Node.js: implementare la gestione dei cookie di terze parti

In questo tutorial vedremo come permettere agli utenti di negare il loro consenso all'uso di cookie di terze parti.

Per cominciare creiamo un semplice file JavaScript che simulerà la creazione di un cookie di terze parti.

```
'use strict';
(function() {

    const date = new Date();
    const daysToExpire = 365;
    const cookieName = 'testcookie';
    const cookieValue = 'test';

    date.setTime(date.getTime() + (daysToExpire * 24
* 60 * 60 * 1000));

    document.cookie = `${cookieName}=${cookieValue};
expires=${date.toGMTString()}`;
})();
```

La logica è questa: se l'utente acconsente all'uso dei cookie, inseriamo lo script nell'elemento `head` delle pagine, viceversa non lo inseriamo. Per memorizzare la scelta dell'utente useremo un nostro cookie tecnico oppure potremmo usare il web storage ma in questo caso non saremo in grado di gestirne la durata e l'utente avrà più difficoltà a gestirlo nel browser.

Scriviamo quindi:

```
<head>
<% if(cookieAccepted === 1) { %>
    <script src="/public/js/cookie.js"></script>
    <% } %>
</head>
```

L'inserimento del banner di scelta seguirà la stessa logica:

```
<% if(cookieAccepted === null) { %>

<div id="cookie-policy-banner">
    <p>
        Lorem ipsum dolor sit amet ete ideo autem
aquis ut meus est mos.
    </p>
    <div>
        <button type="button" id="reject-cookie-
policy" data-choice="0">No thanks</button>
        <button type="button" id="accept-cookie-
policy" data-choice="1">Accept</button>
    </div>
</div>
<% } %>
```

Il banner verrà inserito solo se l'utente non ha ancora effettuato una scelta. Gli attributi di dati dei pulsanti serviranno ad effettuare l'operazione via AJAX (di seguito useremo jQuery). Quindi implementiamo la creazione del cookie che memorizzerà la scelta dell'utente utilizzando il middleware cookie-parser:

```
'use strict';
```

```
const express = require('express');
const router = express.Router();


router.get('/', (req, res, next) => {
    const cookieAccepted = req.cookies.cookieAccepted
? parseInt(req.cookies.cookieAccepted, 10) : null;
    res.render('index', { cookieAccepted });
});

router.post('/setcookie', (req, res, next) => {
    const { choice } = req.body;
    const value = parseInt(choice, 10);
    const allowedValues = [0, 1];

    if(!allowedValues.includes(value)) {
        return res.sendStatus(403);
    }


    res.cookie('cookieAccepted', value, { maxAge: 60
* 60 * 24 * 365 * 1000 } ).json({ value });
});

module.exports = router;
```

Noterete che entrambi i cookie hanno una scadenza di 1 anno. Questo valore è usato in modo puramente indicativo e si può adattarlo alle varie esigenze della vostra policy.

Non ci resta che implementare la richiesta in AJAX:

```
"use strict";
```

```
$(function() {
   var $cookieBanner = $( "#cookie-policy-banner" );
   if( $cookieBanner.length ) {
       $cookieBanner.find( "button"
).click(function() {
            var choice = $( this ).data( "choice" );
            $.post( "/setcookie", { choice: choice },
function( res ) {
                if( $.isNumeric( res.value ) ) {
                    window.location = location.href;
                }
            });
       });
   }
});
```

Qui ci limitiamo a ricaricare la pagina corrente in modo da rendere immediatamente effettive le modifiche apportate.