
Pop-ups are back
(if you haven't already noticed)
THIS WEBSITE USES COOKIES
I ACCEPT

Beyond Pop-Ups

Engineering a browser firewall to respect user's consent preferences, without the UX nightmares.

Ben Brook
Co-founder and CEO, Transcend

JUNE 10, 2021

Transcend

Pitch

1. How we got here

2. Engineering a way out of here

Pitch

Pops-ups are back. What went wrong?

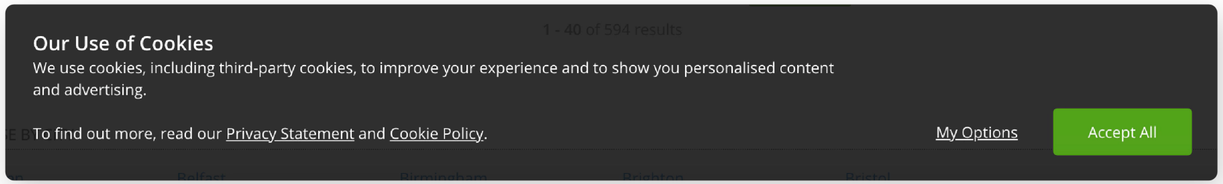
Web teams rely on **conversion funnels**, which track a user's journey from landing on the website to converting, **but...**

... GDPR requires websites to get **consent before collecting data about a user. So...**

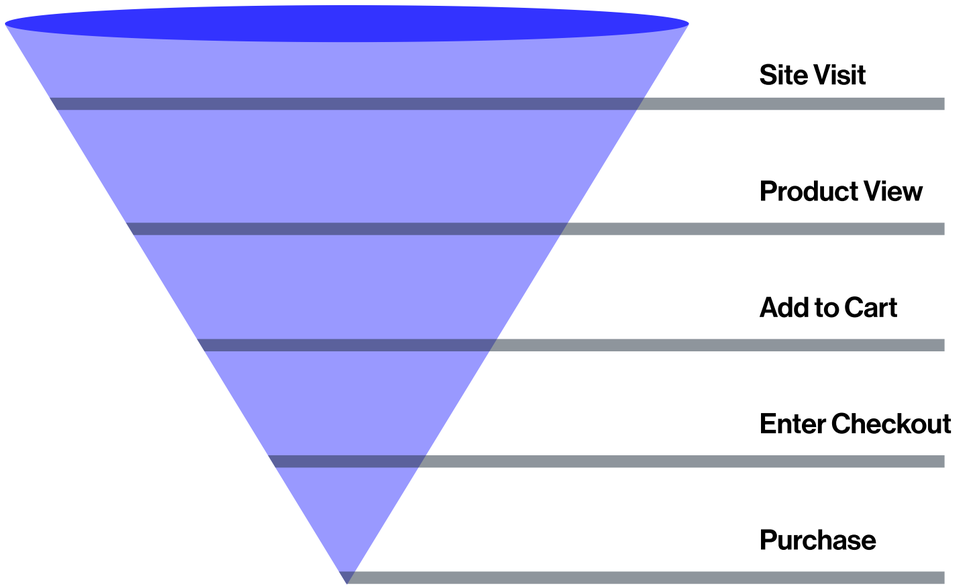
Outcome:
In order to get the full conversion story, while complying with modern privacy laws, **companies ask for consent as early as possible**.

Pitch

The Status Quo UX



Get consent and begin **tracking** + **uploading**



Pitch

The pop-up remains the best way to get consent as early as possible.

We ask for consent as early as possible because we need to start tracking as early as possible.

Pitch

Are pop-ups really the new user experience, now and forever?

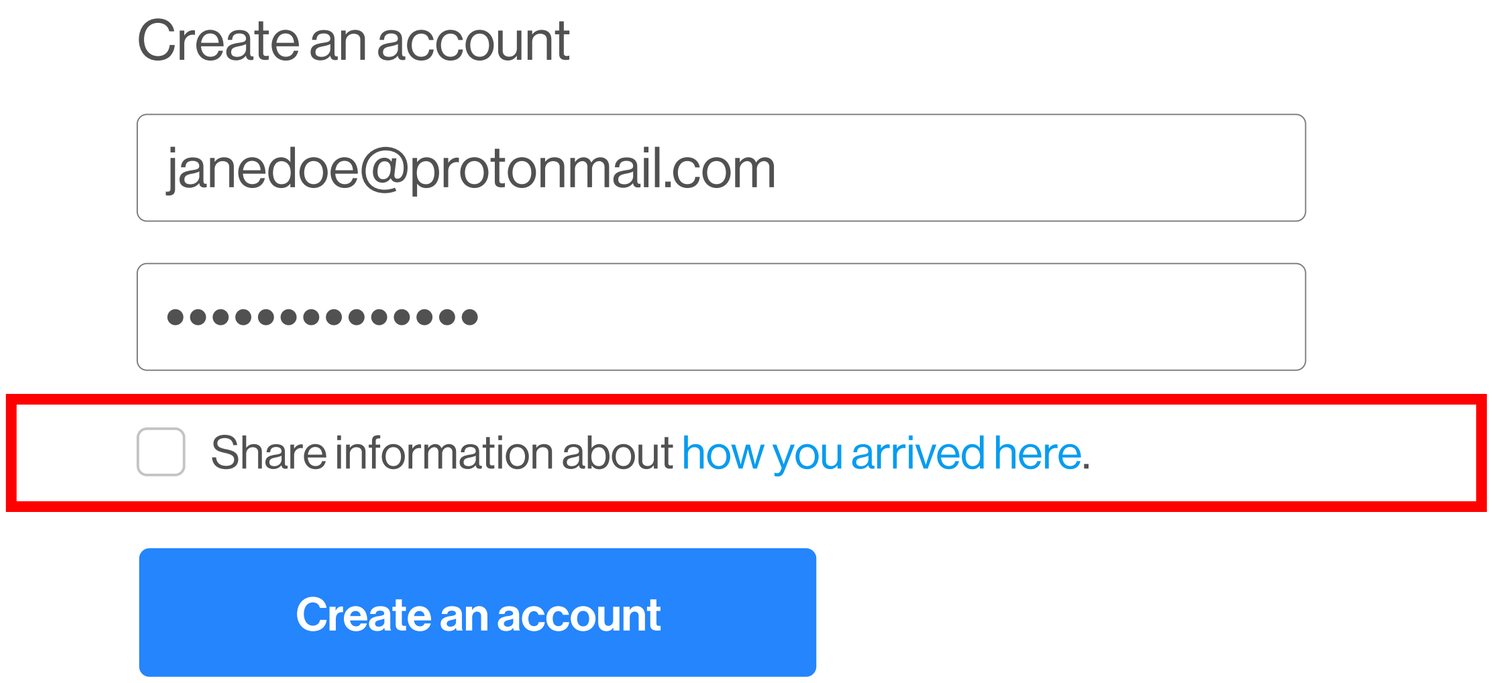
Our Use of Cookies
We use cookies, including third-party cookies, to improve your experience and to show you personalised content and advertising.

To find out more, read our Privacy Statement and Cookie Policy.

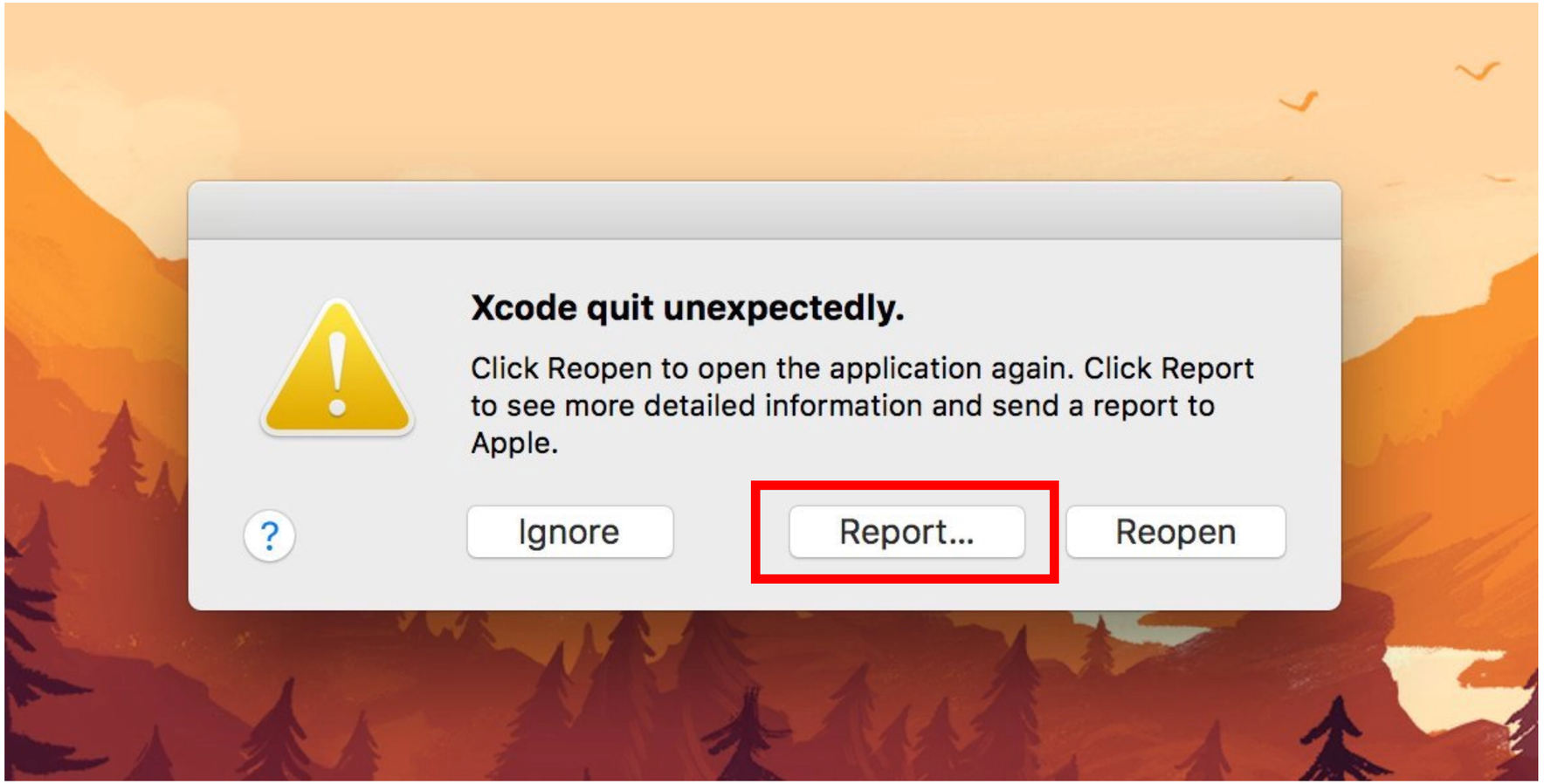
My Options

Accept All

To get past pop-ups, we need to gain the ability to ask for consent later in the user journey.



Pitch

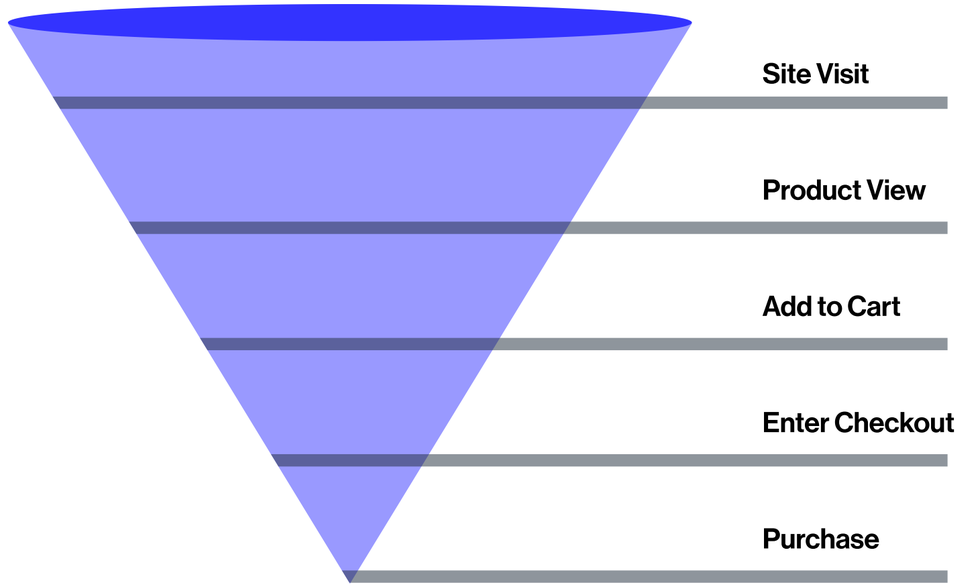

Xcode quit unexpectedly.
Click Reopen to open the application again. Click Report to see more detailed information and send a report to Apple.
Ignore
Report...
Reopen

Pitch

To do so, we need local tracking.

Begin **tracking locally,** but **don't upload**

Get consent and **upload**



Share information about how you arrived here.

Pitch

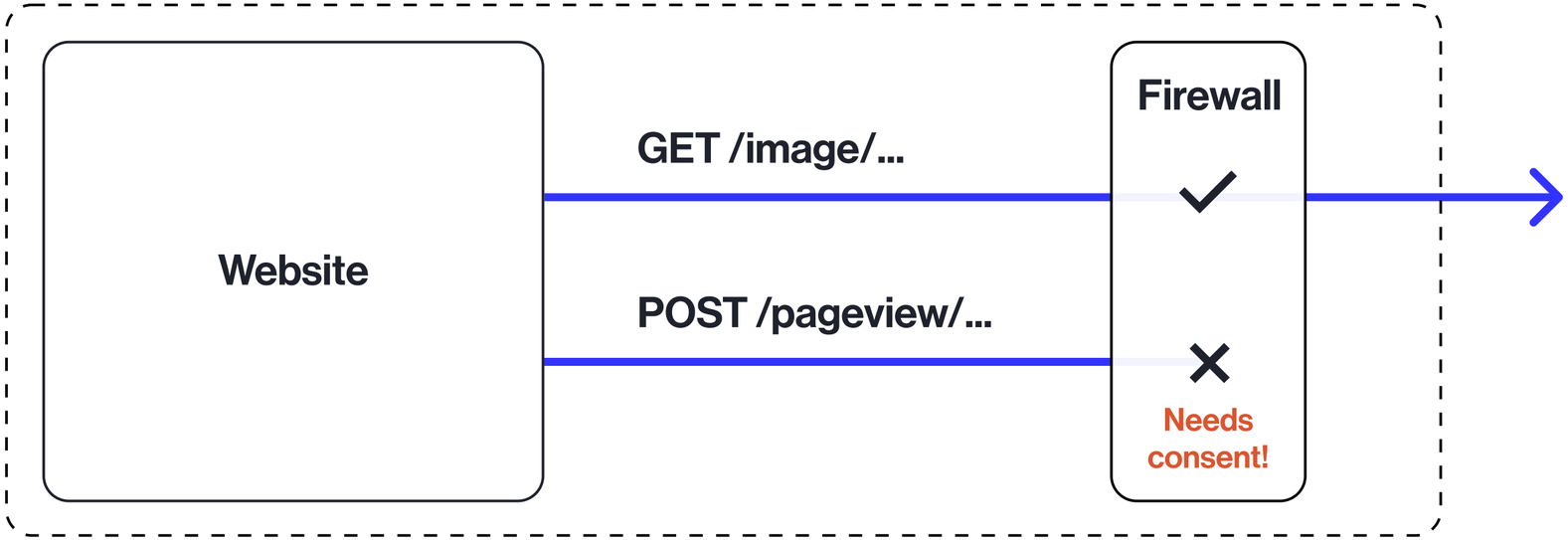
Switching to local tracking is hard

- Most analytics happen through imported third-party scripts, like Google Analytics.
- When imported, these scripts begin **tracking** *and* **uploading**
- It's very difficult to alter the functionality of third-party scripts.

How can consent managers block trackers from uploading data?

Pitch

What if there were a firewall?



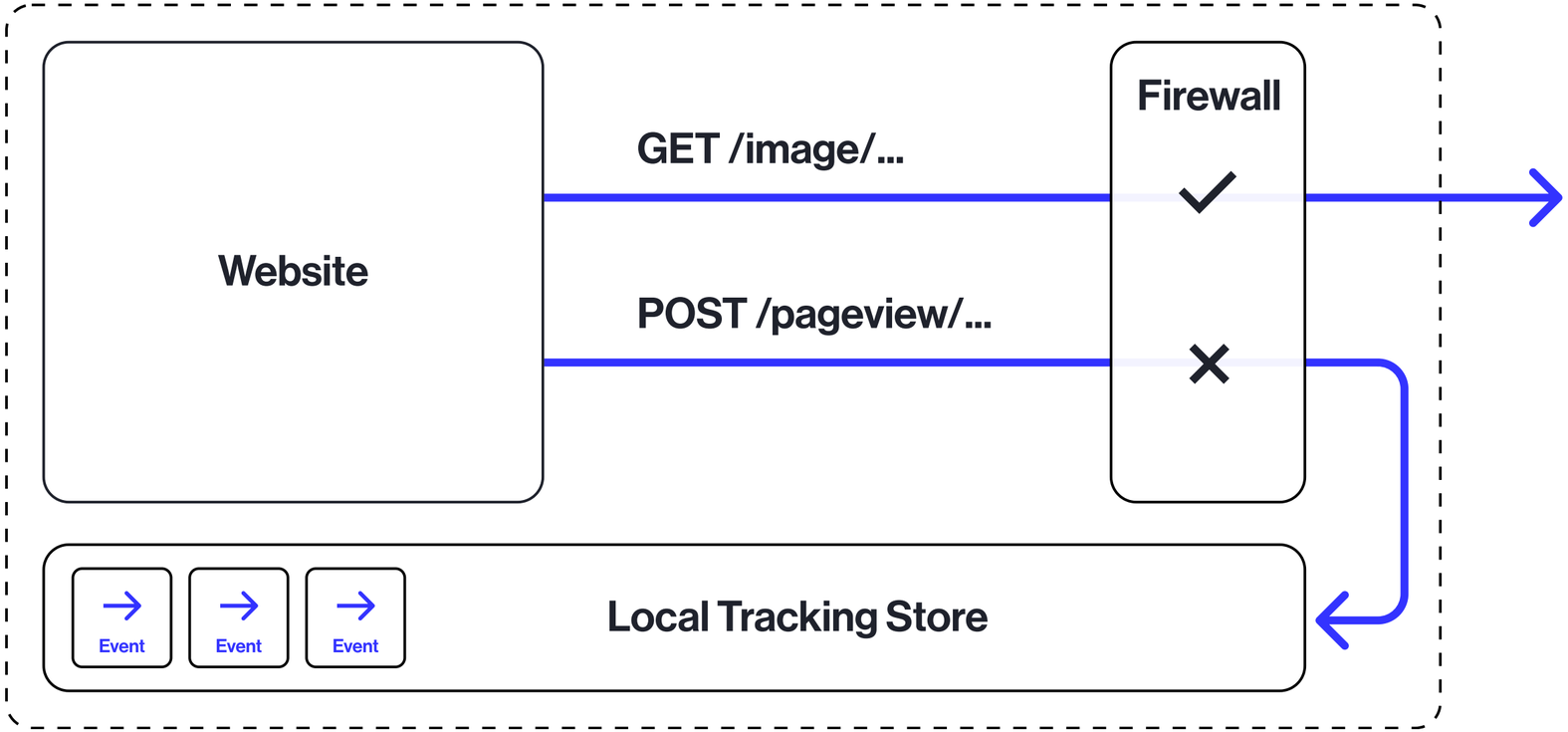
Governance rules could be set on all network traffic

Essential network traffic could be allowed through

Tracking could be blocked or allowed based on consent

Pitch

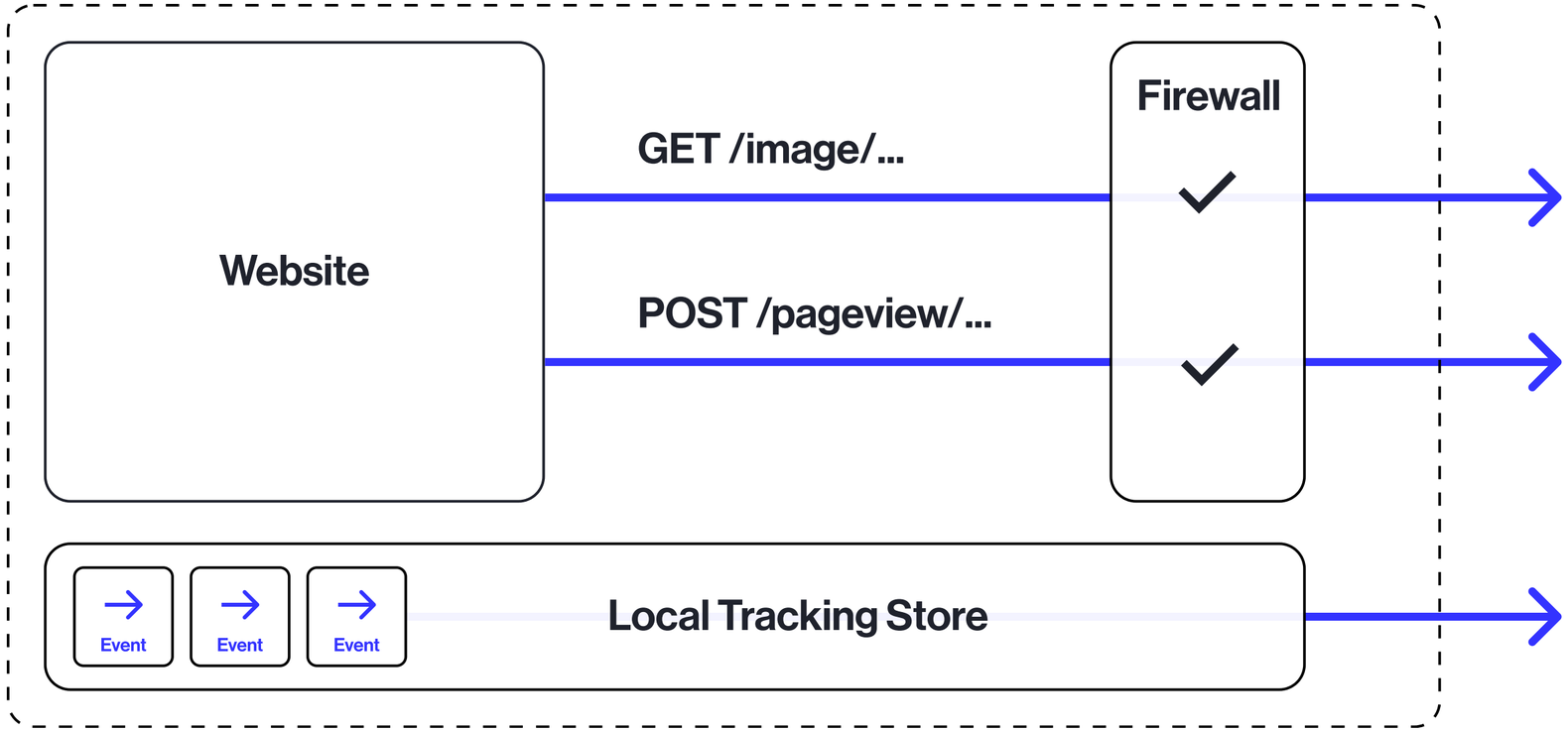
Quarantining Traffic



A firewall could locally store tracking events.

Pitch

Replaying Traffic



After consent is given, locally stored events could be uploaded, and new tracking let through.

Share information about how you arrived here.

Pitch

Building a Firewall in the Browser:
An Engineer's Journey

Pitch

Idea 1

Sandboxed iframe Documents

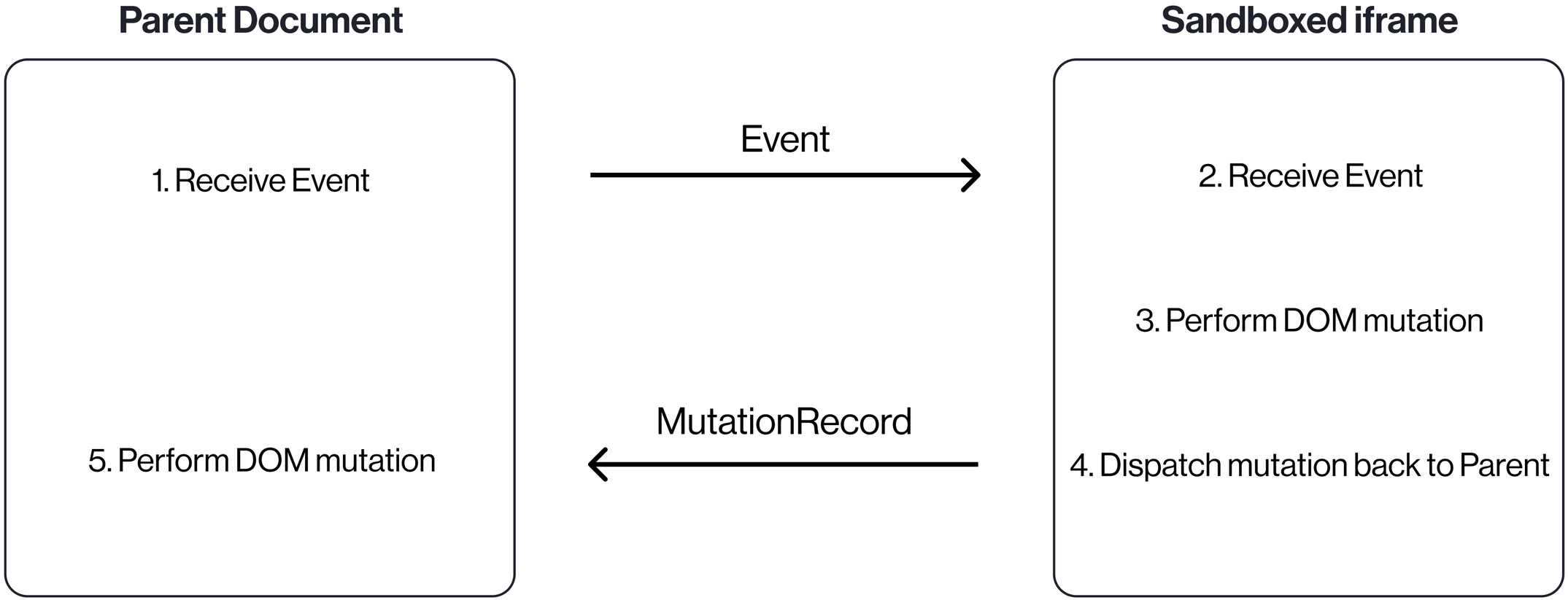
Parent Document

```
<html>
  <head><title>My Website</title></head>
  ...
  <script src="//example.com/analytics.js" />
  ...
</html>
```

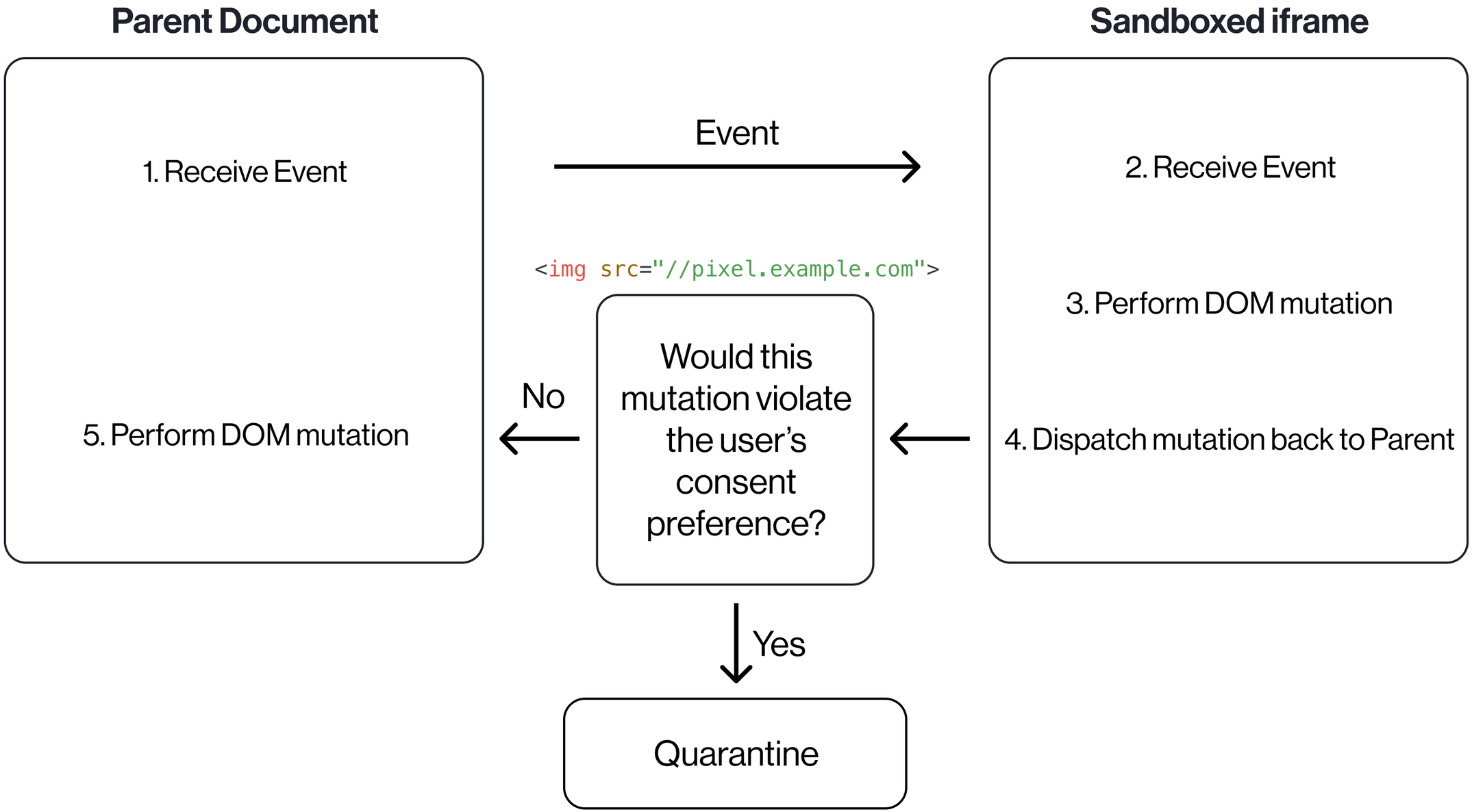
Sandboxed iframe with imported JS

```
<html>
  <head><title>My Website</title></head>
  ...
  <iframe sandbox>
    #document
    <html>
      <head><title>My Website</title></head>
      ...
      <script>
        const tracker = document.createElement('img');
        tracker.src = '//pixel.example.com';
        document.body.appendChild(tracker);
      </script>
      ...
    </html>
  </iframe>
  ...
</html>
```

Pitch


Parent Document
Sandboxed iframe
1. Receive Event
Event
2. Receive Event
3. Perform DOM mutation
MutationRecord
5. Perform DOM mutation
4. Dispatch mutation back to Parent

Pitch


Parent Document
Sandboxed iframe
1. Receive Event
Event
2. Receive Event
<img src="//pixel.example.com">
3. Perform DOM mutation
Would this mutation violate the user's consent preference?
No
4. Dispatch mutation back to Parent
5. Perform DOM mutation
Yes
Quarantine

Pitch

Results

Sandboxed iframe Documents

- Simple implementation that proved this can work
- High CPU demand, scalability problems
- Mis-ordered replays could cause site functionality breakage
- <script async> and <script defer> issues

Pitch

Idea 2

Dynamic Content Security Policies

```
<html>
  <head>
    <title>My Website</title>
    <meta
      http-equiv="Content-Security-Policy"
      content="default-src 'self' allowed-url.com 'unsafe-inline' 'unsafe-eval'"
    >
  </head>
  ...

  ...
</html>
```

This allows network requests to:

- /any/path/on/this/url
- allowed-url.com

Pitch

CSP when opted out

```
<html>
  <head>
    <title>My Website</title>
    <meta
      http-equiv="Content-Security-Policy"
      content="default-src 'self' data: blob: *.imgix.net js.intercomcdn.com wss://*.intercom.io ..."
    >
  </head>
  ...

  ...
</html>
```

Only allow essential URLs

CSP when opted in

```
<html>
  <head>
    <title>My Website</title>
    <meta
      http-equiv="Content-Security-Policy"
      content="default-src 'self' data: blob: *.imgix.net js.intercomcdn.com wss://*.intercom.io px.ads.linkedin.com analytics.google.com ..."
    >
  </head>
  ...

  ...
</html>
```

Allow tracking URLs too

Pitch

Results

Dynamic Content Security Policies

- Highly effective at blocking tracking attempts
- Doesn't offer an easy way to locally quarantine traffic
  - HTTP POST body data lost
  - Limited metadata from blocked events
- On a given page-load, a CSP can only be updated to become more strict
  - Opting in would require a page reload

What should the default behavior be for unknown network requests?

- We don't know if it's for tracking or essential purposes.
- Should be up to the site owner

Pitch

Idea 3

Patchers and Virtual DOM

Patch the prototype of each JavaScript API and Element which may generate a network request.

Easy part: *Fetch, XHR, Workers, WebSockets, navigator.sendBeacon, and more*

```
fetch('http://example.com/movies.json');
```

Hard part: HTML DOM Elements such as images often cause network requests too

```
<img src="//pixel.example.com/tracker.png">
```

Pitch

Example

Patching **Image.src**

We patch Image.src ...

```
const nativeImageSrc = Object.getOwnPropertyDescriptor(
  Image.prototype,
  'src'
);

Object.defineProperty(Image.prototype, 'src', {
  ...nativeImageSrc,
  set(url) {
    if (allowed) {
      return nativeImageSrc.set.call(this, url);
    } else {
      // quarantine mutation
    }
  }
});
```

Simplified example.

... so code like this can be regulated

```
const tracker = document.createElement('img');
tracker.src = '//pixel.example.com';
document.body.appendChild(tracker);
```

Exceptions

Patching **Image.src**

This network request is regulated by a patch to the constructor ...

```
const tracker = document.createElement('img');
tracker.src = '//pixel.example.com';
document.body.appendChild(tracker);
```

... but this one slips by!

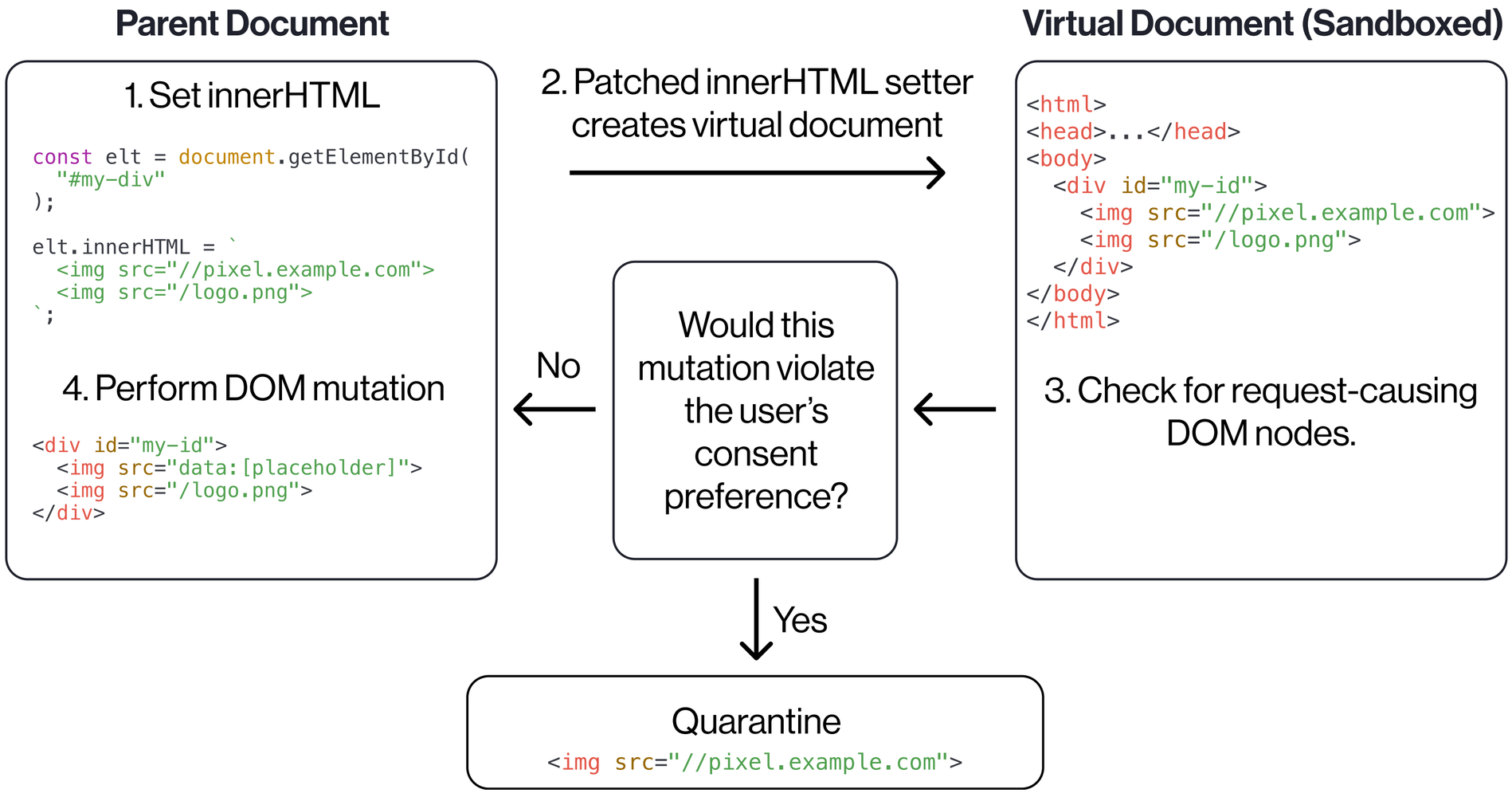
```
const elt = document.getElementById("#my-div");
elt.innerHTML = '<img src="//pixel.example.com">';
```

Because innerHTML does not use the Image constructor!

This is true for all DOM APIs that take HTML strings as input (outerHTML, etc.)

Pitch

Using Virtual Documents to patch innerHTML



Pitch

There's a lot to cover!

- Fetch, XHR, EventSource, Beacon, WebSocket
- <script> <style>
- prefetch, prerender, preconnect
- <iframe>
- <img>, <video>, <audio>
- <object>, <embed>, <applet>
- <a ping>
- @font-face
- WebWorker, SharedWorker, Service Workers
- WebTransport
- ... and many more!

Patchers, virtual documents, and sometimes CSP cover everything.

Pitch

Winner: Patchers and Virtual DOM

Blazing fast!

- No CPU strain; memory bloat
- Tested on heavy sites
  - Email clients
  - Video streaming platforms
  - Multi-user collaboration apps
  - Sites with dozens of trackers

Complete!

- Governs every type of network request.

Replay works!

- We can ask for consent when it makes sense.

Pitch

airgap.js

Policy-based data flow governance.

```
<script src="//cdn.transcend.io/cm/:id/airgap.js"></script>
```

Size: 33kb

Pitch

Thank you

Email

ben.brook@transcend.io

Twitter

@bencmbrook

@transcend_io

Website

https://transcend.io

Pitch