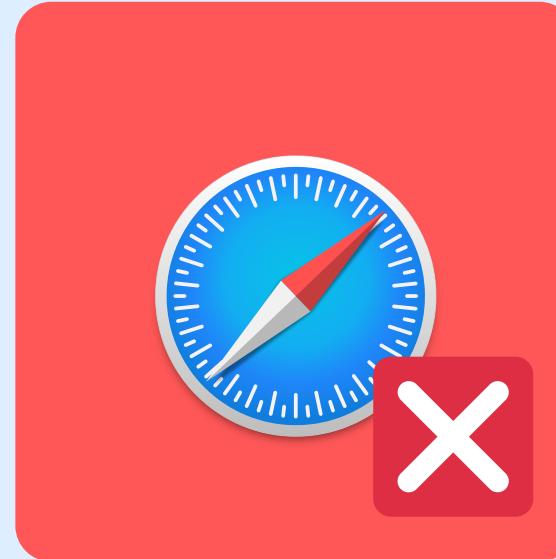
```
// https://github.com/hjdesigner/vibration-api

// Super Mario
navigator.vibrate([
  87, 89, 104, 176, 96, 176, 88, 88,
  79, 241, 176, 377, 191
])

// Game of Thrones
navigator.vibrate([
  950, 50, 530, 80, 100, 100, 100, 60,
  930, 50, 530, 80, 100, 100, 100, 60, 980
])
```

VIBRATION API

Control the devices vibration for
haptic feedback

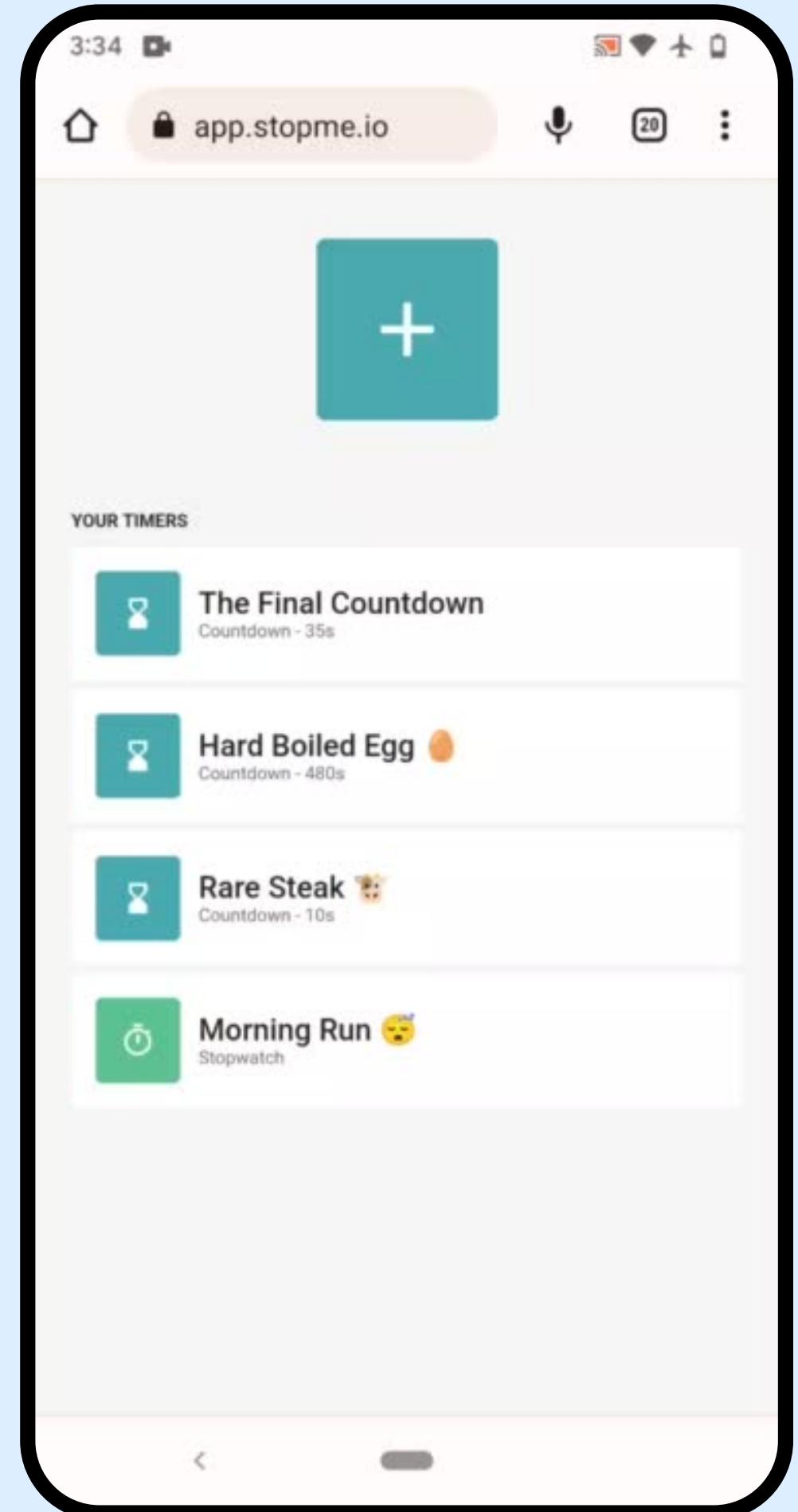


CONTACT PICKER API

Let users select from their device contacts list with the native picker

CONTACT PICKER API

Let users select from their device contacts list with the native picker



CONTACT PICKER API

Let users select from their device contacts list with the native picker

```
const supportedProperties =  
  await navigator.contacts.getProperties()  
  // ["name", "email", "tel", "address", "icon"]  
  
try {  
  const props = ["name", "email"]  
  const options = { multiple: true }  
  const contacts = await navigator  
    .contacts  
    .select(props, opts)  
} catch (e) {  
  // Failure or cancellation  
}
```

CONTACT PICKER API

Let users select from their device contacts list with the native picker

```
const supportedProperties =  
  await navigator.contacts.getProperties()  
  // ["name", "email", "tel", "address", "icon"]  
  
try {  
  const props = ["name", "email"]  
  const options = { multiple: true }  
  const contacts = await navigator  
    .contacts  
    .select(props, opts)  
} catch (e) {  
  // Failure or cancellation  
}
```

CONTACT PICKER API

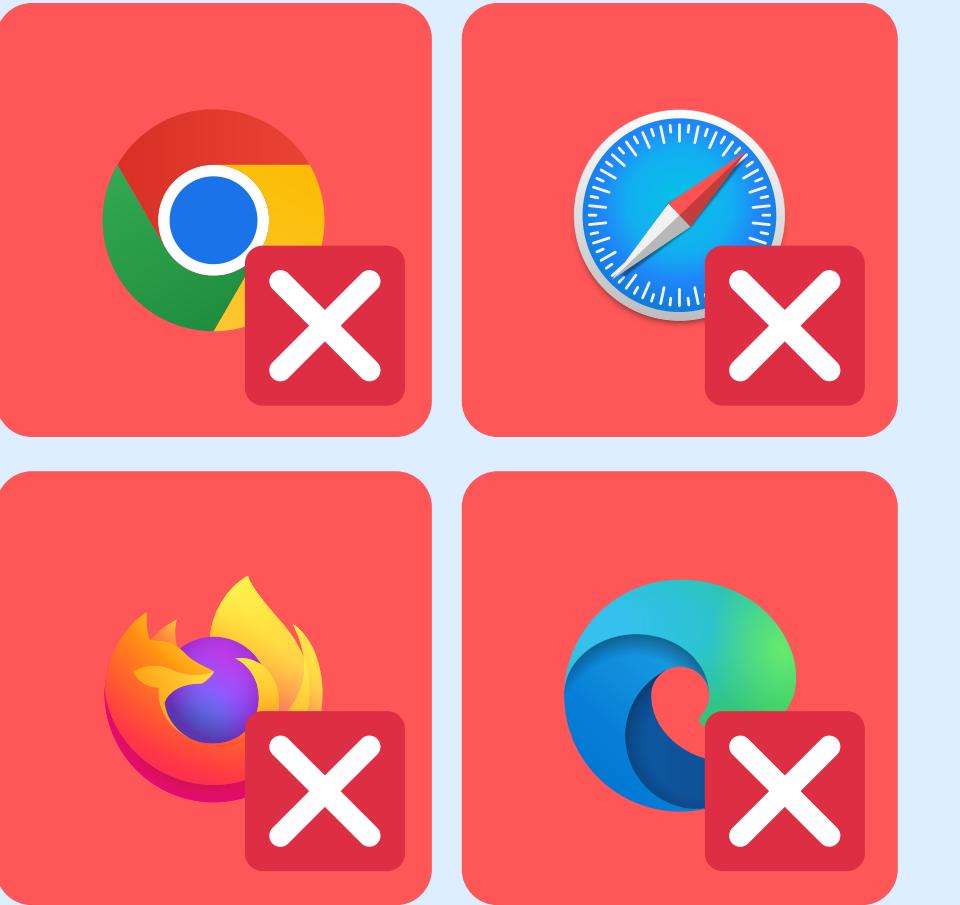
Let users select from their device contacts list with the native picker

```
const supportedProperties =  
  await navigator.contacts.getProperties()  
  // ["name", "email", "tel", "address", "icon"]  
  
try {  
  const props = ["name", "email"]  
  const options = { multiple: true }  
  const contacts = await navigator  
    .contacts  
    .select(props, opts)  
} catch (e) {  
  // Failure or cancellation  
}
```

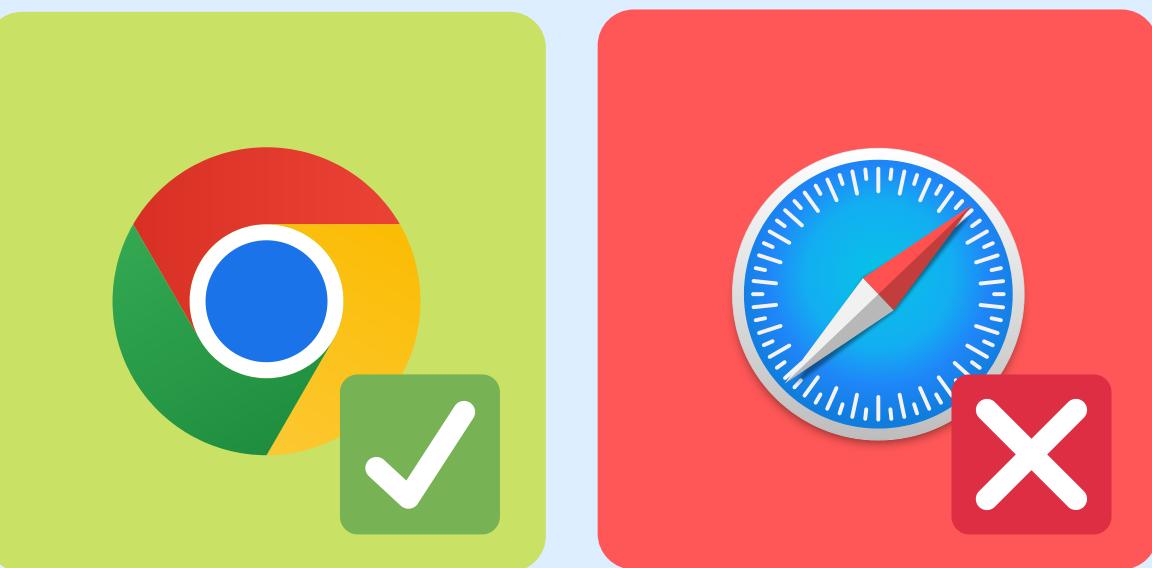
CONTACT PICKER API

Let users select from their device contacts list with the native picker

DESKTOP



MOBILE

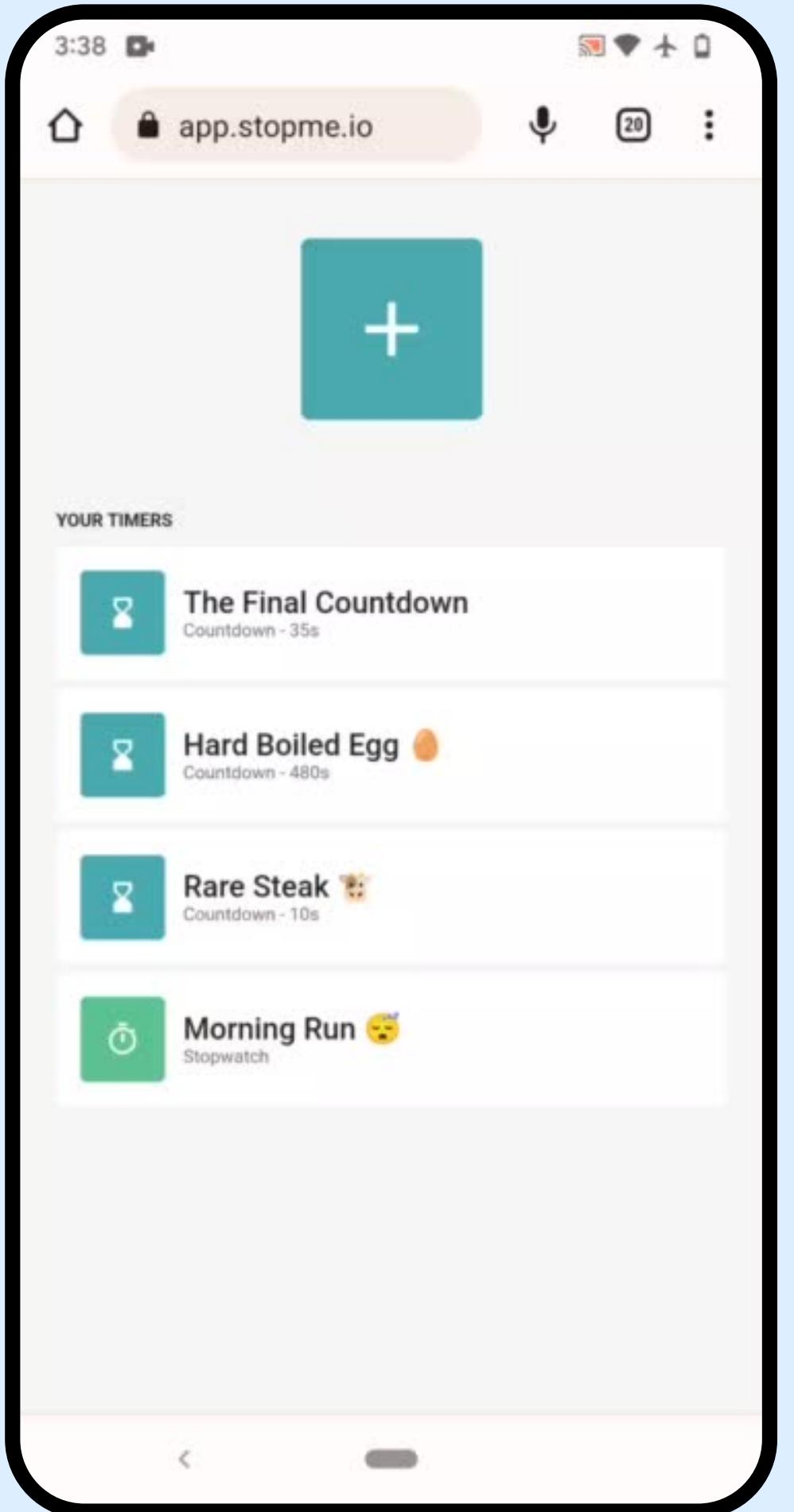


WEB SHARE API

Allow users to share content
through any of their installed apps
using the native picker

WEB SHARE API

Allow users to share content through any of their installed apps using the native picker



WEB SHARE API

Allow users to share content
through any of their installed apps
using the native picker

```
const shareData = {  
  title: "stopme.io",  
  text: "The best SaaS product in the world",  
  url: "https://stopme.io"  
}  
  
// Must be triggered by user interaction  
btn.addEventListener("click", async () => {  
  try {  
    await navigator.share(shareData);  
  } catch (e) {  
    // failed or cancelled  
  }  
});
```

WEB SHARE API

Allow users to share content through any of their installed apps using the native picker

```
const shareData = {  
  title: "stopme.io",  
  text: "The best SaaS product in the world",  
  url: "https://stopme.io"  
}  
  
// Must be triggered by user interaction  
btn.addEventListener("click", async () => {  
  try {  
    await navigator.share(shareData);  
  } catch (e) {  
    // failed or cancelled  
  }  
});
```

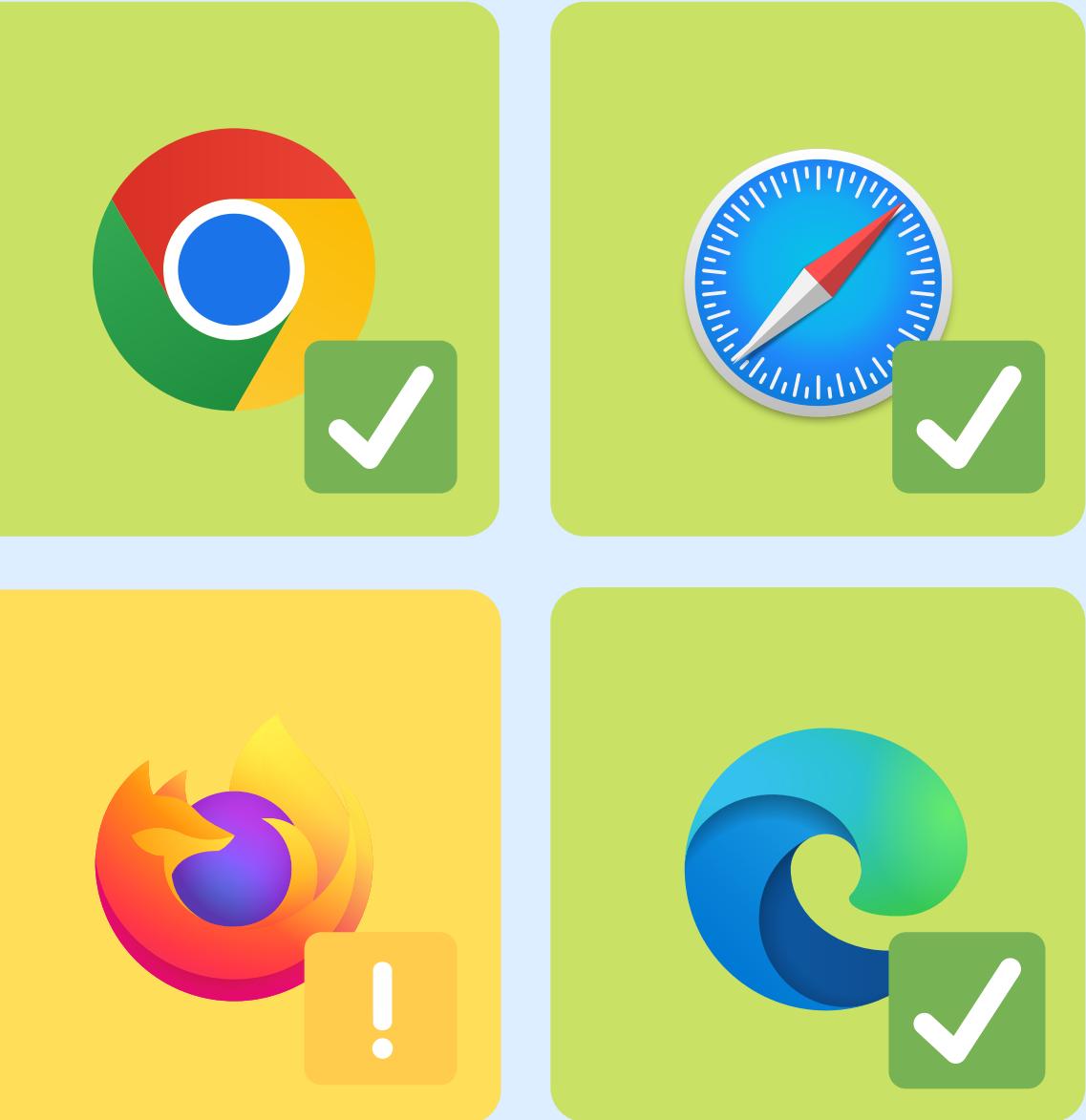
WEB SHARE API

Allow users to share content through any of their installed apps using the native picker

```
const shareData = {  
  title: "stopme.io",  
  text: "The best SaaS product in the world",  
  url: "https://stopme.io"  
}  
  
// Must be triggered by user interaction  
btn.addEventListener("click", async () => {  
  try {  
    await navigator.share(shareData);  
  } catch (e) {  
    // failed or cancelled  
  }  
});
```

WEB SHARE API

Allow users to share content
through any of their installed apps
using the native picker



HONORABLE MENTIONS

HONORABLE MENTIONS

```
// Bluetooth API
const devices =
  await navigator.bluetooth.getDevices()

const device = await navigator.bluetooth
  .requestDevice({
    filters: [{ services: [A, B] }]
  })

// USB API
const devices =
  await navigator.usb.getDevices()

const device = await navigator.usb.
  .requestDevice({
    filters: [{ vendorId: id }]
  })
```

HONORABLE MENTIONS

```
// Geolocation API
// Old but gold
navigator.geolocation
  .getCurrentPosition(pos => {
    // pos.coords.latitude
    // *.coords.longitude
  })

const watchId = navigator.geolocation
  .watchPosition(pos => {
    // ...
  })

navigator.geolocation.clearWatch(watchId)
```

HONORABLE MENTIONS

```
// File System Access API
const options = { types: [ A, B ] }
const [handle] =
    await window.showOpenFilePicker(options)
const fileData = await handle.getFile()

// Handling directories
const dirHandle =
    await window.showDirectoryPicker()

// Saving files
const options = {
    types: [
        {
            description: "Text file",
            accept: { "text/plain": [ ".txt" ] },
        }
    ]
}
await window.showSaveFilePicker(options)
```

HONORABLE MENTIONS

```
// Clipboard API
await navigator.clipboard.writeText(txt)

const text =
  await navigator.clipboard.readText();
```

HONORABLE MENTIONS

```
// Presentation API
const urls = [ presentationUrl, altUrl ]
const request = new PresentationRequest(urls)

const availability =
  await request.getAvailability()

availability.onchange = () => {
  // availability.value
}

const connection = await request.start()
connection.close()
```

HONORABLE MENTIONS

```
// Web Speech API

// Recognition
const recognition = new SpeechRecognition()
recognition.start()

recognition.onresult = (event) => {
  const word = event.results[0][0].transcript
  // event.results[0][0].confidence
}

// Synthesis
const synth = window.speechSynthesis
const words =
  new SpeechSynthesisUtterance(txt)
synth.speak(words)
```

HONORABLE MENTIONS

```
// And so many more...  
  
// Check out:  
// https://developer.mozilla.org/en-US/docs/Web/API
```

THANKS!

@JBURR90 / JULIAN BURR
DDD BRISBANE

