Feature Policy | Permission-Policy

We use lot of third party scripts / iFrame in our app. What if they try to access geolocation, audio, video, mic etc without our permission. How can we trust them ? We need to ensure security. For that we use Permissions-policy

Permissions Policy provides mechanisms for web developers to explicitly declare what functionality can and cannot be used on a website. You define a set of "policies" that restrict what APIs the site's code can access or modify the browser's default behavior for certain features. This allows you to enforce best practices, even as the codebase evolves — as well as more safely compose third-party content.

Permissions Policy is similar to Content Security Policy but controls features instead of security behavior.

Examples of what you can do with Permissions Policy:

- Change the default behavior of autoplay on mobile and third-party videos.
- Restrict a site from using sensitive devices like the camera, microphone, or speakers.
- Allow iframes to use the Fullscreen API.
- Stop items from being scripted if they are not visible in the viewport, to improve performance.

Permissions Policy provides two ways to specify policies:

- The `Permissions-Policy` HTTP header, to control feature usage in received responses and any embedded content within the page (which includes `<iframe>`s).

- The `<iframe>` `allow` attribute, to control feature usage only in specific `<iframe>`s.

Syntax

Permissions-Policy: <directive>=<allowlist>

Allowlists

An allowlist is a list of origins that takes one or more of the following values contained in parentheses, separated by spaces:

- `*`: The feature will be allowed in this document, and all nested browsing contexts (`<iframe>`s) regardless of their origin.
- `()` (empty allowlist): The feature is disabled in top-level and nested browsing contexts. The equivalent for `<iframe>` `allow` attributes is `'none'`.
- `self`: The feature will be allowed in this document, and in all nested browsing contexts (`<iframe>`s) in the same origin only. The feature is not allowed in cross-origin documents in nested browsing contexts. `self` can be considered shorthand for `https://your-site.example.com`. The equivalent for `<iframe>` `allow` attributes is `self`.
- `src`: The feature will be allowed in this `<iframe>`, as long as the document loaded into it comes from the same origin as the URL in its src attribute. This value is only used in the `<iframe>` `allow` attribute, and is the *default* `allowlist` value in `<iframe>`s.
- `"<origin>"`: The feature is allowed for specific origins (for example, `"https://a.example.com"`). Origins should be separated by spaces. Note that origins in `<iframe>` allow attributes are not quoted.

The values `*` and `()` may only be used on their own, while `self` and `src` may be used in combination with one or more origins.

Directives

accelerometer, battery, camera, microphone, geolocation etc etc

Example-

index js

```
const express = require('express');
const app = express();

app.get('/page', (req, res) => {
  res.send(`
  <!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
l-scale=1.0">
    <title>Fetch geolocation permission Example</title>
</head>
<body>
    <h1>Fetch geolocation permission Example</h1>
    <button onclick="getGeolocation()">Fetch Data</button>
    <div id="result"></div>

    <script>
        function getGeolocation() {
          // Check if the Geolocation API is supported by the
browser
          if (navigator.geolocation) {
            // Geolocation is supported
            navigator.geolocation.getCurrentPosition(
              function (position) {
                // Success callback - position object contain
s the user's location
                const latitude = position.coords.latitude;
                const longitude = position.coords.longitude;
```

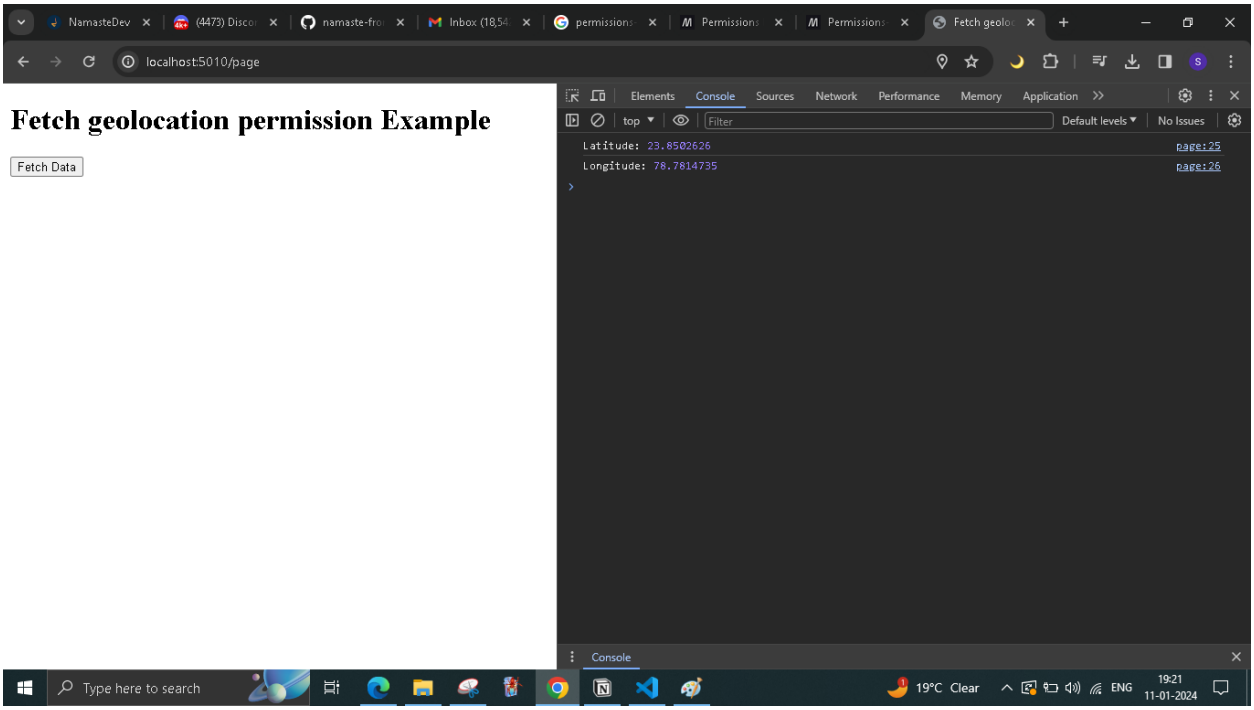
```
                console.log('Latitude:', latitude);
                console.log('Longitude:', longitude);
              },
              function (error) {
                // Error callback - handle errors
                switch (error.code) {
                  case error.PERMISSION_DENIED:
                    console.error('User denied the request fo
r Geolocation.');
                    break;
                  case error.POSITION_UNAVAILABLE:
                    console.error('Location information is un
available.');
                    break;
                  case error.TIMEOUT:
                    console.error('The request to get user lo
cation timed out.');
                    break;
                  case error.UNKNOWN_ERROR:
                    console.error('An unknown error occurre
d.');
                    break;
                }
              }
            );
          } else {
            // Geolocation is not supported by the browser
            console.error('Geolocation is not supported by th
is browser.');
          }

        }
    </script>
</body>
</html>
```

```
  `)
})

const port = process.env.PORT || 5010;
app.listen(port, () => {
  console.log(`Server is running on port ${port}`);
});
```

Imagine script we are returning is coming from third party



We clicked on fetch data, out geolocation(latitude and longitude) is consoled

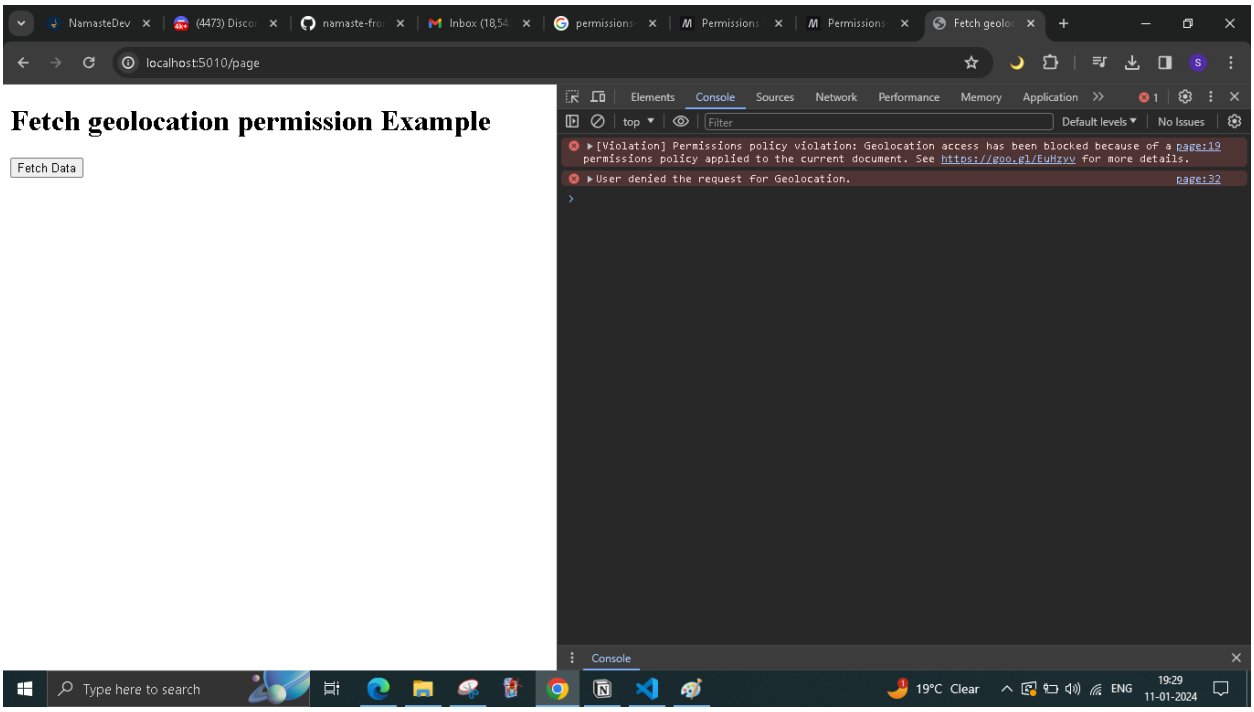
if we don’t want this, we need to set Permissions-Policy header

geolocation=() means access to it is disabled

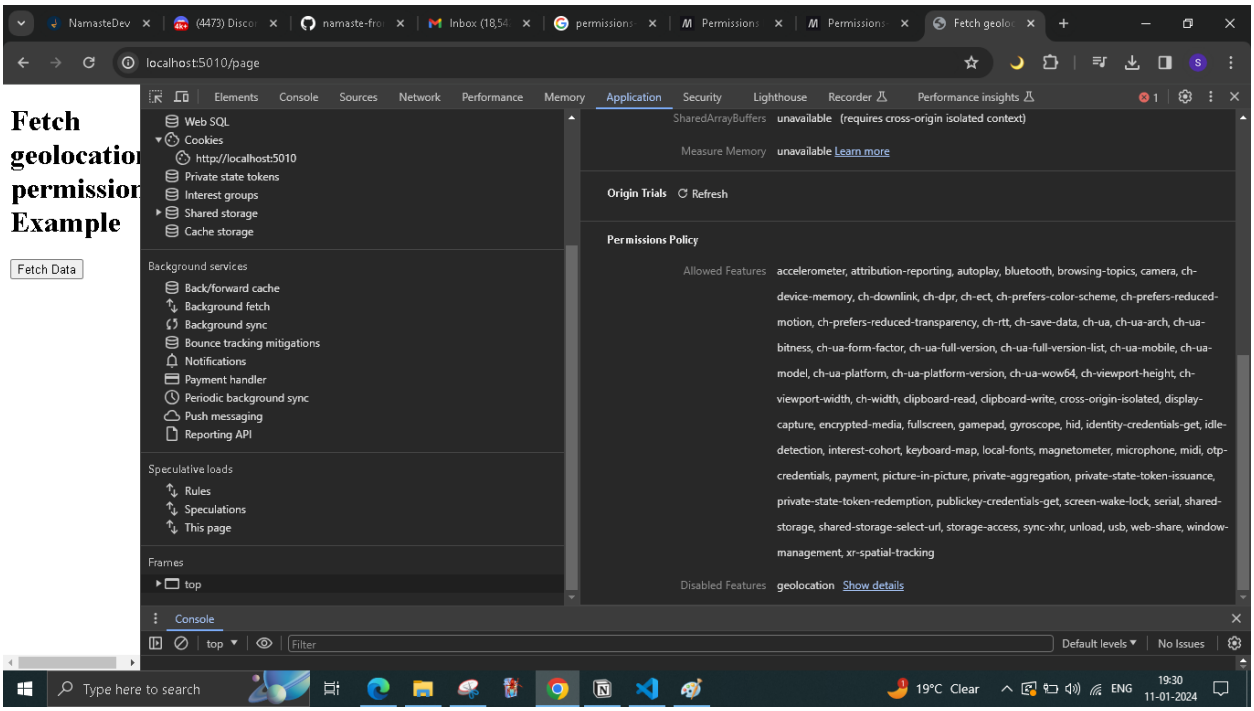
add below code in index js

```
app.use((req, res, next) => {
  res.setHeader('Permissions-Policy', 'geolocation=()');
  next();
})
```

Again we click on fetch data



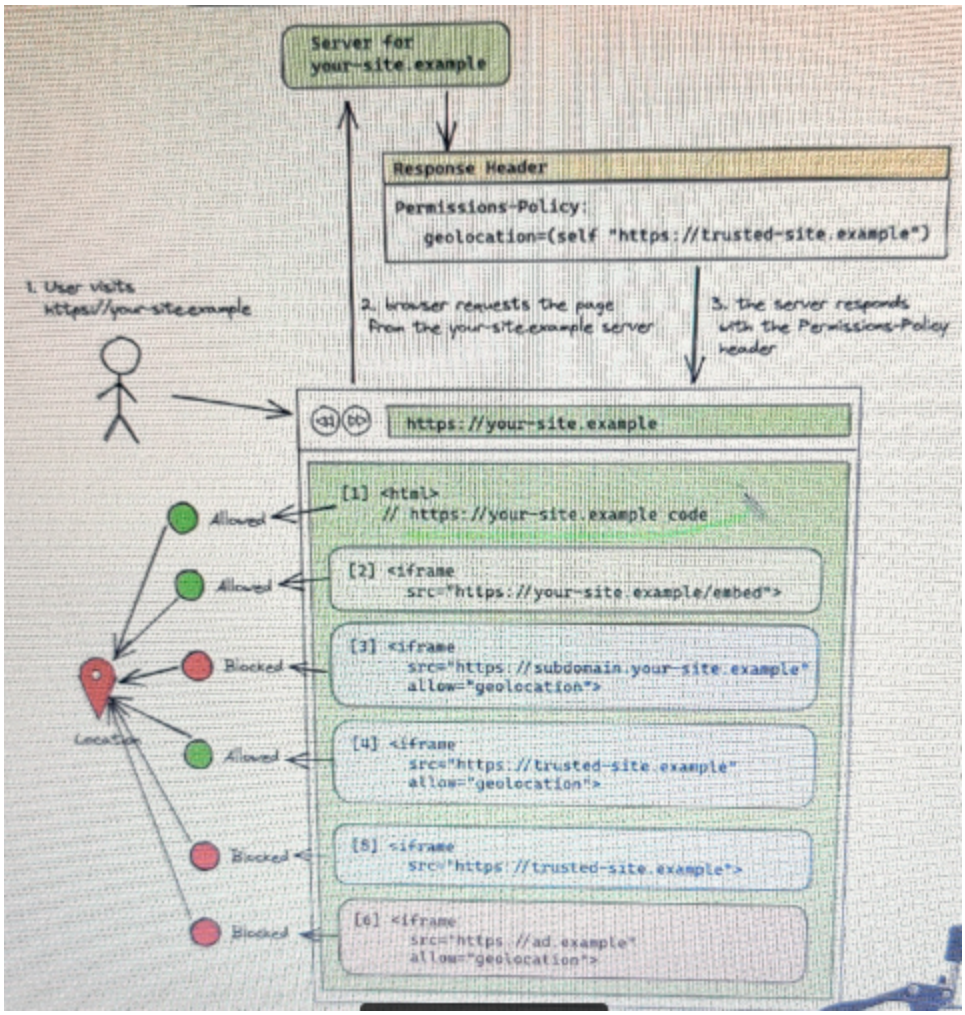
Where we can see all Permission policies allowed feature ?



Application ⇒ Sidebar(top under frames)

For more examples

https://permissions-policy-demo.glitch.me/


Server for
your-site.example
Response Header
Permissions-Policy:
geolocation=(self "https://trusted-site.example")
1. User visits
https://your-site.example
2. Browser requests the page
From the your-site.example server
3. The server responds
with the Permissions-Policy
header
https://your-site.example
[1] <html>
// https://your-site.example code
[2] <iframe
src="https://your-site.example/embed">
[3] <iframe
src="https://subdomain.your-site.example"
allow="geolocation">
[4] <iframe
src="https://trusted-site.example"
allow="geolocation">
[5] <iframe
src="https://trusted-site.example">
[6] <iframe
src="https://ad.example"
allow="geolocation">
Allowed
Allowed
Blocked
Allowed
Blocked
Blocked