

## Same Origin Test

### Test Plan

For probing the fact that browser does not see the ip address and the corresponding domain name as the same origin for a web page, we will test the localhost and the corresponding ip address (127.0.0.1). The page that we will access will try to make a request for each domains. The request will try to access the `test.json`'s page content as a result. The expected behaviour is that the request that corresponds with the domain of the page accessed should pass and to show in the console the content requested. The request to the other domain should throw a CORS error even if they have the same origin.

For this we will need the following:

- a server that runs locally (We will use a Node.js server). The server's code is stored in `server.js` file

```
const express = require('express');
const app = express();
app.use(express.static(__dirname));
app.listen(8000, () => console.log('Server running on http://localhost:8000'));
```

the server can be run with this command in the same folder as the file:

```
node server.js
```

- a file that can be accessed and that should make the requests once it is checked. The code is in the file named `same-origin-test.html`

this is the code that runs in this page:

```
// same-origin-test.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Same Origin Test</title>
</head>
<body>
  <h1>Same-Origin Policy Test</h1>
  <p>Open the console to see the results.</p>
  <script>
    // Define the localhost and 127.0.0.1 versions of the same server
    const localhostUrl = 'http://localhost:8000/test.json';
    const loopbackUrl = 'http://127.0.0.1:8000/test.json';

    // Fetch from localhost
    fetch(localhostUrl)
```

```

        .then(response => response.json())
        .then(data => console.log('Localhost response:', data))
        .catch(error => console.error('Localhost fetch error:', error));

// Fetch from 127.0.0.1
fetch(loopbackUrl)
    .then(response => response.json())
    .then(data => console.log('127.0.0.1 response:', data))
    .catch(error => console.error('127.0.0.1 fetch error:', error));
</script>
</body>
</html>

```

- finally we will need the content to be requested (from the `test.json`):

```

{
  "message": "This is a test JSON file."
}

```

## Tests results

- Accessing `localhost:8000/same-origin-test.html`

The screenshot shows a web browser window with the address bar set to `localhost:8000/same-origin-test.html`. The page content includes a heading "Same-Origin Policy Test" and instructions to "Open the console to see the results." and "The request to the '127.0.0.1' fails with a CORS error". The browser's developer tools are open, showing the Network tab with a list of requests. A red box highlights a failed request to `127.0.0.1:8000/test.json` with status 404 and error "CORS Missing Allow Origin". The console shows a message: "The request that corresponds to the domain accessed succeeded, showing in the console the content from the 'test.json' file".

Similar results can be seen accessing the ip address:

The screenshot shows a web browser window with the address bar set to `127.0.0.1:8000/same-origin-test.html`. The page content includes a heading "Same-Origin Policy Test" and instructions to "Open the console to see the results." and "The request to the '127.0.0.1' fails with a CORS error". The browser's developer tools are open, showing the Network tab with a list of requests. A red box highlights a failed request to `127.0.0.1:8000/test.json` with status 404 and error "CORS Missing Allow Origin". The console shows a message: "The request that corresponds to the domain accessed succeeded, showing in the console the content from the 'test.json' file".