<WA1/>

2020

CORS

Cross-Origin Resource Sharing

Enrico Masala
Fulvio Corno
Luigi De Russis

Goal

- What is an origin
- Cross-origin requests
- Why using CORS
- How CORS works
- How to enable CORS in Express

Mozilla Developer Network:
Web technology for developers —
HTTP — Cross-Origin Resource Sharing (CORS)
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Accessing multiple websites

CROSS-ORIGIN REQUEST SHARING

Loading a Web Page

- Loading a web page requires to load external resources (images, CSS, JS)
- They (JS, CSS) can, in turn, load other resources and generate network requests (asynchronous JS requests – XHR, fetch)
- For security reasons, JS runs in the browser sandbox
- Network access for JS code is, by default, limited to the **same origin**
- An **origin** consists of a **URI scheme**, **domain** and **port number**:

  http://**example.com**:**3456**/example/

Same-Origin Policy (SOP)

- Access only same URI scheme, domain and port number of the initial page

`http://normal-website.com/example/example.html`

URL accessed	Access permitted?
`http://normal-website.com/example/`	Yes: same scheme, domain, and port
`http://normal-website.com/example2/`	Yes: same scheme, domain, and port
`https://normal-website.com/example/`	No: different scheme and port
`http://en.normal-website.com/example/`	No: different domain
`http://www.normal-website.com/example/`	No: different domain
`http://normal-website.com:8080/example/`	No: different port*

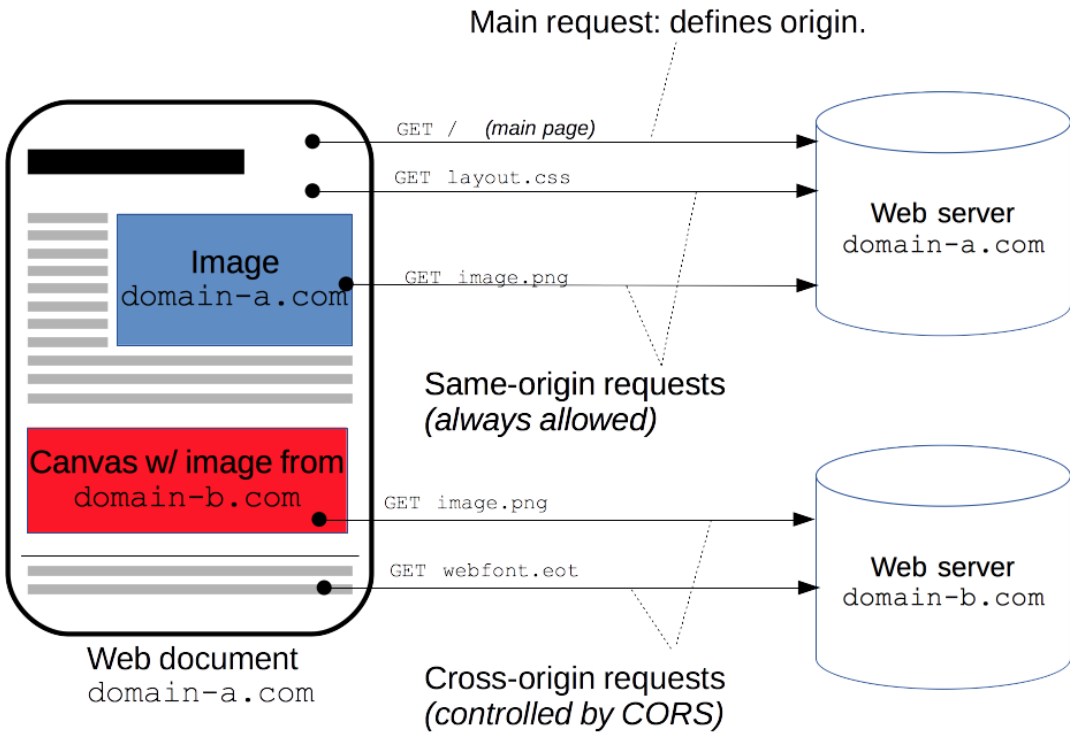
https://portswigger.net/web-security/cors/same-origin-policy

Cross-Origin Risks

- Loading page resources (images, CSS, JS) from different origins (cross-origin requests) without restrictions is a huge security risk
  - Browser always sends any cookies relevant to the domain with any request
  - If valid authentication/session cookies are sent, a request could operate from one origin (**example.com**) but as authenticated in another (**bank.com**)
- However, sometimes it is useful to load resources from other origins
  - Other subdomains/ports of the original one, e.g., static-content.example.com
  - Other domains: content delivery networks (common libraries, etc.), public services information (weather, news, stock values, etc.), content provided by third parties (advertisement, etc.)
  - REST API servers, in your network (but different server) or publicly accessible

Solving the Cross-Origin Problem: CORS

- Cross-Origin Resource Sharing (**CORS**): a **standard** mechanism to implement cross-domain requests
- CORS defines a **set of HTTP headers** that allow the browser and server to communicate about which requests are (or are not) allowed
- The **server** defines which origins are accepted for any request



https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

https://fetch.spec.whatwg.org/#http-cors-protocol

CORS in Practice

- Many modern websites use CORS to allow the page resources (e.g., scripts) access subdomains and *trusted* third parties
- Need careful implementation and attention in configuration to avoid exploitable vulnerabilities

- NB: This is **not** a way to address security issues such as Cross-Site Request Forgery (CSRF), Cross-Site Scripting (XSS), etc.

CORS Main Headers

- CORS *requests* must include the origin via a specific header:

  `Origin: https://foo.example`

- *Response* includes which origins can do the request:

  `Access-Control-Allow-Origin: https://foo.example`

- If the two match, the **browser** allows the script to access the response, otherwise the content appears to have failed to load from the script

- Any origin can also be allowed (e.g., publicly accessible services):

  `Access-Control-Allow-Origin: *`

https://fetch.spec.whatwg.org/#http-cors-protocol

CORS with Authentication

- By default, fetch requests do not send credentials (e.g., cookies)
- If needed, `fetch` has an option in the `init` object to include them
- Values: `'omit'` (default), `'same origin'` (send only in requests to the same origin), `'include'`

```
fetch('https://example.com', {
  credentials: 'include'
});
```

CORS Preflight Requests

- CORS requests might be preceded by means of an initial (HTTP method) OPTION request to determine if the actual request is safe to send
  - Example: with special headers
- Such cross-site requests are said "*preflighted*". This is done since actual requests may have implications to user data (sending *private information* etc.)
- This is typically **done automatically** by the browser
  - Need to know because it might impact application performance

https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

CORS Preflight example

```
OPTIONS /the/resource/you/request
Access-Control-Request-Method: POST
Access-Control-Request-Headers: origin, x-requested-with, accept
Origin: https://your-origin.com
```

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://your-origin.com
Access-Control-Allow-Methods: POST, GET, OPTIONS, DELETE
```

https://flaviocopes.com/express-cors/

Loading scripts from other origins

- Scripts loaded via `<script>` tag from other origins run with the same privileges of the other scripts in the web application
- Only load scripts you trust, and always include integrity check to prevent malicious code injection
  - With integrity attribute, the browser additionally check the resource using CORS, to ensure the origin serving the resource allows it to be shared with the requesting origin (i.e., server must have used: `Access-Control-Allow-Origin: *`)

```
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"
integrity="sha384-wfSDF2E50Y2D1uUdj0O3uMBJnjuUD4Ih7YwaYd1iqfktj0Uod8GCExl3Og8ifwB6"
crossorigin="anonymous"></script>
```

Loading modules via script tag

- Modules loaded from file system via `<script>` tag have **origin `null`** so browser prevents modules to be loaded **even from local file system** (file:// URI)
- Solution: serve content from a (local) web server

```
<body>
 ...
 <script type="module" src="main.js"></script>
 <script type="module" src="index.js"></script>
</body>
```

```
// In index.js:
import * as jsdom from 'main.js';
```

https://github.com/expressjs/cors

https://flaviocopes.com/express-cors/

Controlling Allowed Origins in your API Server

CORS ON THE SERVER SIDE

CORS on the server side

- Requires careful server configuration
- In express.js: careful when REST server is on different ports than the main Web Application
- NB: CORS DOES NOT APPLY when making requests outside browsers (e.g., curl, wget, REST browsers, etc.)
  - origin = null for external tools

Enabling CORS on Express application

- Use the middleware `cors`
  - http://expressjs.com/en/resources/middleware/cors.html
  - `npm install cors`

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

app.use(cors()) ;
```

Simple Usage

- Enable **All CORS** Requests (for this server)

```
app.use(cors())
```

- Enable CORS for a **Single Route**

```
app.get('/products/:id', cors(), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for a Single Route'})
})
```

- By default, all origins will be enabled for all HTTP methods

Configuration options

- The `cors(options)` call accepts configuration instructions
- Specify the allowed origins (as a string, function, regexp, array)
- Specify the allowed methods
- Fine-tune allowed headers and credentials

Default configuration options

```
{
  "origin": "*",
  "methods": "GET,HEAD,PUT,PATCH,POST,DELETE",
  "preflightContinue": false,
  "optionsSuccessStatus": 204
}
```

Enabling preflight requests

- Must define a route for OPTION, for the routes that require a successful preflight call

```
app.options('/products/:id', cors());
// enable pre-flight request
```

- May enable globally for all routes

```
app.options('*', cors());
// NOTE: include before other routes
```

http://expressjs.com/en/resources/middleware/cors.html#enabling-cors-pre-flight

References

- A tutorial on CORS
  - https://auth0.com/blog/cors-tutorial-a-guide-to-cross-origin-resource-sharing/
- https://en.wikipedia.org/wiki/Cross-origin_resource_sharing
- https://github.com/expressjs/cors
- https://flaviocopes.com/express-cors/

License

- These slides are distributed under a Creative Commons license “**Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)**”
- **You are free to:**
  - **Share** — copy and redistribute the material in any medium or format
  - **Adapt** — remix, transform, and build upon the material
  - The licensor cannot revoke these freedoms as long as you follow the license terms.
- **Under the following terms:**
  - **Attribution** — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
  - **NonCommercial** — You may not use the material for commercial purposes.
  - **ShareAlike** — If you remix, transform, or build upon the material, you must distribute your contributions under the same license as the original.
  - **No additional restrictions** — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.
- https://creativecommons.org/licenses/by-nc-sa/4.0/