WebID

And the Privacy Sandbox

@goto, @kaustubhag

WICG Workshop 2021

The Privacy Sandbox

The Privacy Sandbox initiative aims to create web technologies that both protect people's privacy online and give companies and developers the tools to build thriving digital businesses to keep the web open and accessible to everyone.

The Privacy Model [1]

1. Identity is partitioned by First Party Site
2. Third Parties can be allowed access to a first-party identity
3. A per-first-party identity can only be associated with small amounts of cross-site information

[1] https://github.com/michaelkleber/privacy-model

APIs

SaaS Embeds (CHIPS), Spam/Fraud (Trust Tokens), Ads (attribution reporting, etc), First Party Sets, Fenced Frames, ...

... and ...

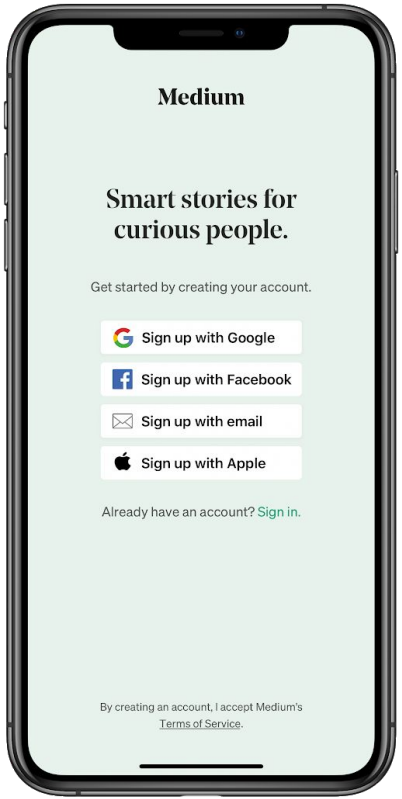
Federated Identity

What is it?

Users sign-in to a RP (relying party) with an IDP (Identity provider)

Why do we think it's important?

Federated identity is safer* than per-site usernames and passwords



* phishing, password reuse, etc

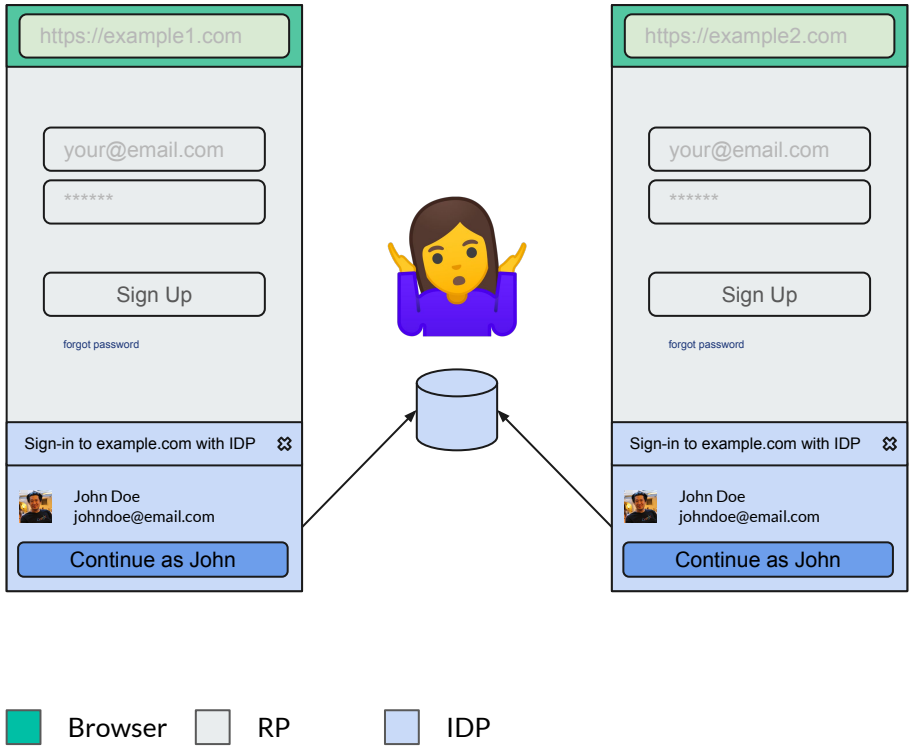
The classification Problem

What's the problem?

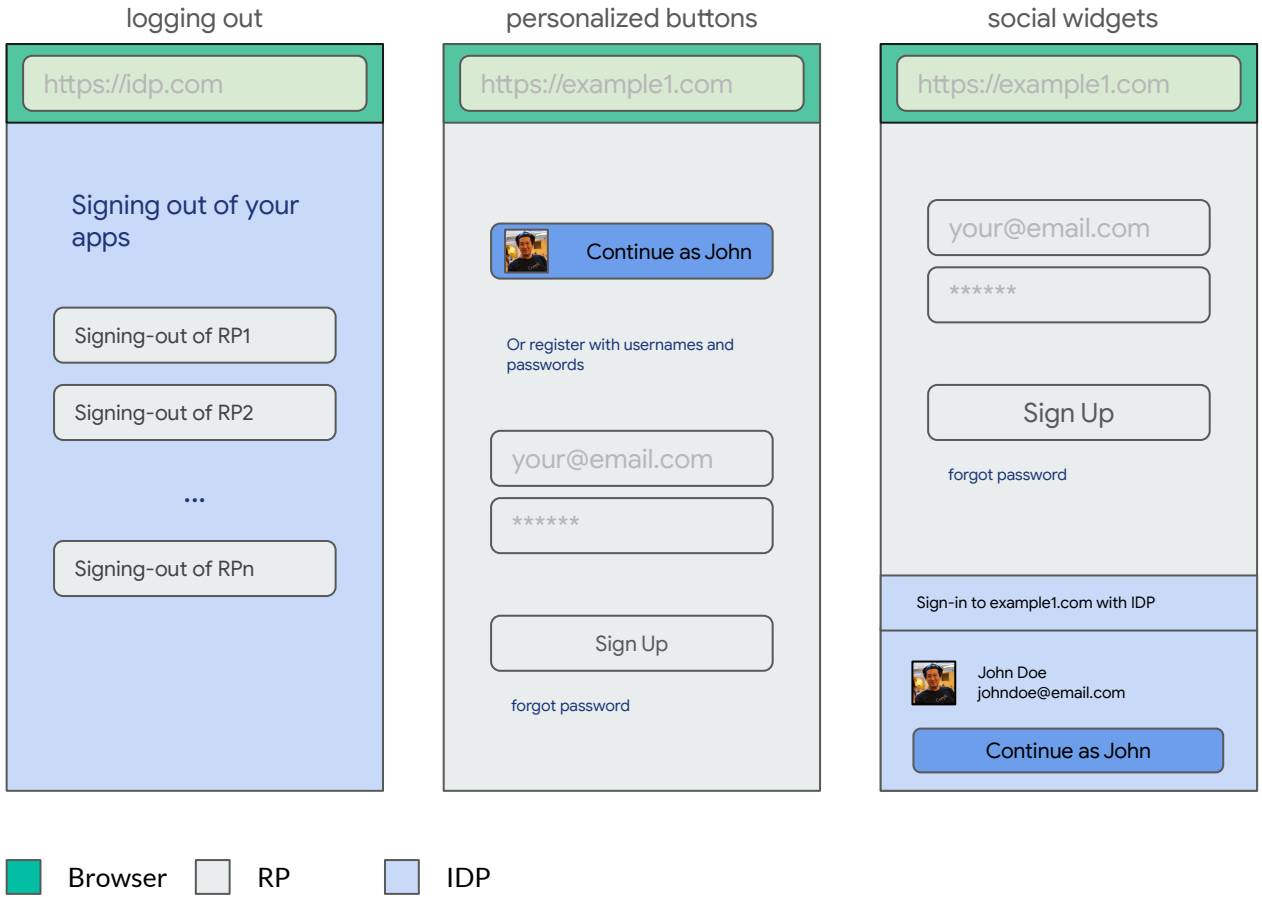
By design, Identity federation was built on top of low-level primitives*.

By accident, the same primitives also enable cross-site tracking.

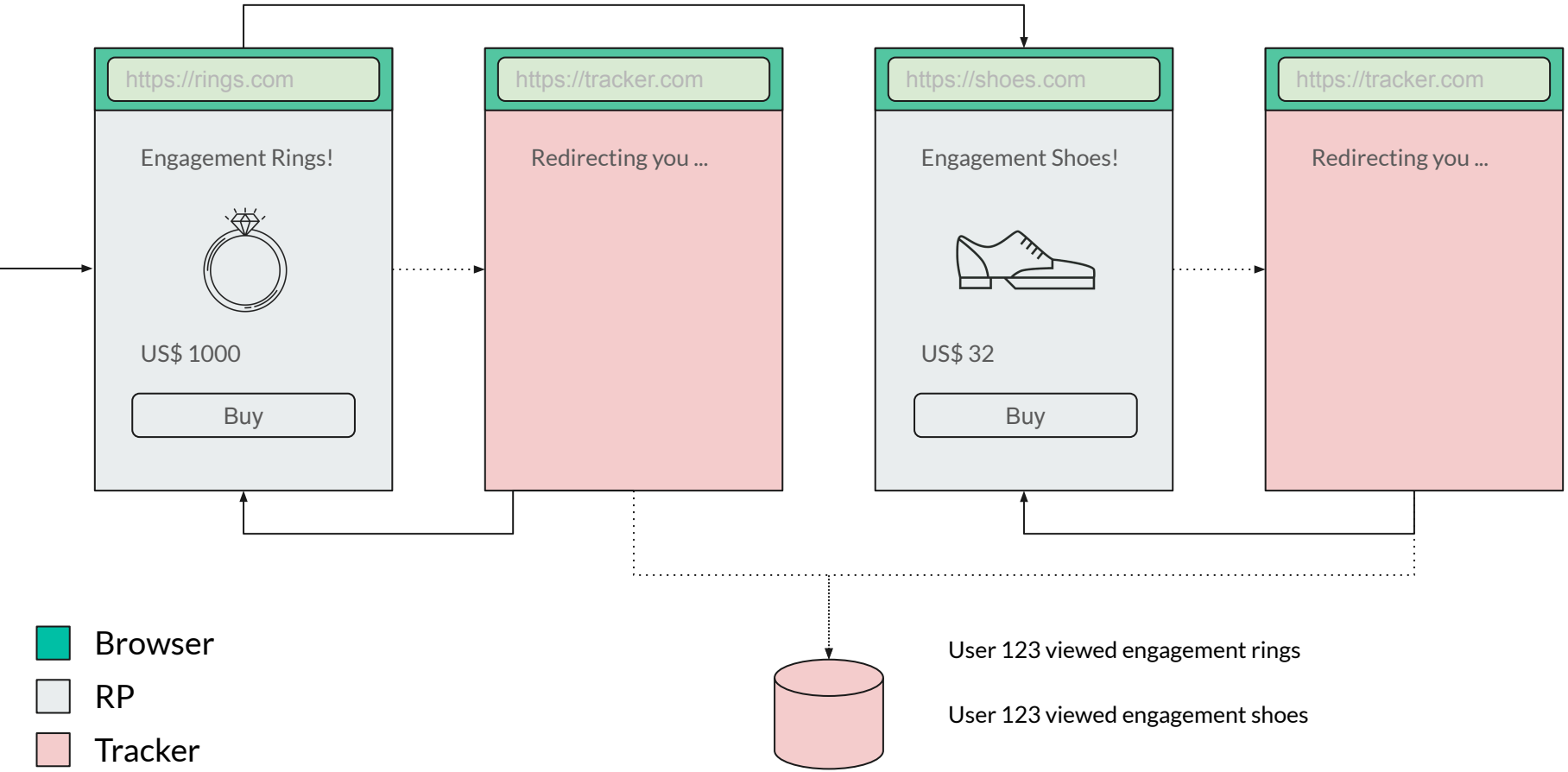
* iframes, third party cookies, redirects



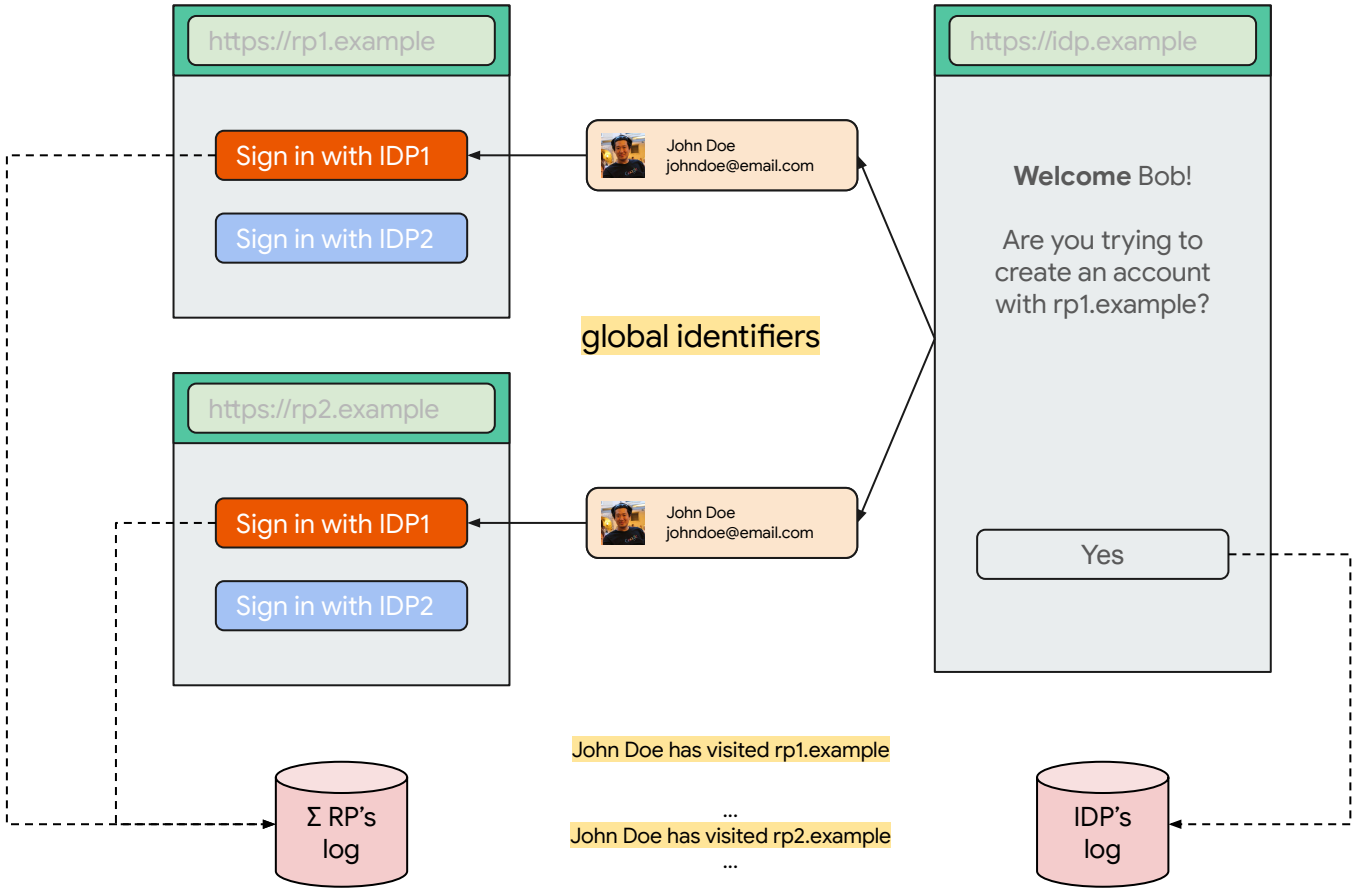
3P cookies use in Federation



The Classification Problem



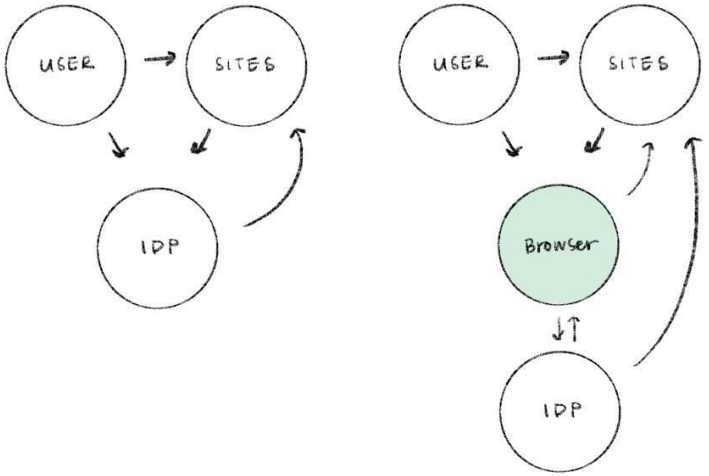
The IDP and RP Tracking Problems



What's WebID*?

A high-level, identity-specific, privacy-preserving browser API that enables identity federation to continue thriving on the web.

* Oops. Web Sign-in?



Useful principles so far

Users first*.

Developers 2nd, browser engines 3rd, technical purity 4th.

* you'd be surprised how often this is used.

Private by default.

Partitioned by first party site by default, global* by choice [1].

* yes, we expect enterprises and education to pick different defaults.

[1] https://github.com/michaelkleber/privacy-model

Minimize redeployment.

O(B) of users*, O(M) of RPs**, O(K) of IDPs***, O(10) browser engines.

* no, we won't start with the NASCAR
** yes, customer support is hard too
*** consumer IDPs control JS SDKs!

Classes of solutions

Permission

Browser is only involved to capture user consent for tracking.

Pros

Backwards compatible. Extensibility.

Cons

Permission-blindness* ineffective at driving change.

* on the way of the job to be done

Mediation

Browser renders parts of the IDP flow in the browser consent moments.

Pros

Deployable by IDPs. Meaningful permission.

Cons

Ossification*.

* basic auth anyone?

Delegation

IDP delegates much of the responsibility for minting tokens to the browser.

Pros

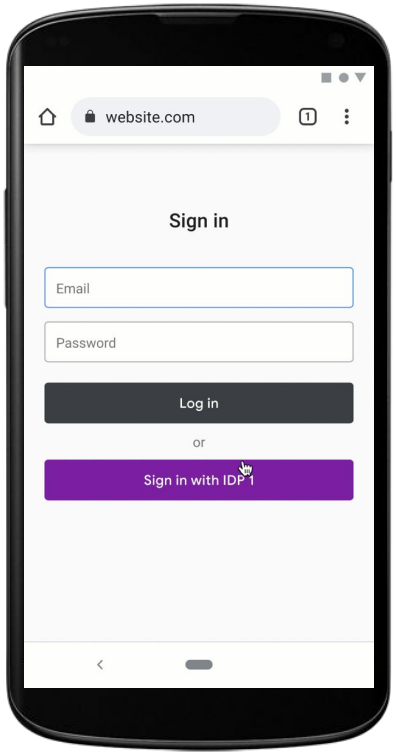
Frictionless, consequence-free.

Cons

RP backwards incompatible *.

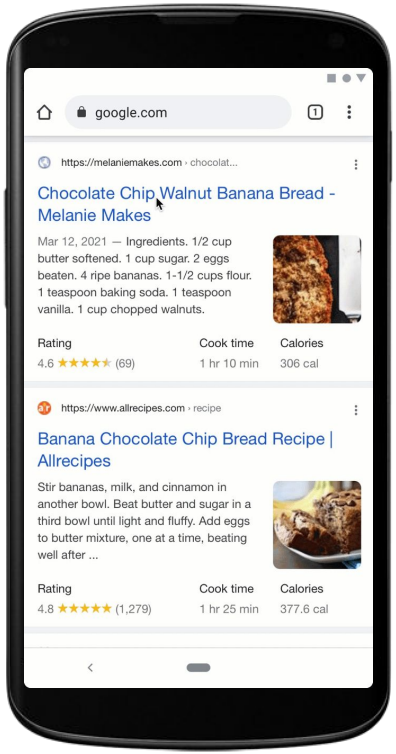
* reminder: O(M) of RPs

The Permission-oriented API



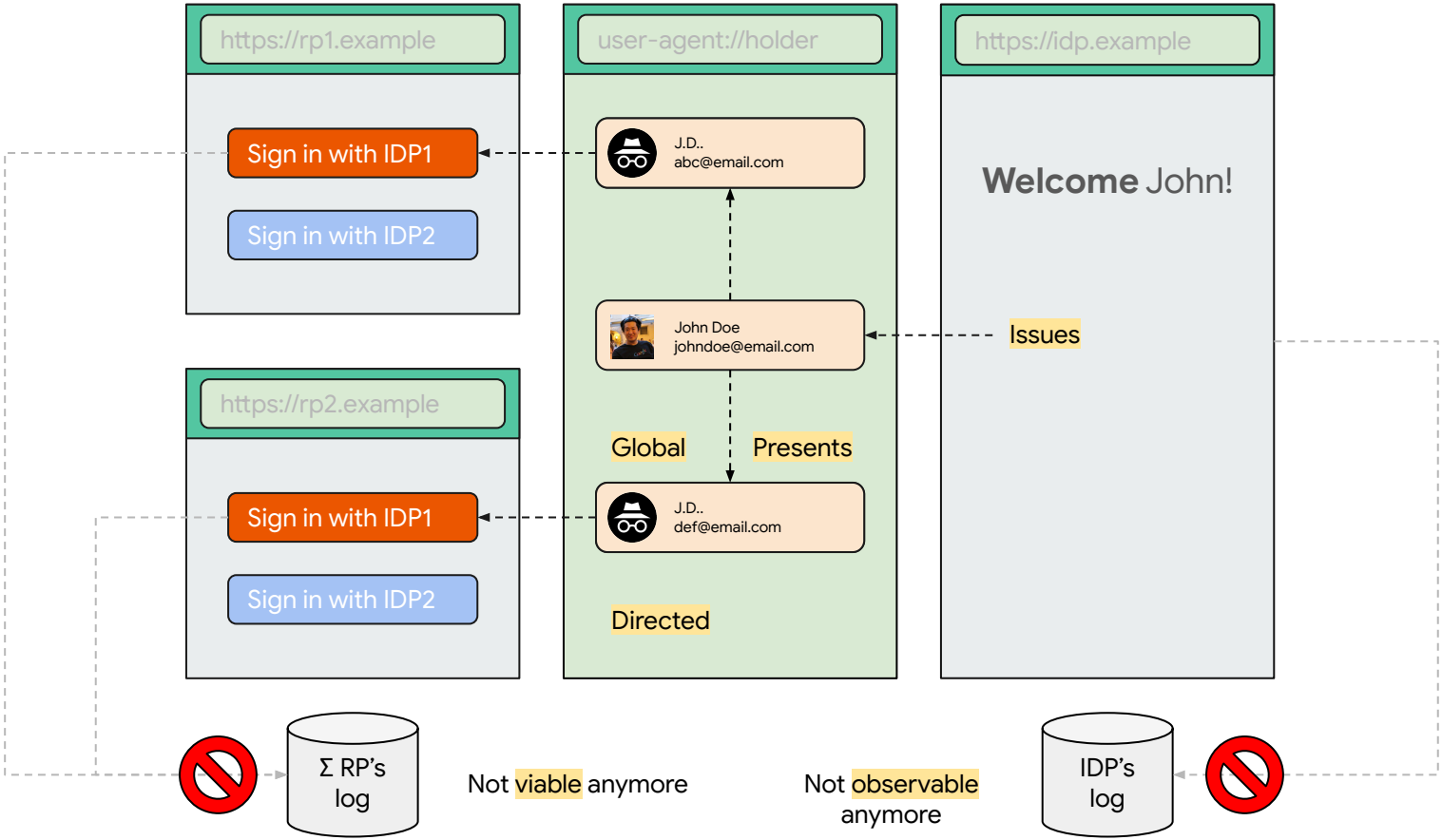
Pros: most backwards compatible
Cons: permission-blindness

The Mediation-oriented API

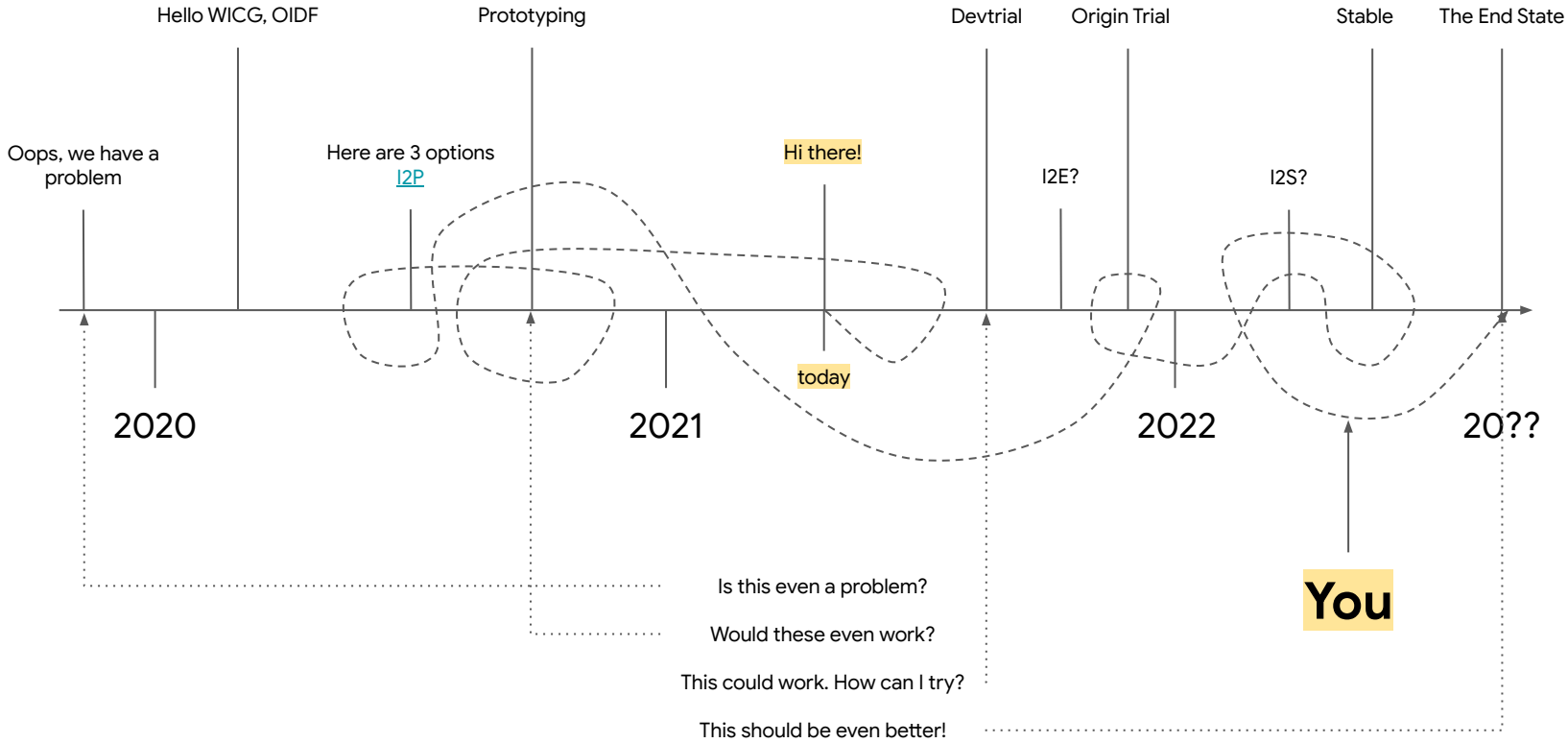


Pros: meaningful permission
Cons: ossification

The **unbundling** of global identification into directed and The **unbundling** of issuing and presentation



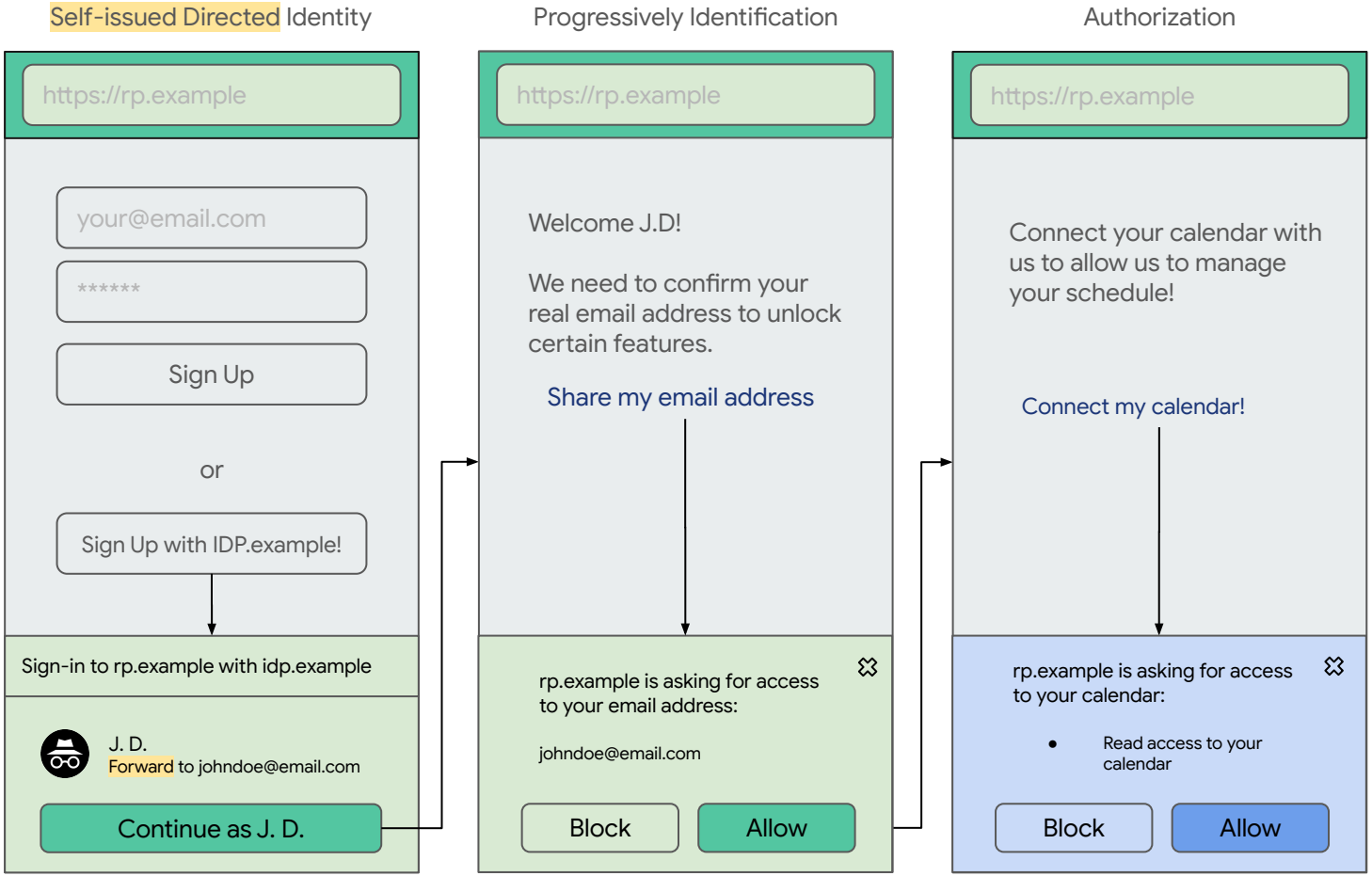
Why > What > How > Who > When



Thanks!

ANNEX

The **progressive** disclosure of identification

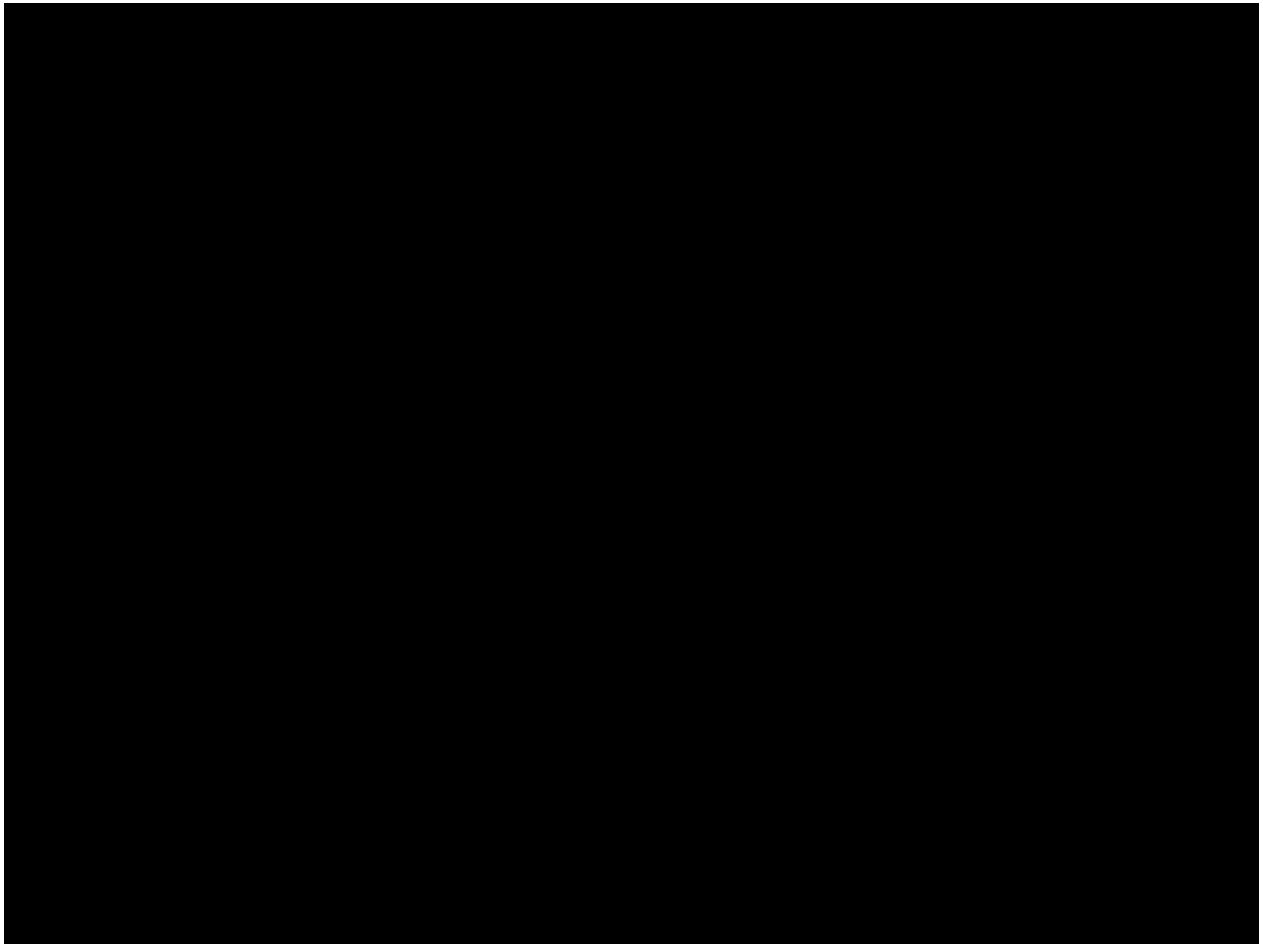


Show me the code

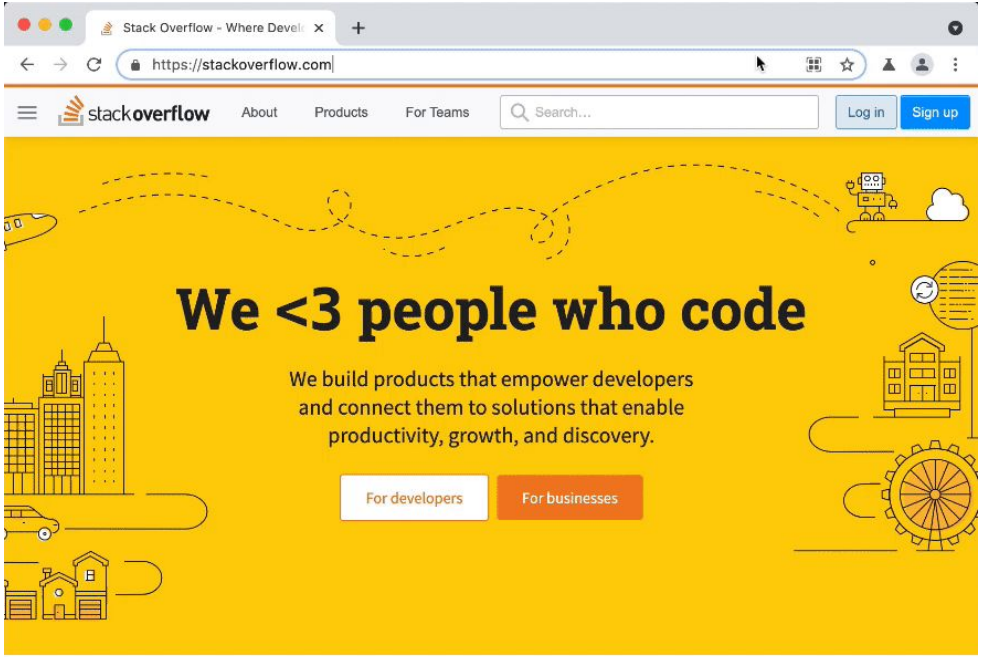
```
// Exact API largely TBD. Our best guess so far:

const token = await navigator.credentials.get({
  provider: "https://idp.example",
  request: {
    client_id=”1234”,
    nonce: “Ct60bD”,
  },
  mode: "permission" || “mediation” || “delegation”,
});
```

Demos

The Mediation-oriented API



The Permission-oriented API

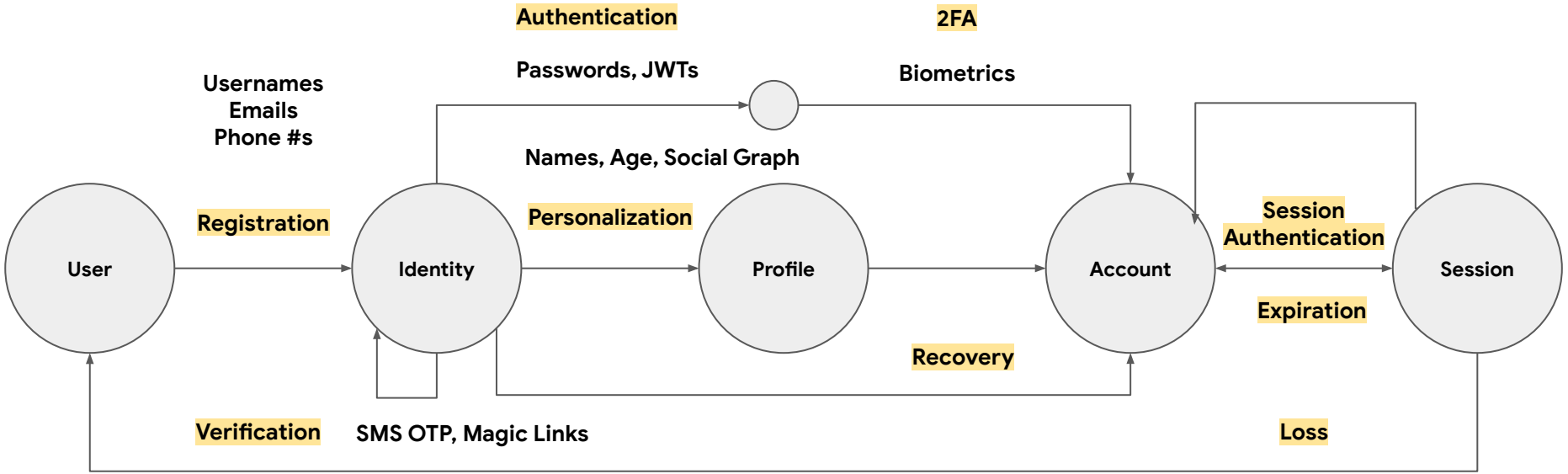


WebAuthn?
WebOTP?
Forms?

The Identity Lifecycle

WebID + Forms

WebAuthn



WebOTP

Cookies

GC