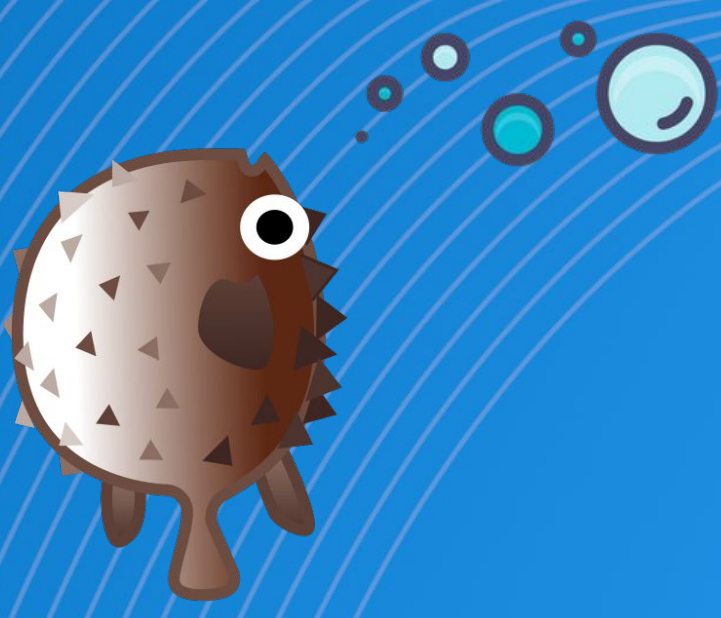


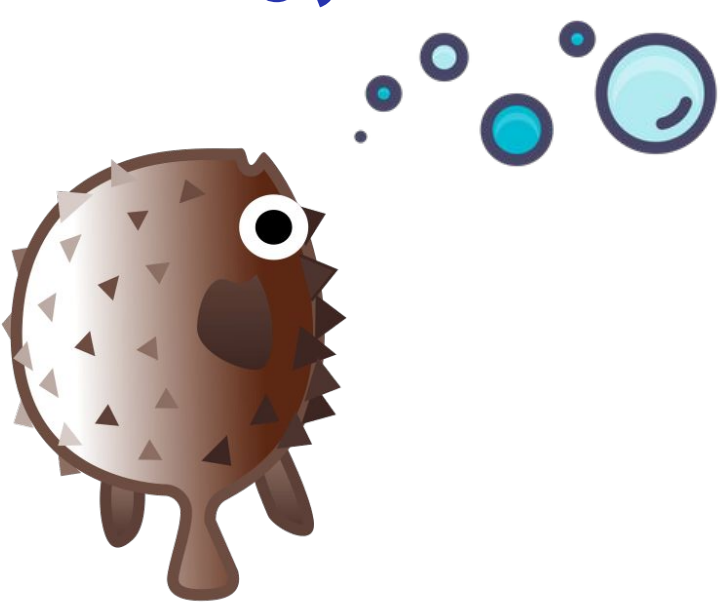
GOTO AARHUS 2021

#GOTOaar



Project Fugu

The first two years

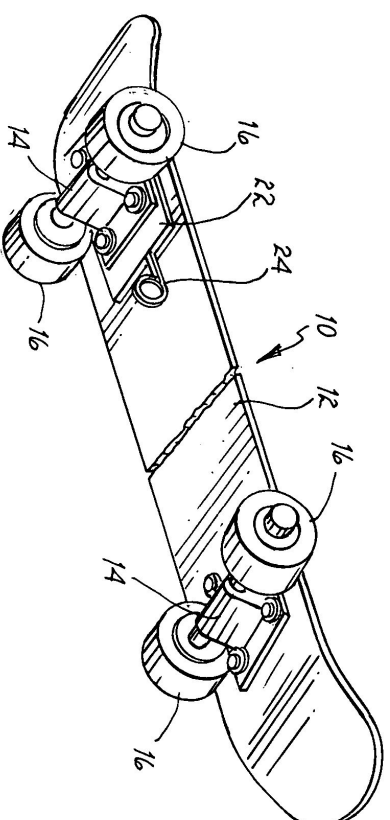


Kenneth Christiansen
Web Platform Architect, Intel



Hello there!

I am Kenneth, your
presenter today!



THE W3C TECHNICAL ARCHITECTURE GROUP (TAG)

Introduction to the TAG

The TAG is a special working group within the W3C, chartered (under the [W3C Process Document](#)) with stewardship of the Web architecture.

As outlined in its [charter](#), there are three aspects to this mission:

- to document and build consensus around principles of Web architecture and to interpret and clarify these principles when necessary;
- to resolve issues involving general Web architecture brought to the TAG;
- to help coordinate cross-technology architecture developments inside and outside W3C.

See the [TAG current work page](#) for details of how to work with it, upcoming events, and its publications.

Who are the TAG?

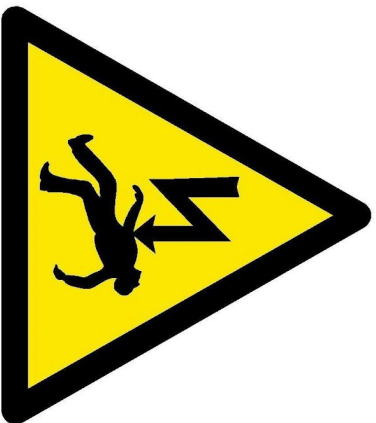


I ♥ web

The world's only true democratized platform

- 👁️ Accessible to everyone, world wide reach
- 👁️ Based on standards - IETF, W3C, WHATWG etc
- 👁️ Works on all major architectures, OS's and form factors
- 👁️ Flexible for apps, books, news, games etc
- 👁️ Thriving open source ecosystem around it

But we cannot
take it for granted



Danger
Of death

😐 Not doing all that great on mobile

😐 Desktop will only stay relevant if there are “apps” available

Some people on mobile seldom browse
the web, “there is an app for that”

Where are my users today
and tomorrow?!

Chrome OS

Windows

Win32

UWP

Developer focus in mostly on mobile
today, and native desktop is hard!

macOS / iPad

Web / Cloud??

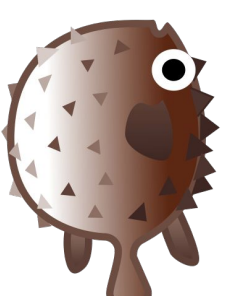
Electron? Flutter?
Cross platform?

\$\$\$\$\$



blogs.windows.com/windowsexperience/2019/11/04/introducing-the-new-microsoft-edge-and-bing/

In our mission to empower people with knowledge with Microsoft 365, one critical opportunity is the internet. We know that 60% of the time people spend on the PC is within the web browser, and it has become the primary way we work, learn and play. The internet has become an everyday utility, that we reach for automatically without thinking, yet two decades after its introduction, we are bumping into challenges and obstacles that raise questions for us.



Project Fugu

A little step back

How did we get to this?

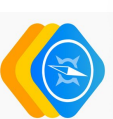
PWA



This is my story



@kennethrohde



WebKit
Open Source Web Browser Engine



Great, but short lived success


@kennethrohde

Headed up a WebKit team in Brazil
Our demo browser became the
official browser product for the N9

7.9

VERGE SCORE

NOKIA N9



GOOD STUFF

- Utterly gorgeous
- Innovative, fresh UI
- Quick camera operation

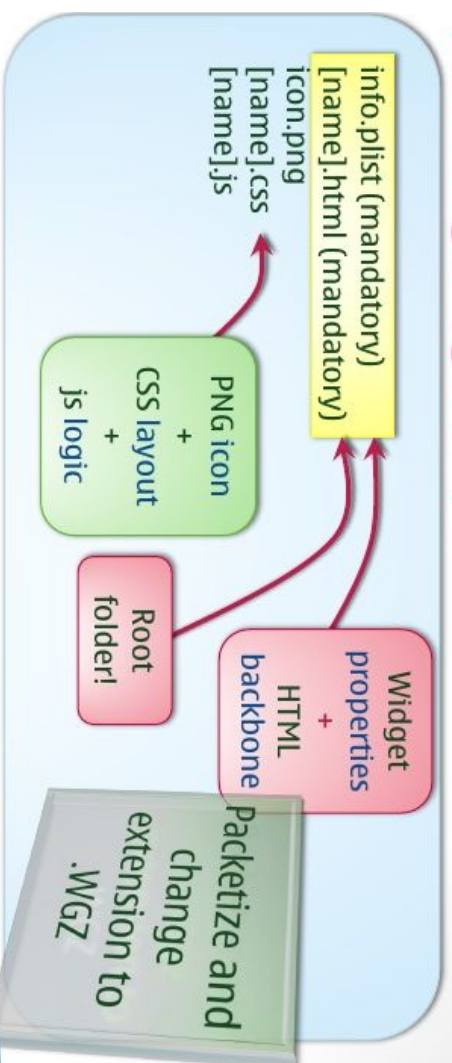
BAD STUFF

- Doomed ecosystem
- Old processor
- Unstable software

Nokia wanted apps by leveraging web tech, Nokia Web Runtime

18.06.2010 © 2010 Nokia 7

WRT Widget Ingredients



ForumNokia

18.06.2010 © 2010 Nokia 3

The Developer Offering – Simplification

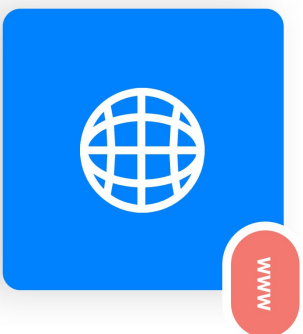
Via Cross platform tools, Ovi Services and strong ecosystem partners
Innovation (and transparency) through open source



ForumNokia

It was terrible! 😞 Not really web and lots of low quality APIs

@kennethrohde



safe

Common web properties

Low friction

ephemeral

(deep) linkable

indexable

composable

My N9 team wanted ***installable web apps*** and had basic support, but browser was short lived

I joined Intel and they were replicating Nokia Web Runtime with Samsung - The Tizen Project



Found like-minded people in W3C SysApps Working Group

Web App Manifest

W3C Working Draft 22 January 2021

This version:

<https://www.w3.org/TR/2021/WD-appmanifest-20210122/>

Latest published version:

<https://www.w3.org/TR/appmanifest/>

Latest editor's draft:

<https://w3c.github.io/manifest/>

Previous version:

<https://www.w3.org/TR/2021/WD-appmanifest-20210119/>

Editors:

Marcos Cáceres (W3C Invited Expert)

Kenneth Rohde Christiansen (Intel Corporation)

Mounir Lamouri (Google Inc.)

Anssi Kostainen (Intel Corporation)

Matt Giuca (Google Inc.)

Aaron Gustafson (Microsoft Corporation)

@kennethrohde



Manifest for app
meta data

~~NavigationController~~
Service Workers

~~Runtime spec~~

Became a team effort

Marcos C. (Mozilla), Paul Kinlan (Google), Alex Russell (G), Jeff Burtoft (Microsoft) etc

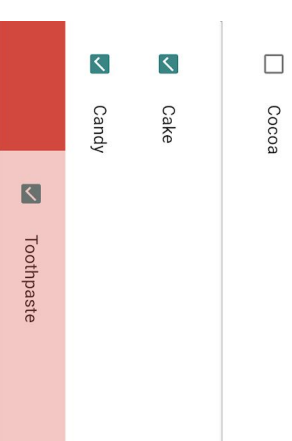
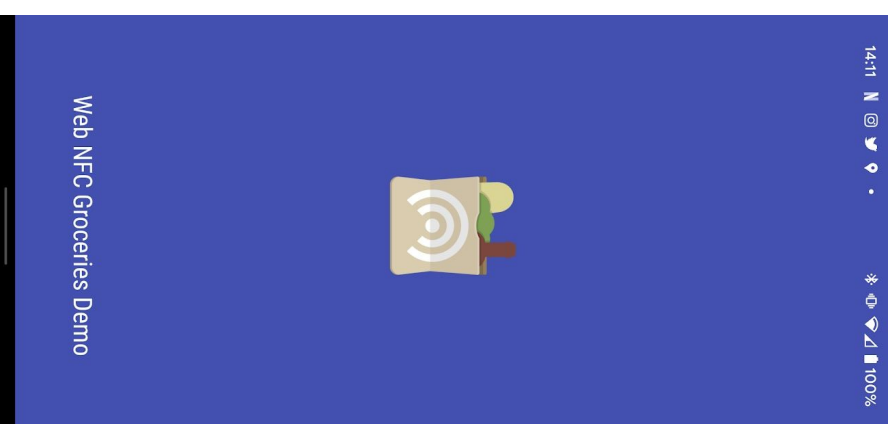
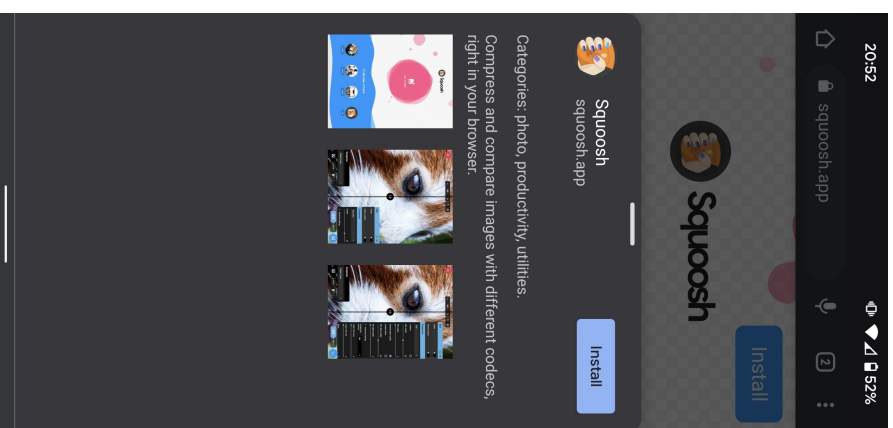
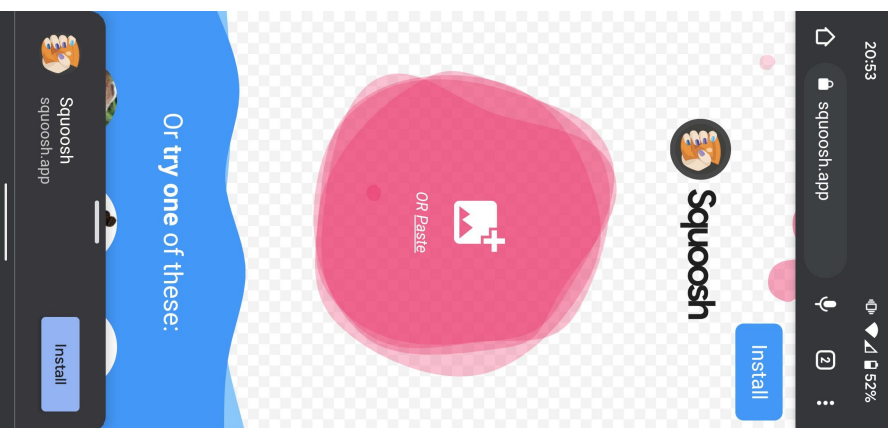


Manifest for app
meta data

Service Workers

Add to homescreen

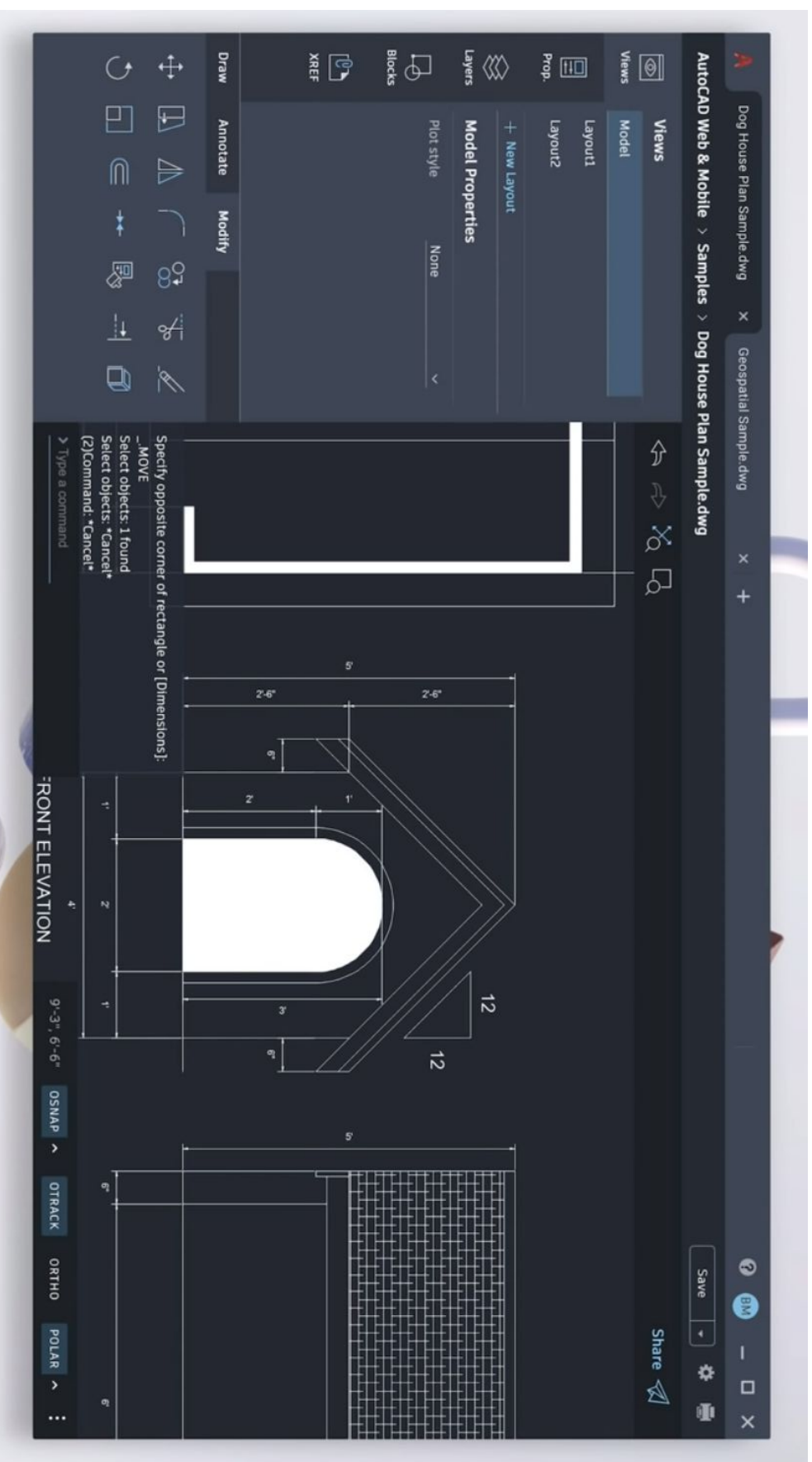
Progressive Web Apps



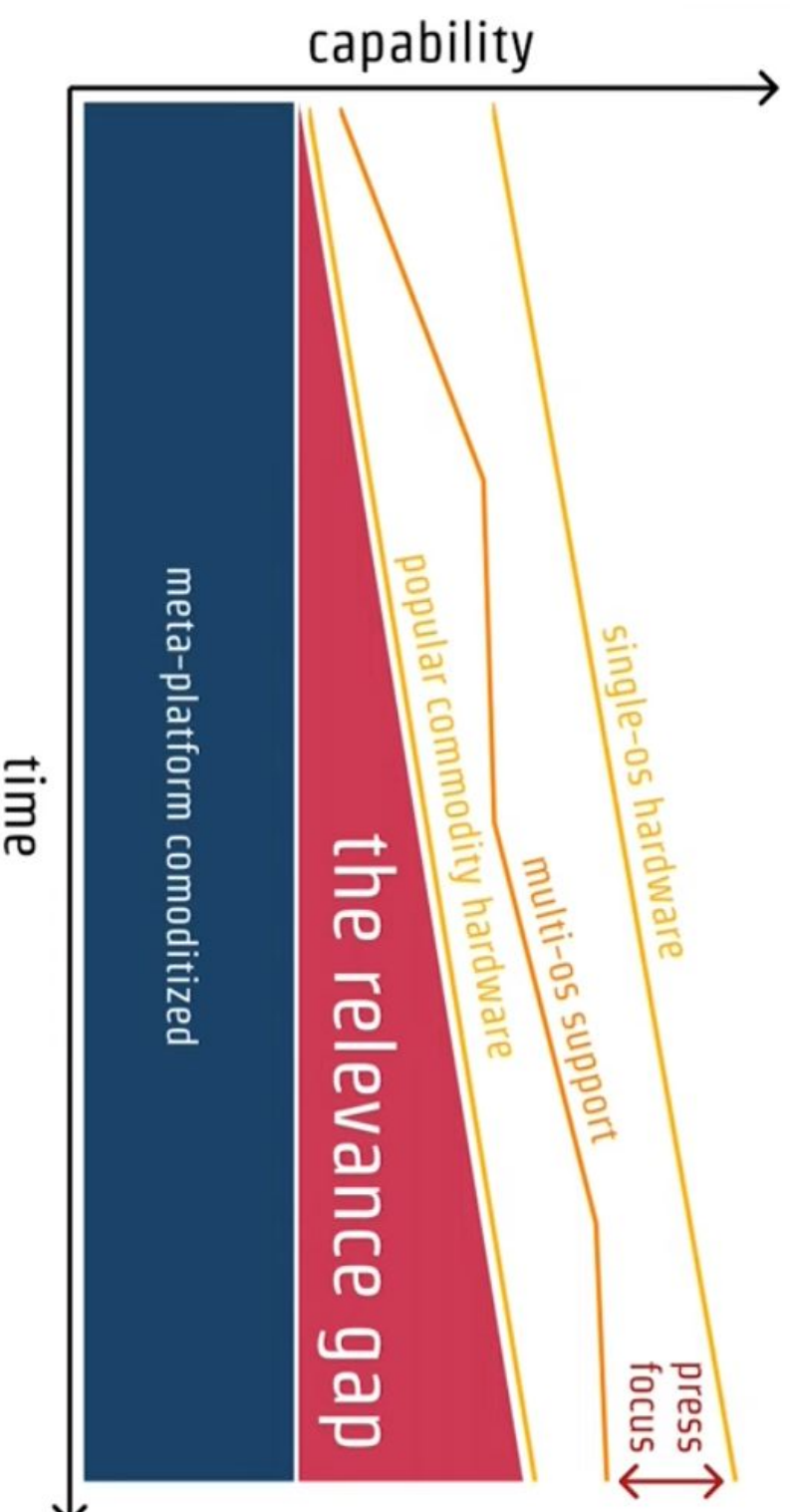
@kennethrohde

PWA

Progressive Web Apps

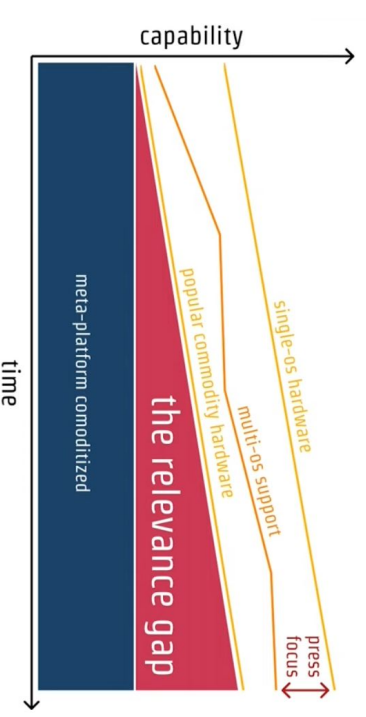


The relevance gap by Alex Russell



The relevance gap by Alex Russell

- When people decide what tech to bet on, it depends on
 - Whether the platform supports the needed capabilities
 - Whether they believe the platform will keep up (projects are multi-year)
- Devices get more features per year

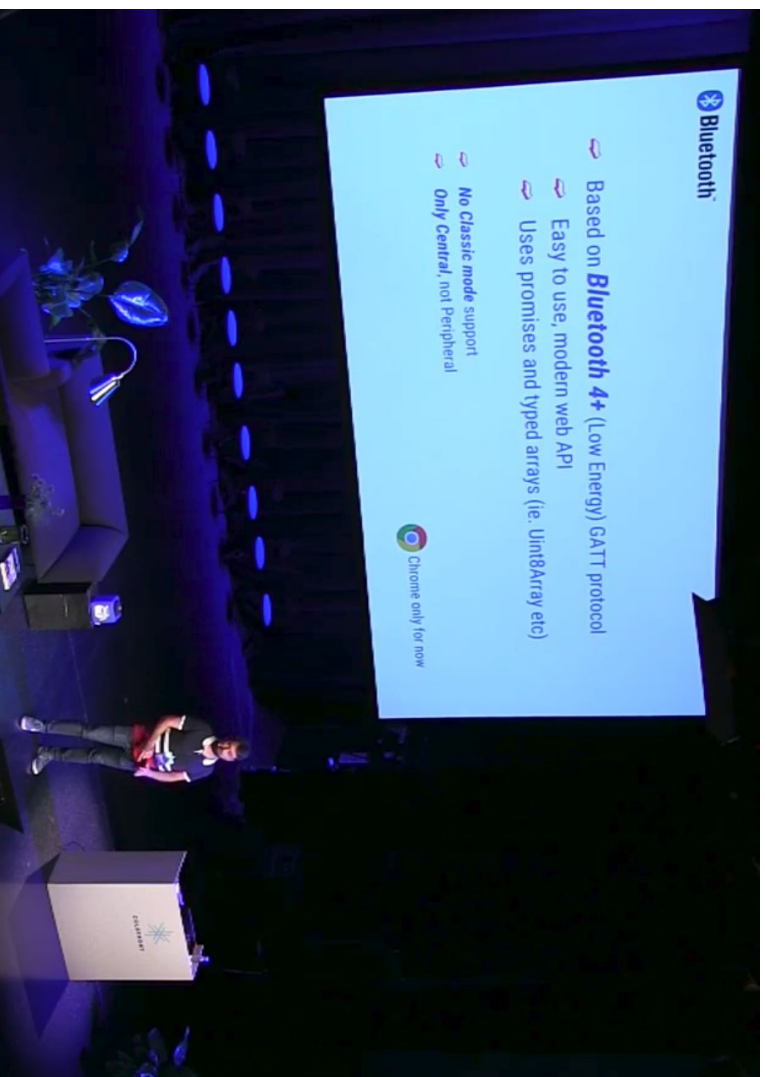


Summary: Betting on web might not be feasible

- 😞 Lack of API, making the core experience not possible
- 😞 Fear of future lacking API, giving competitor an advantage
- 😞 Lack of API roadmap

→ distrust in the platform

Chrome used to get a few “one-off” APIs like,

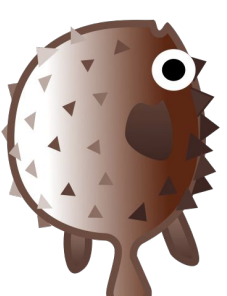


youtube.com/watch?v=qEVIhQB07QY

@kennethrohde

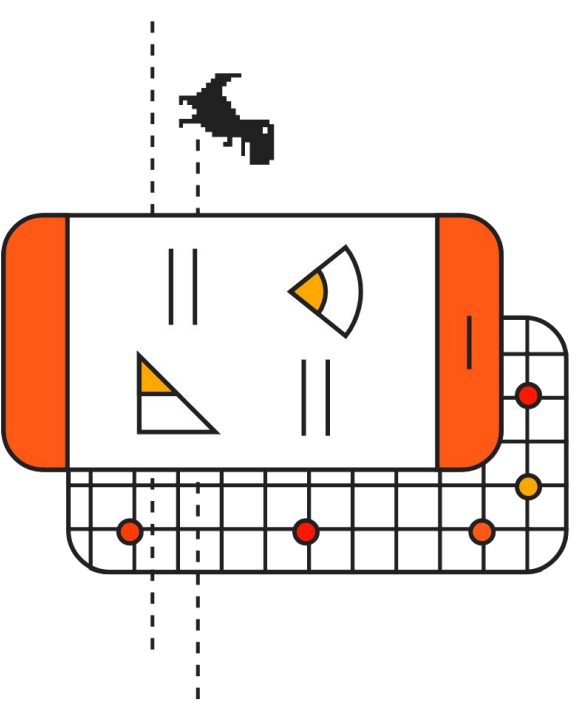
Web USB and Web Bluetooth

Watch my earlier talk
on how to use Web Bluetooth and
Web USB

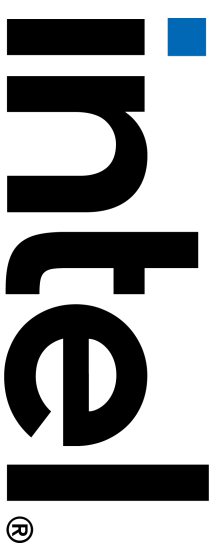


Project Fugu

Close the gap and regain trust in the web







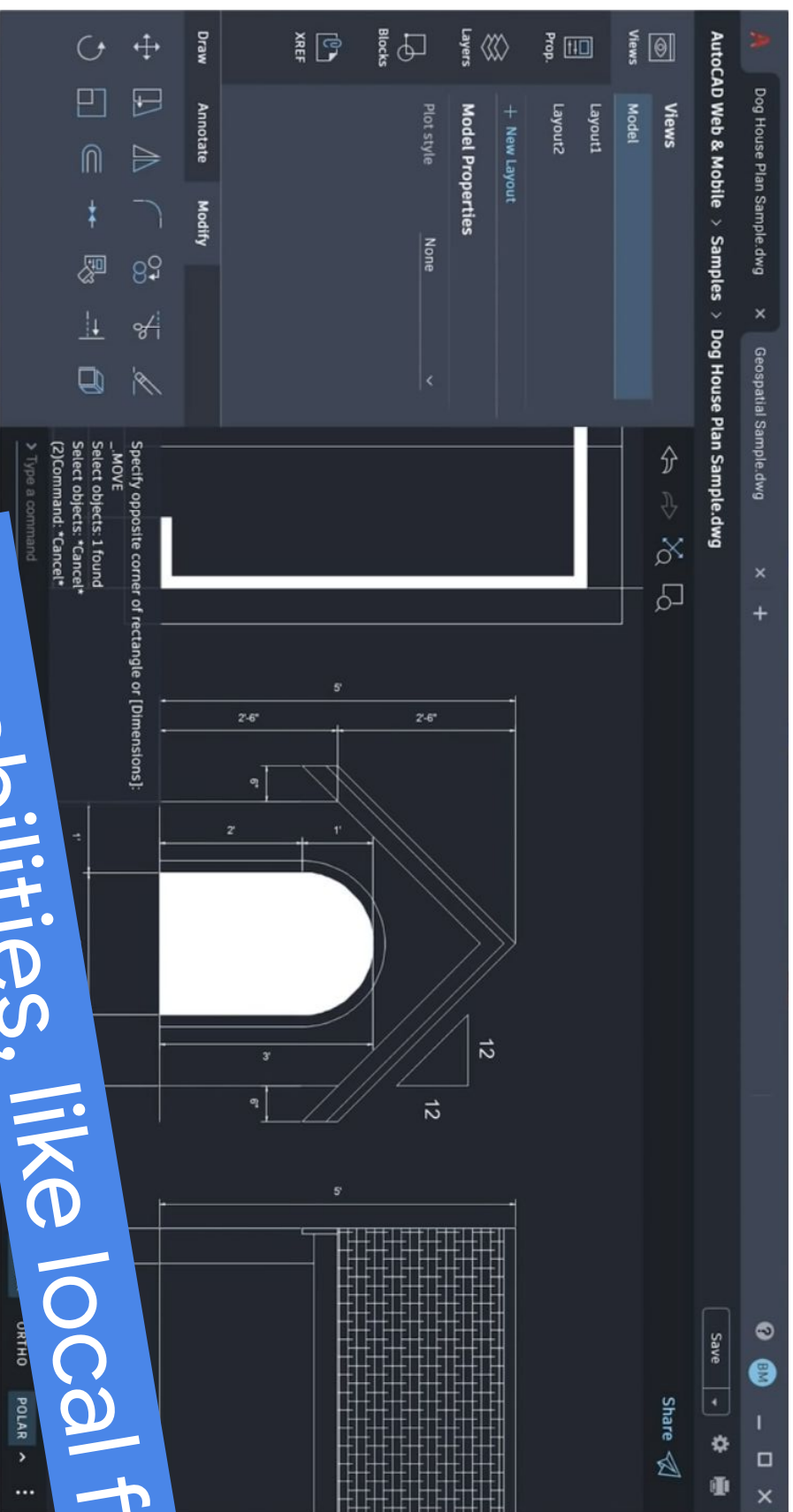
Microsoft



Delicious if prepared correctly, **deadly** if not

Enable web apps to do anything native apps can, by exposing the **capabilities of native platforms** to the web platform, while maintaining user security, privacy, trust, and other core tenets of the web.

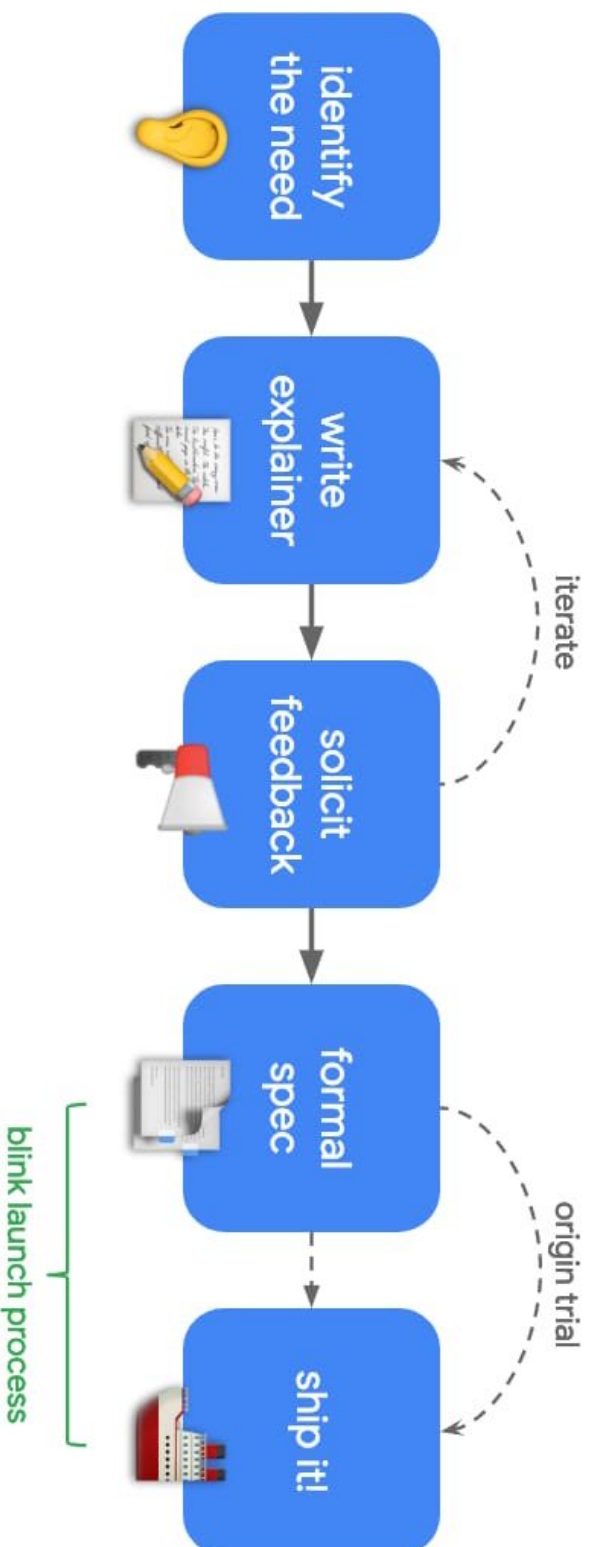
Enable web apps to do anything native apps can, by exposing the capabilities of native platforms to the web platform, while maintaining **user security, privacy, trust**, and other **core tenets of the web**.



@kennethrohde

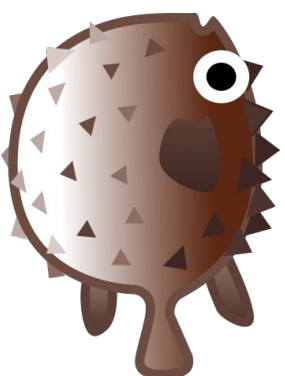
Faster process

ROADMAP!
More resources





It's been **Two Years**



Fugu API Tracker

<https://fugu-tracker.web.app>



Chrome 88
13 days ago
(Jan 19, 2021)

Chrome 89
In 29 days
(Mar 2, 2021)

Chrome 90
In 71 days
(Apr 13, 2021)

Shipped

Web Share Target	M71		+
Web Share API Level 2	M75		+
Web Share Target Level 2	M76		+
Async Clipboard: Read and Write Images	M76		+
Enter Key Hint	M77		+
Expand Storage Quota	M78		+
Periodic Background Sync	M80		+
PWA desktop-pwas: Support "minimal-ui" display mode	M80		+
Compression codecs	M80		+
Get Installed Related Apps API	M80		+
Contacts API	M80		+
PWA Allow the Badging API to be used from a service worker via Push	M81		+
PWA Badging API	M81		+
Barcode Detection API	M83		+





<https://fugu-tracker.web.app>

The [capabilities project](#), also known as Project Fugu, is a cross-company effort to make it possible for web apps to do anything iOS, Android, or desktop apps can, by exposing the capabilities of these platforms to the web while maintaining user security, privacy, trust, and other core tenets of the web.

If you'd like to work with this data programmatically, you can [download it](#) (approx. 96KB)

This view shows all of the APIs being considered. To see just the APIs that are available to test with or use in the order they became available, swap to our [timeline view](#). A large screen is recommended.

Icons

Icon	Meaning
	Shipped, or expecting to ship, in Chrome
	In origin trial
	In developer trial
	More info

Availability

Icon	Meaning
	Android
	Chrome OS
	Linux
	Mac
	PWA
	Windows



Fugu API Timeline <https://fugu-tracker.web.app>

Chrome 75 608 days ago (Jun 4, 2019)		Chrome 76 552 days ago (Jul 30, 2019)		Chrome 77 510 days ago (Sep 10, 2019)		Chrome 78 468 days ago (Oct 22, 2019)	
Web Share Target		+					
Shipped							
Web Share API Level 2		+					
Shipped							
		Web Share Target Level 2		+			
		Shipped					
		Async Clipboard: Read and Write Images		+			
		Shipped					
		Enter Key Hint				+	
		Shipped					
						Expand Storage Quota	
						Shipped	
						Periodic Background Sync	
						+	
						Origin Trial	

<div> <div>  </div> <div> <div>STABLE</div> <div>  <div>Chrome 88</div> <div>13 days ago (Jan 19, 2021)</div> </div> </div> </div>				<div> <div>  <div>BETA</div> <div>Chrome 89</div> <div>In 29 days (Mar 2, 2021)</div> </div> </div>				<div> <div>  <div>DEV</div> <div>Chrome 90</div> <div>In 71 days (Apr 13, 2021)</div> </div> </div>				<div> <div>  <div>CANARY</div> <div>Chrome 91</div> <div>In 113 days (May 25, 2021)</div> </div> </div>			
Web Share Target Level 2				+											
Shipped															
Contacts API				+											
Shipped															
PWA Badging API				+											
Shipped															
Barcode Detection API				+											
Shipped															
Standby API				+											
Shipped															
WebOTP				+											
Shipped															
PWA App shortcuts				+											
Shipped															
File System Access				+											
Shipped															
Pan/Tilt support for Camera				+											
Shipped															
Web Serial API				+											
Origin Trial				Expected to ship											
Web NFC				+											



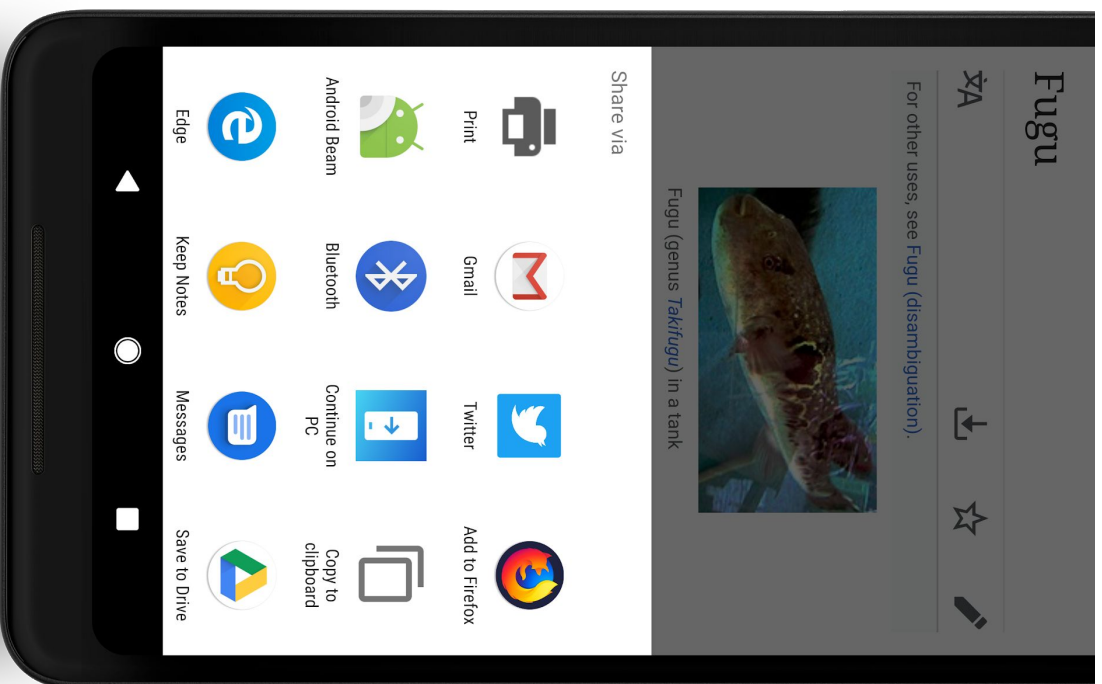
Sharing Content +
Receiving Shares



Web Share

Use Cases

The web should be able trigger shares to the system share sheet, making sharing super simple!



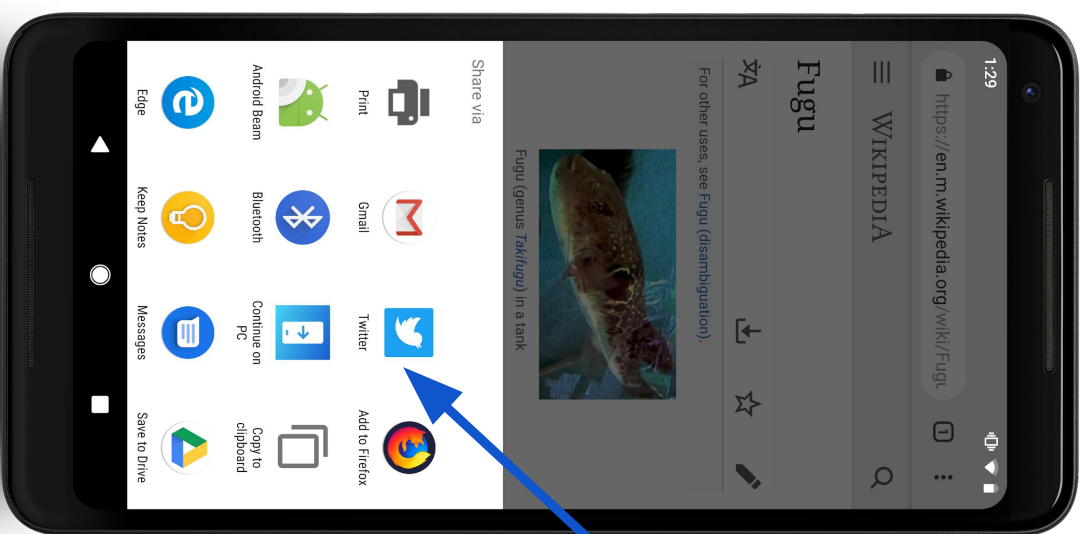
```
if ('share' in navigator) {  
  const shareOpts = {  
    title: 'Unlocking New Capabilities for the Web',  
    text: 'One of the most exciting talks at I/O',  
    url: 'https://www.google.com',  
  }  
  navigator.share(shareOpts)  
    .then(() => {...})  
    .catch((e) => {...});  
}
```



Receiving Shares

Native applications can act as share targets and receive content like text or files.

Web apps like social media sites should be able to register themselves as share targets to the system.



Extends Web App Manifest

```
"share_target": {  
  "action": "/share-target/",  
  "method": "GET",  
  "enctype": "application/x-www-form-urlencoded",  
  "params": {  
    "title": "title",  
    "text": "text",  
    "url": "url"  
  }  
}
```

Web Share Web Share Target

Explainer

bit.ly/2H0PuUF

Specification

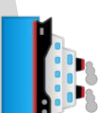
bit.ly/2HegLmX

Updates Post

bit.ly/2J4unUj

Demo

bit.ly/2HehOyn



Shipped!

But working on more OS
support

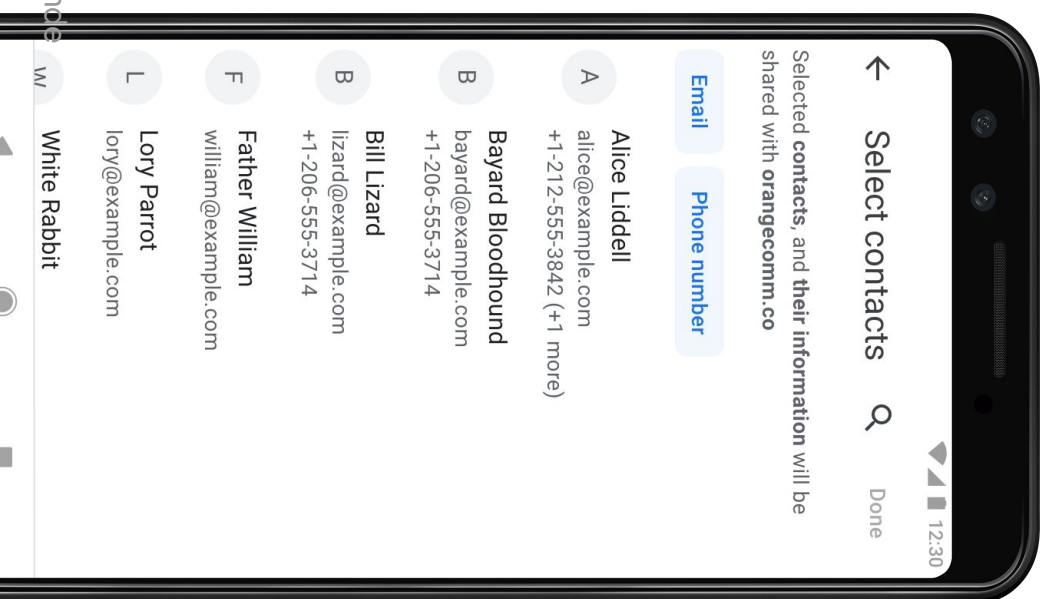




Accessing the users contacts

Native apps can easily ask for a access to a user's contacts in their address book.

Web apps should be able to



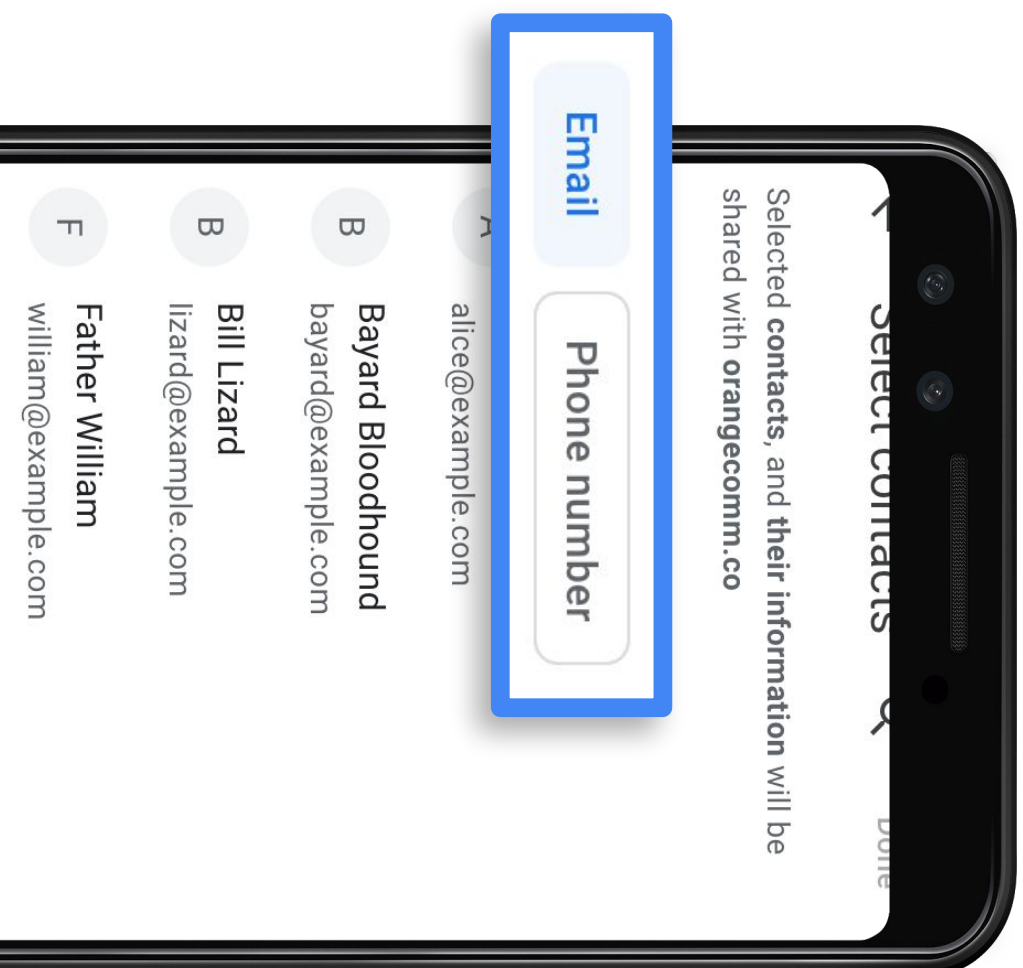
Contacts Picker

Security Consideration

Give developers the features they need while ensuring users are in control of their own and their contacts' information and understand what they share.

The contacts picker will only work when served from a secure host, and it can only be shown with a deliberate user gesture.

Users can also explicitly choose which contacts they want to share, and they see what is being shared before they share it.



```
const options = {  
  multiple: false,  
  properties: ['name', 'email', 'tel'],  
};
```

```
try {  
  const contacts = await navigator.contacts.select(options);  
  // Do something with the contacts  
} catch (e) {  
  // Handle error  
}
```

```
[{  
  "email": ["kenneth.r.christiansen@intel.com"],  
  "name": ["Kenneth Christiansen"],  
  "tel": []  
}]
```

Contact Picker

Explainer

bit.ly/2Jd34XV

Specification

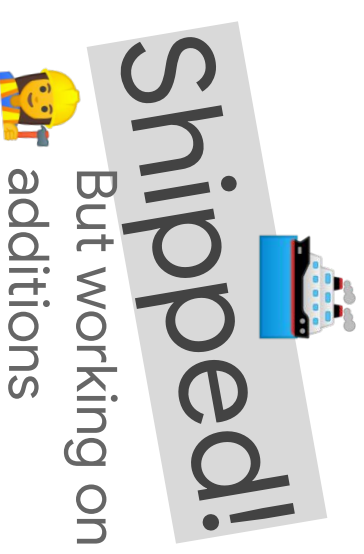
bit.ly/2UzD5y0

Updates Post

web.dev/contact-picker

Demo

bit.ly/2Y5bbK6





Badging

Use Cases

Subtly alert users of incoming events like a long-running task that has finished or an app's state, like the number of unread messages.



Mock - demonstration purposes only.

@kennethrohde

```
// Setting a badge  
window.Badge.set(42);
```

```
// Set the badge to a simple dot on the icon  
window.Badge.set();
```

```
// Alternatively, explicitly clearing it  
window.Badge.clear();  
window.Badge.set(0);
```

Badging API

Explainer

bit.ly/2Hejk8z

Specification

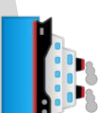
bit.ly/2H7osec

Updates Post

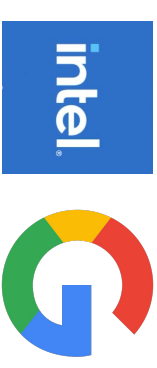
bit.ly/301m1cX

Demo

bit.ly/2H4XT9N



Shipped!



Detecting Barcodes ~~and~~ Faces

A lot of things in the world have barcodes that you can scan with your phone camera, and when your camera detects a face, it will highlight it with a boundary box.



Shape Detection API

Use Cases

Scanning all sorts of barcodes (QR codes, EAN-13 codes, Aztec codes, etc.) to identify products, make payments, and many more.



```
if ('FaceDetector' in window) {  
  const faces = await new FaceDetector().detect(img);  
} else {  
  // Use fallback  
}
```

```
if ('BarcodeDetector' in window) {  
  const codes = await new BarcodeDetector().detect(img);  
} else {  
  // Use fallback  
}
```

Shape Detection API

Explainer

bit.ly/2JkrmpI

Specification

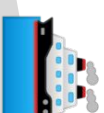
bit.ly/2VkgIvE

Updates post

bit.ly/2Vl6ZJm

Demo

bit.ly/2H1gGTg



Partly shipped

Shipped Barcode API



Keeping an App Alive

Native apps can use wake locks to prevent a device from sleeping or to keep a device's screen on.

Web apps should be able to do the same.



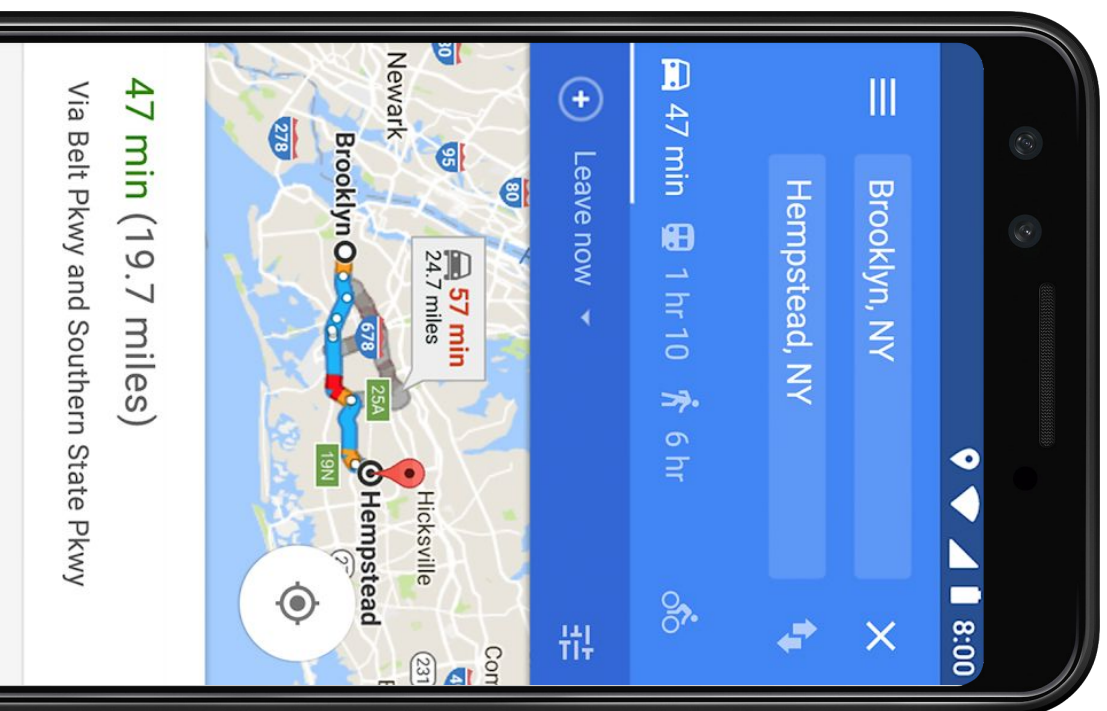


Wake Lock

Use Cases

Prevent a device from sleeping so a navigation app can get you from point A to point B and can notify you of turns you need to take.

Keep the screen of a device on while playing chess, so you can think about your next move as long as you want.





Two different kinds of wake locks



A **screen** wake lock prevents the device's screen from turning off so that the user can see the information that's displayed on screen.



~~A **system** wake lock prevents the device's CPU from entering standby mode so that your app can continue running.~~


```
function tryKeepScreenAlive(minutes) {  
  navigator.wakeLock.request("screen").then(lock => {  
    setTimeout(() => lock.release(), minutes * 60 * 1000);  
  });  
}
```

```
tryKeepScreenAlive(10);
```

Wake Lock

Updates post

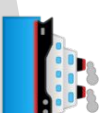
bit.ly/2IVyCCd

Specification

bit.ly/2DNcINh

Demos

bit.ly/2VBZBJ7



Shipped!



One Time Password

Use Cases

One time password delivered via SMS for:

- Two-factor authentication
- Phone number verification
- Account recovery
- Payment confirmation



Two standards



Origin-bound one-time codes delivered via SMS

A special way to format an SMS so that a code can be read and delivered to the right site



Web OTC API

Site can access code if user allows directly using this API

```
navigator.credentials.get({  
  otp: {transport:['sms']}  
})  
.then(otp => input.value = otp.code);
```

Your OTP is 123456

@web-otp.glitch.me #123456

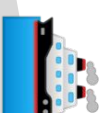
Web OTC

Updates post

web.dev/sms-otp-form/

Specification

>check post<

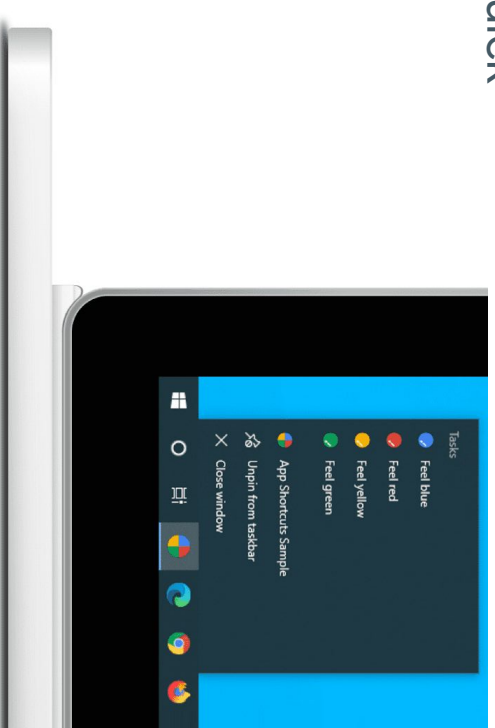
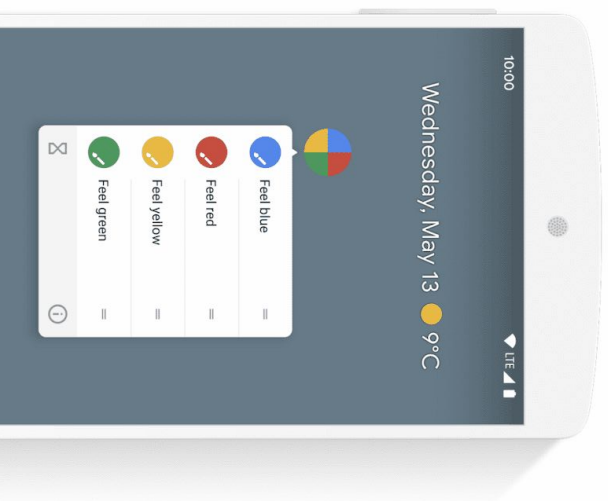


Shipped!



App Shortcuts

Long press on mobile and right click on desktop brings up a shortcut list to quick actions



```

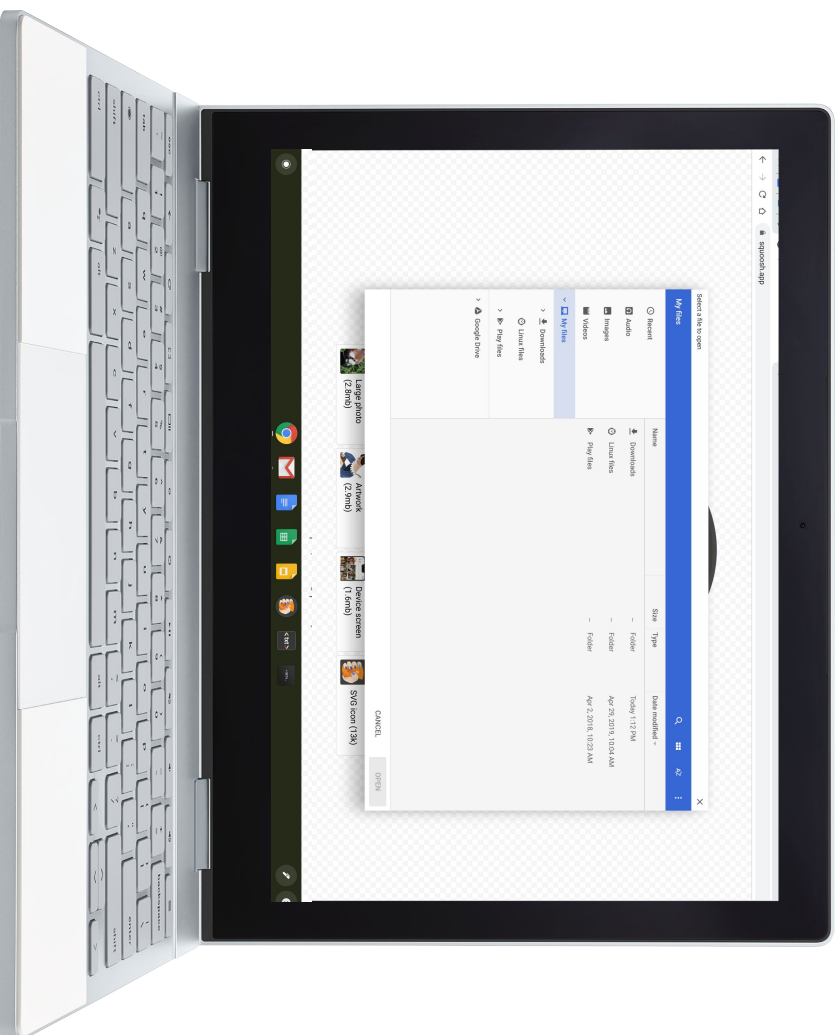
{
  "name": "Player FM",
  "start_url": "https://player.fm?utm_source=homescreen",
  ...
  "shortcuts": [
    {
      "name": "Open Play Later",
      "short_name": "Play Later",
      "description": "View the list of podcasts you saved for later",
      "url": "/play-later?utm_source=homescreen",
      "icons": [{ "src": "/icons/play-later.png", "sizes": "192x192" }]
    },
    {
      "name": "View Subscriptions",
      "short_name": "Subscriptions",
      "description": "View the list of podcasts you listen to",
      "url": "/subscriptions?utm_source=homescreen",
      "icons": [{ "src": "/icons/subscriptions.png", "sizes": "192x192" }]
    }
  ]
}

```




Reading and writing to files directly





File System Access

Web apps can open files and upload them to a server or work with them locally, but when you want to save, you have to download them and overwrite your local copy.

Web apps should be able to just read and write local files upon permission being granted.

📁 File System Access API

With File System Access API, we can save file to disk!

In this demo, i am using .txt to demonstrate but any file can be used.

— The File System Access API (formerly known as Native File System API and prior to that it was called Writeable Files API) enables developers to build powerful web apps that interact with files on the user's local device, like IDEs, photo and video editors, text editors, and more. After a user grants a web app access, this API allows them to read or save changes directly to files and folders on the user's device. Beyond reading and writing files, the File System Access API provides the ability to open a directory and enumerate its contents.

For more info: <https://web.dev/file-system-access/>

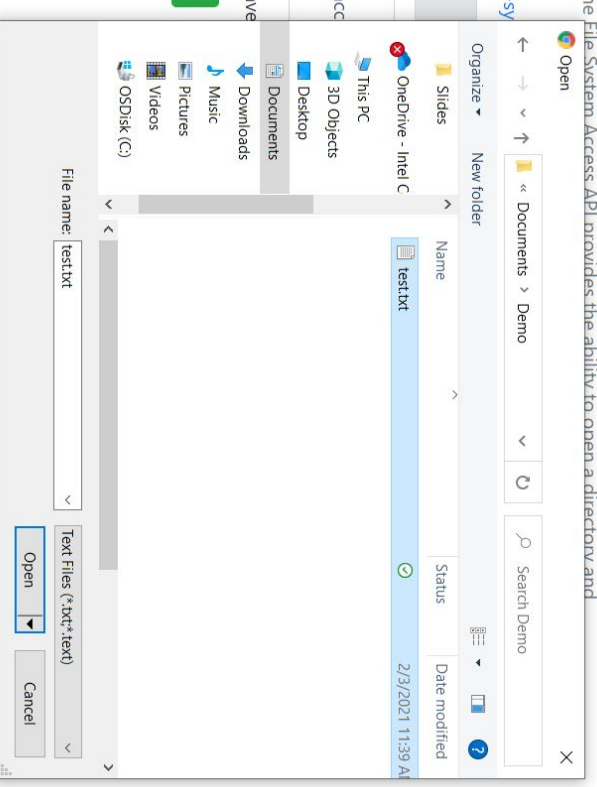
Try on Chrome 86 and above.

It would be so nice if PWAs had access to the File System Access API!

Now you can edit test.txt and save changes

Pick a .txt file

Save changes



<https://googlechromelabs.github.io/text-editor/>

@kennethrohde

```
const fileRef = await self.chooseFileSystemEntries(fileOpts);

// Read from the file
const file = await fileRef.getFile();
const image_data = await file.arrayBuffer();

// Write to the file
const writer = await fileRef.getWriter();
await writer.write(0, new_image_data);
```

Working on additions

- Seek past end of file
- Suggested file name and location
 - Save an existing file in a different directory
 - Save a file using a different name in the same directory
 - Open a file from the same directory as a currently open file or project
 - Prompt a user to open a image (or video, or audio file)
 - Remember different last-used directories for different purposes

File System Access API

Explainer

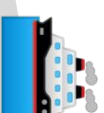
bit.ly/2Hek3GP

Specification

bit.ly/2H1titL

Updates Post

bit.ly/2LpvnVJ



Shipped!

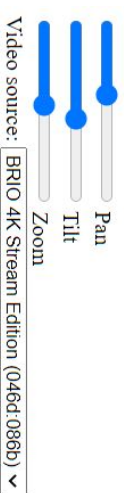


But working extensions

Image Capture Pan/Tilt/Zoom



See the [README](#) for a list of supported cameras.



See the [README](#) for a list of supported cameras.

@kennethrohde

<https://rju.github.io/WebCamera/samples/panTilt/>



Pan/Tilt/Zoom

Explainer

[N/A](#)

Specification

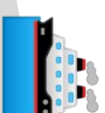
[bit.ly/33Kkovp](#)

Updates Post

[bit.ly/2UwmulK](#)

Demo

[bit.ly/2xm3xTG](#)



Shipped!

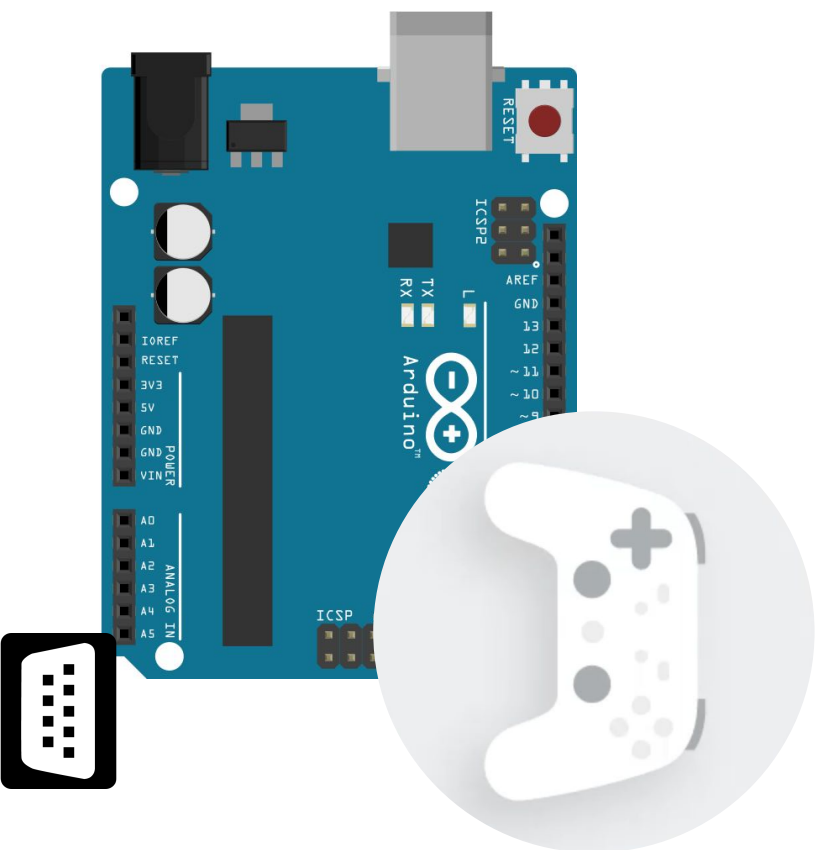


Serial API / Web HID

Use Case

Serial: Connect to an Arduino or other microcontroller kit and program its microcontroller right from a web app.

HID: Logic for gamepads and other devices can be moved to JavaScript, providing more support than is possible from the browser.



Talk to external hardware

- Multiple standards
 - Web Bluetooth (shipped)
 - Web USB (shipped)
 - Web Serial (over USB but claimed by OS first)
 - Web HID (Human Interface Device, over USB or BT)

Web Serial

Explainer

bit.ly/2VcZRP5

Specification

bit.ly/2GSld9B



Shipping soon

Tentatively shipping in M89

Web HID

Explainer

bit.ly/2GY8inc

Specification

bit.ly/2H4rzUe



Shipping soon

Tentatively shipping in M89



Near Field Communication

Read and write to NFC tags using the
NFC data exchange format (NDEF)

Near Field Communication



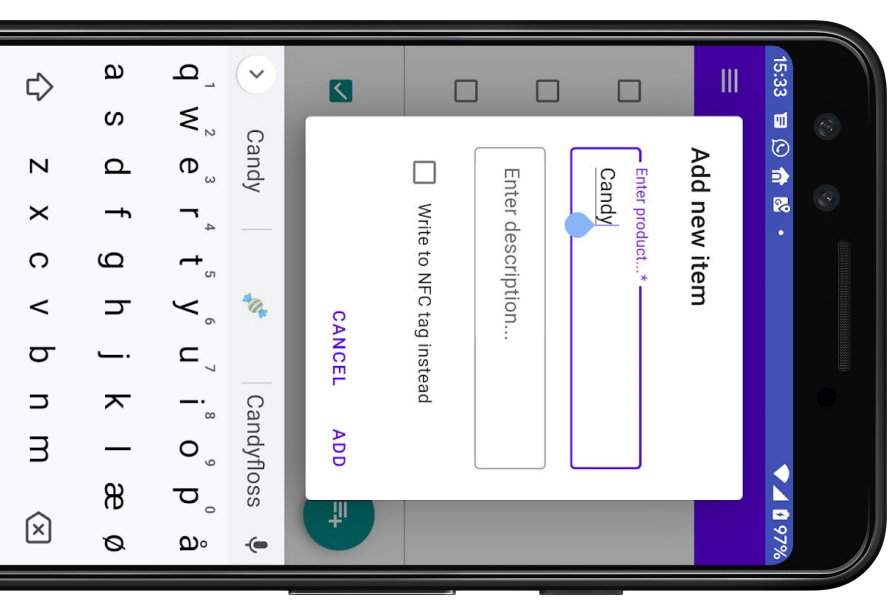
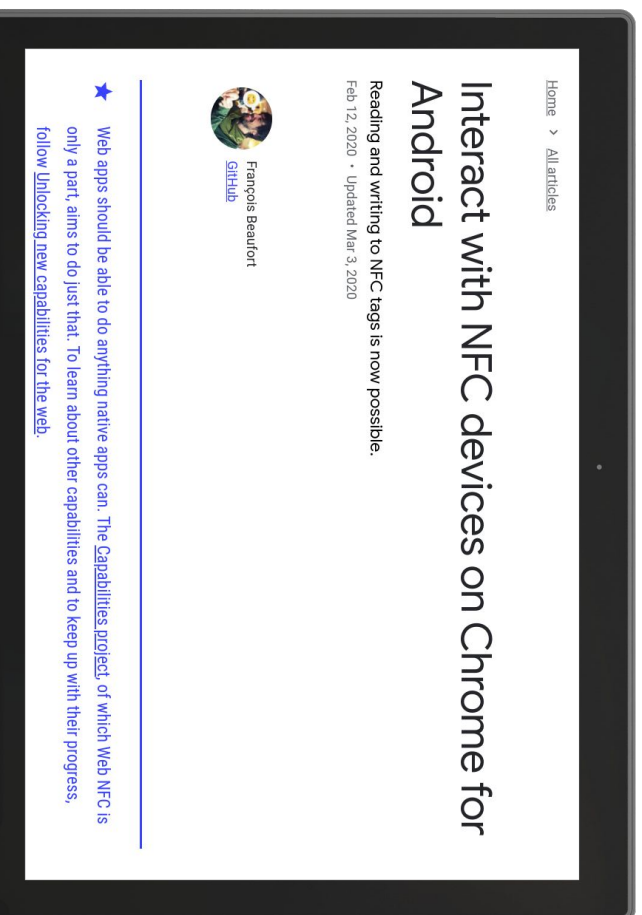
Read and write NDEF records to NFC tags



Very complete NDEF support



Learn more @ web.dev/nfc

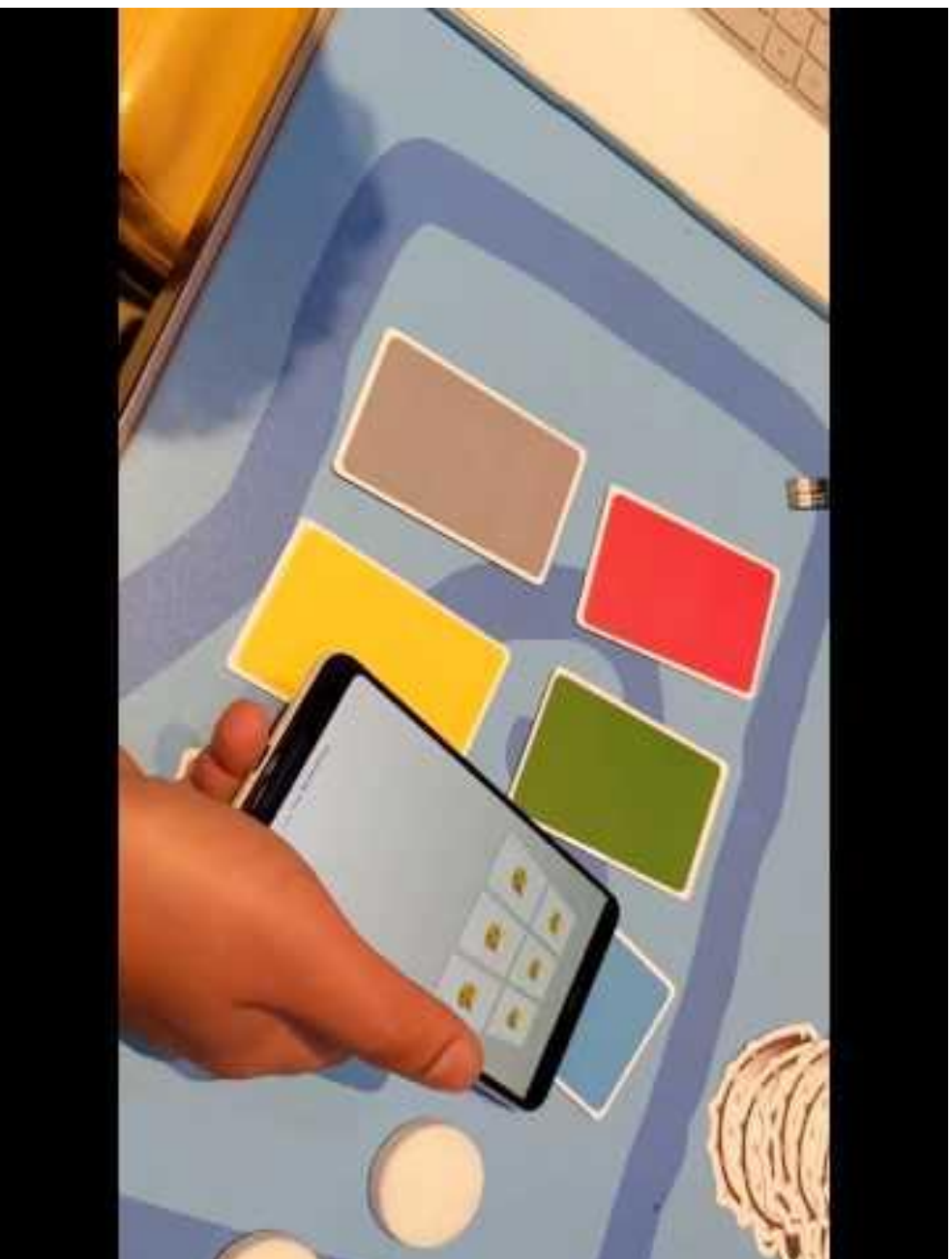


Pretty cool, even

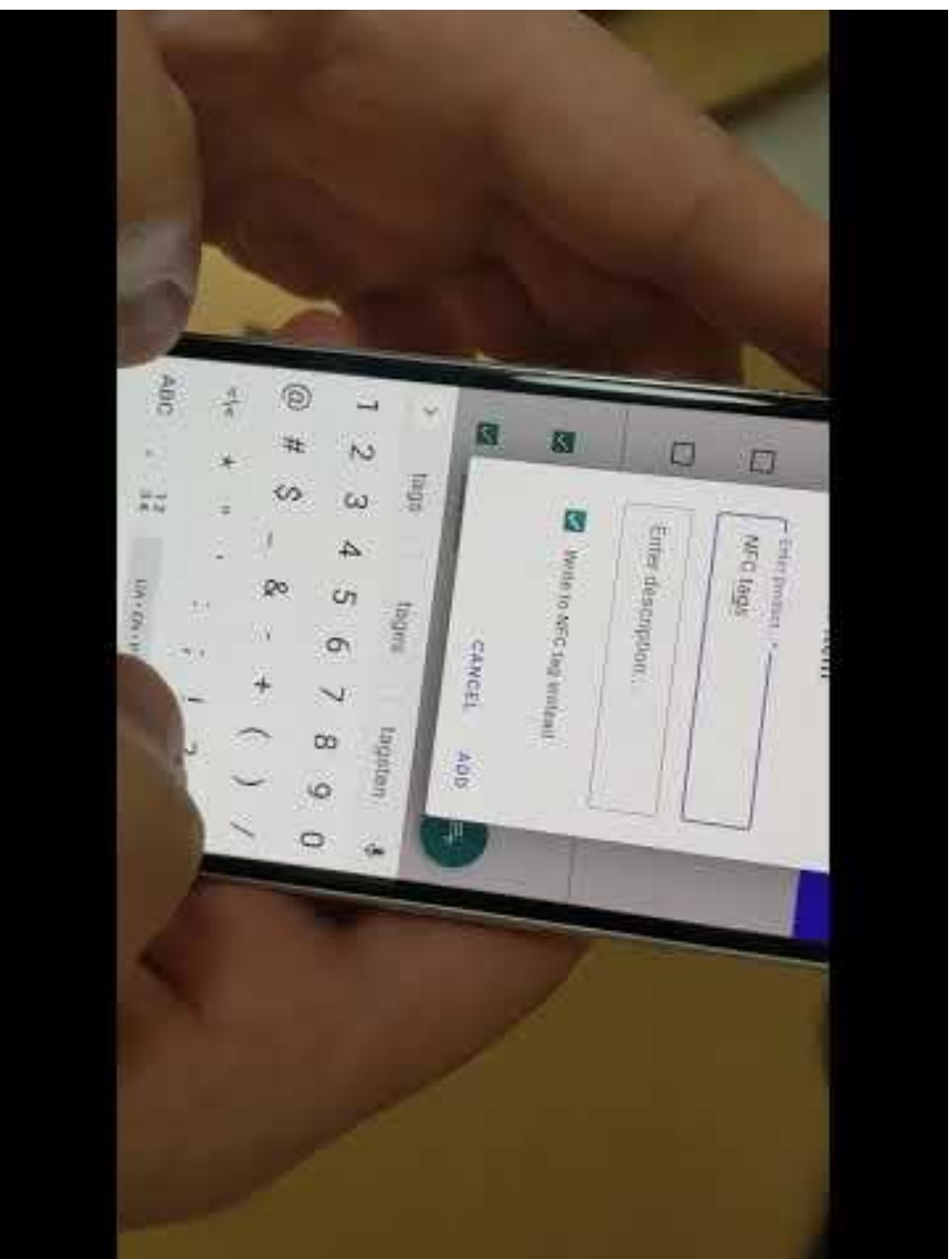
Sir Tim Berners-Lee is excited



NFC based card game



NFC based grocery list



```
const reader = new NDEFReader();

reader.onreading = event => {
  for (let record of event.message.records) {
    // ...
  }
};

const ctr = new AbortController;
reader.scan({ signal: ctr.signal });
```

```
const reader = new NDEFReader();

reader.onreading = event => {
  for (let record of event.message.records) {
    if (record.recordType === "json") {
      console.log(`JSON: ${record.toJSON().myProperty}`);
    }

    if (record.recordType === "opaque" && record.mediaType.startsWith('image/')) {
      const blob = new Blob([record.toArrayBuffer()], {type: record.mediaType});

      const img = document.createElement("img");
      img.src = URL.createObjectURL(blob);
      img.onload = () => window.URL.revokeObjectURL(this.src);
      document.body.appendChild(img);
    }
  }
};

const ctr = new AbortController;
reader.scan({ signal: ctr.signal });
```

Web NFC

Explainer

bit.ly/3adX4sx

Specification

bit.ly/33AySh2

Updates post

web.dev/nfc

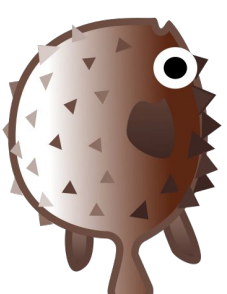
Demos

bit.ly/2ZdaNqO

























































































































Shipping soon

Tentatively shipping in M89















































Project **Fugu**








































Ready for the third year

Fugu API Tracker			
Shipped			
Web Share Target	 M71		+
Web Share API Level 2	 M75		+
Async Clipboard: Read and Write Images	 M76	    	+
Web Share Target Level 2	 M76		+
Enter Key Hint	 M77		+
Expand Storage Quota	 M78	    	+
Contacts API	 M80		+
Get Installed Related Apps API	 M80	    	+
Compression codecs	 M80	    	+
PWA: desktop-pwas: Support "minimal-ui" display mode	 M80	    	+
Periodic Background Sync	 M80	    	+
PWA: Badging API	 M81	  	+
PWA: Allow the Badging API to be used from a service worker via Push	 M81	    	+
Barcode Detection API	 M83	 	+
Content Indexing API	 M84	    	+
WebOTP	 M84		+
Screen Wake Lock API	 M84	     	+
Streams API: transferable streams	 M85	    	+
PWA: App shortcuts	 M85		+
File System Access	 M86	    	+
text/html support for async clipboard api	 M86	    	+
Pan/Tilt support for Camera	 M87	    	+
PWA: PWA should be able to be uninstalled the same way a "real app" can	 M88	  	+
PointerLock unadjustedMovement	 M88	   	+
Create a Photo/Video picker similar to the Photo Picker on Android	 M88	    	+
Digital Goods API	 M88	    	+

Origin Trial [\(details\)](#)

WebSocketStream	 M72-80 	  	+
Web Serial API	 M80-88 	  	+
Web NFC	 M81-83 		+
QuickTransport	 M84-86	  	+
WebHID (Human Interface Device)	 M85-87 	  	+
PWA Multi-Screen Window Placement	 M85-88 	  	+
WebCodecs	 M85-88	  	+
Notification Triggers	 M85-88	  	+
Local Font Access	 M87-89	  	+
Idle Detection	 M88-90 	  	+

Developer Trial - behind a flag [\(details\)](#)

Ambient Light Sensor API - based on Generic Sensor API.	 M62	  	+
PWA File Handling	 M78 	  	+
Raw Clipboard Access API	 M83	  	+
PWA Tabbed application mode for PWAs	 M83	  	+
Web Bluetooth BluetoothDevice.watchAdvertisements()	 M85	  	+
NativeIO	 M85	  	+
PWA Run PWA on OS Login	 M88 	  	+
PWA URL Protocol Handler Registration for PWAs	 M89	  	+
PWA Declarative Link Capturing	 M89 	  	+

Started

ChromeOS: Allow webpages to consume alt+click



bit.ly/new-fugu-request



Resources



Fugu API tracker

<https://fugu-tracker.web.app/>



File new request

<https://bit.ly/new-fugu-request>



Fugu community sync notes

<https://bit.ly/fugu-sync>

Get involved
today!

Thanks for listening!



Let's keep in touch on
Twitter [@kennethrohde](#)



Don't forget to

vote for this session
in the **GOTO Guide app**