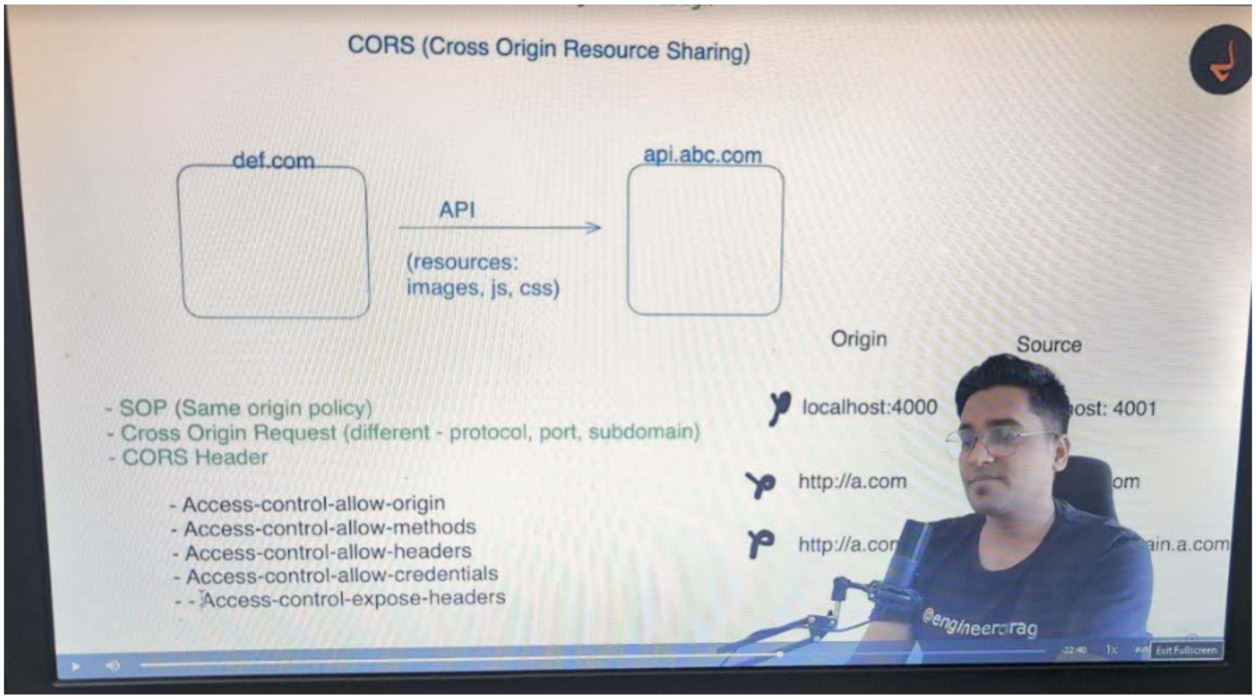
Cross-Origin Resource Sharing(CORS)

Browser comes up with certain security measures which says that if you want to access some resource cross domain, then I am going to ensure that other domain from which you want to access a resource is giving permission to access or not.

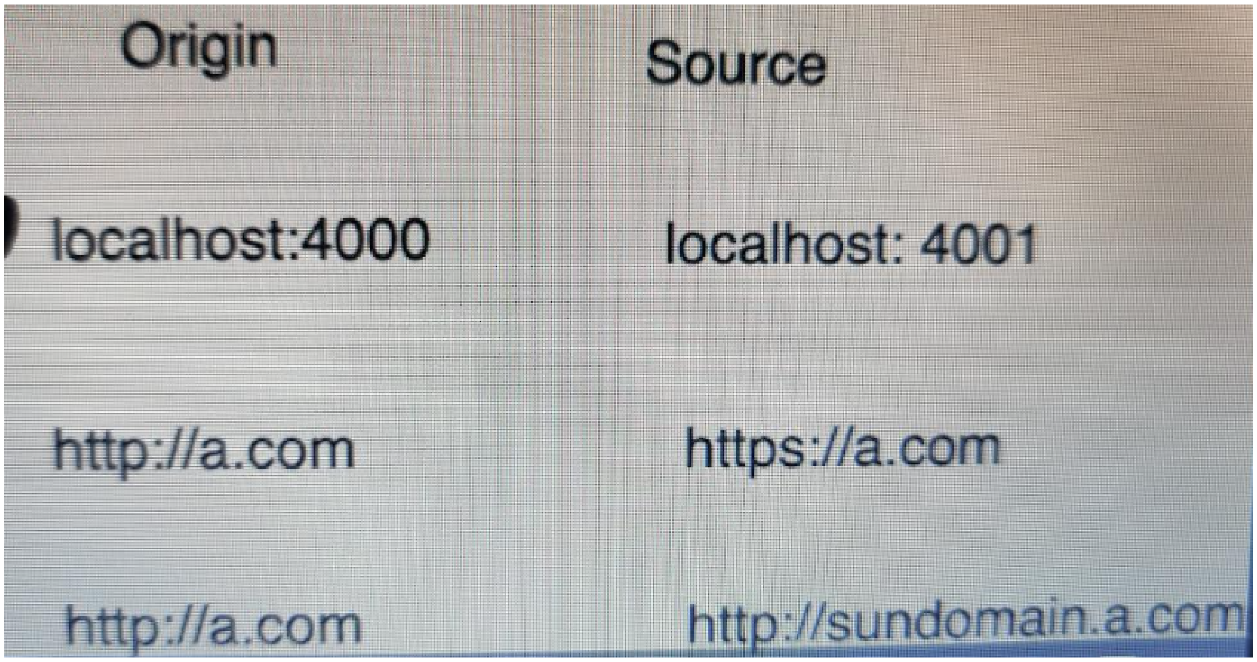
Consider example - we have two domain

def.com wants to access data/resources from api.abc.com . Even though abc.com has public api, browser is going to check whether abc has given permission to access resources or not.


CORS (Cross Origin Resource Sharing)
def.com
API
(resources: images, js, css)
api.abc.com
@engineerdrag
Exit Fullscreen

- SOP (Same origin policy)
- Cross Origin Request (different - protocol, port, subdomain)
- CORS Header
    - Access-control-allow-origin
    - Access-control-allow-methods
    - Access-control-allow-headers
    - Access-control-allow-credentials
    - - Access-control-expose-headers

Origin	Source
localhost:4000	ost: 4001
http://a.com	om
http://a.co	ain.a.com



Origin	Source
localhost:4000	localhost: 4001
http://a.com	https://a.com
http://a.com	http://sundomain.a.com

By default, browser enables same origin policy which means you can access anything which is on same domain but not on other domain.

Cross origin request(different protocol, different port, different domain, different subdomain)

different protocol - http://a.com cannot access https://a.com

different port - 4000 ⇒ 4001

different subdomain ⇒ a.com ⇒ subdomain.a.com

different domain ⇒ a.com ⇒ b.com

CORS Header

Access-control-allow-origin

Access-control-allow-methods

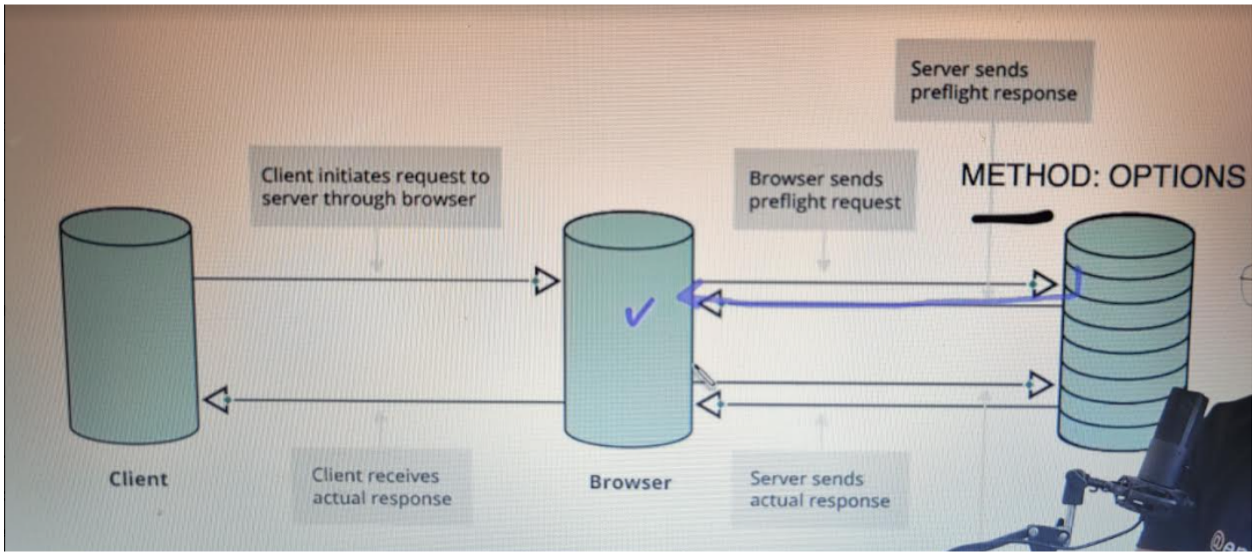
Access-control-allow-headers

Access-control-allow-credentials

Access-control-expose-headers

Read more about headers here

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


Server sends preflight response
Client initiates request to server through browser
Browser sends preflight request
METHOD: OPTIONS
Client
Client receives actual response
Browser
Server sends actual response

Client(javascript code-axios request) tries to access some cross domain api. It initiates request to server through browser. Browser sees it is a cross domain request.

So, browser sends preflight request to server. Preflight request is sent with METHOD:OPTIONS to server. If server says cross domain is request is not allowed to the browser(preflight response), request will be terminated and browser won't even send original request to the server.

If server allows cross domain request, it will set appropriate headers(example-access-control-allow-origin) in its prefilght response to the browser. Browser then sends the original request to the server and server sends actual response in return.

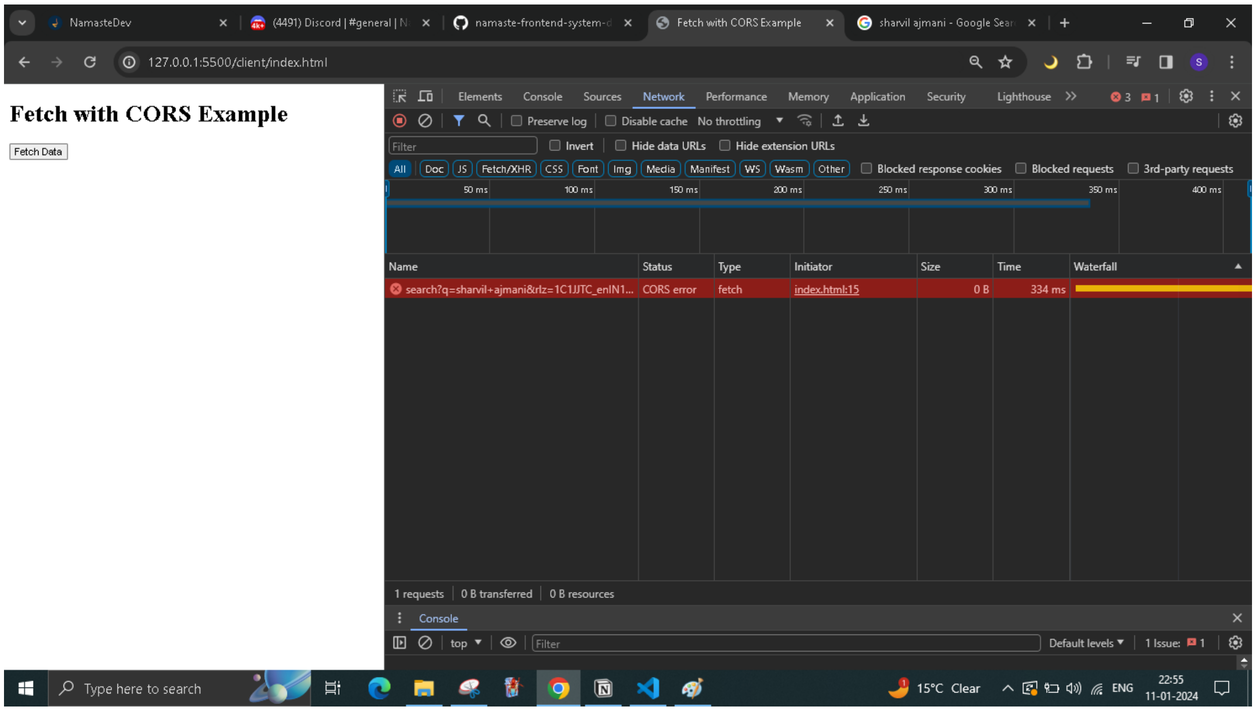
Example 1

let us try to access below google search api of name through our code

index html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
l-scale=1.0">
    <title>Fetch with CORS Example</title>
</head>
<body>
    <h1>Fetch with CORS Example</h1>
    <button onclick="fetchData()">Fetch Data</button>
    <div id="result"></div>

    <script>
        function fetchData() {
            fetch('https://www.google.com/search?q=sharvil+aj
mani&rlz=1C1JJTC_enIN1017IN1017&oq=sharvil+ajmani&gs_lcrp=EgZ
jaHJvbWUyBggAEEUYOTIGCAEQRRg8MgYIAhBFGD3SAQoxNzMzOWowajE1qAIA
sAIA&sourceid=chrome&ie=UTF-8', {
                method: 'GET',
            })
            .then(response => response.json())
            .then(data => {
                // Display the fetched data
                document.getElementById('result').innerText =
JSON.stringify(data, null, 2);
```

```
            })
            .catch(error => {
                console.error('Error:', error);
            });
        }
    </script>
</body>
</html>
```

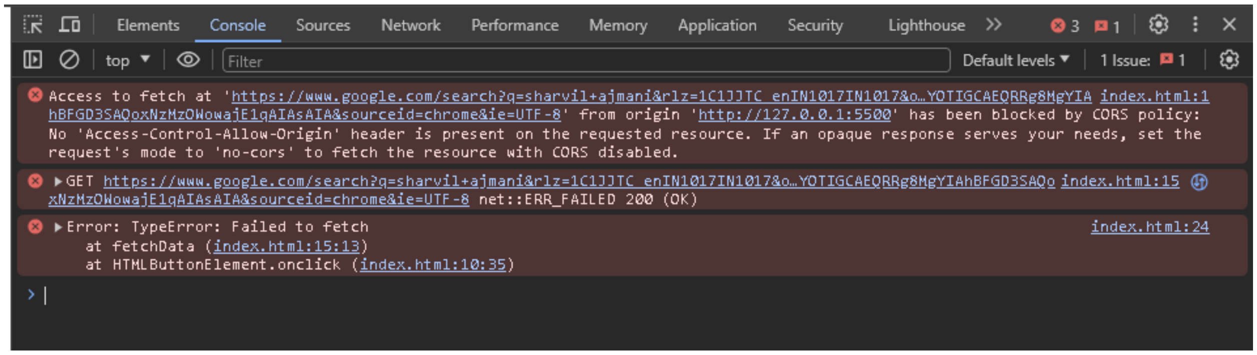


127.0.0.1:5500/client/index.html

Fetch with CORS Example

Name	Status	Type	Initiator	Size	Time
search?q=sharvil+ajmani&rlz=1C1JJTC_enIN1...	CORS error	fetch	index.html:15	0 B	334 ms

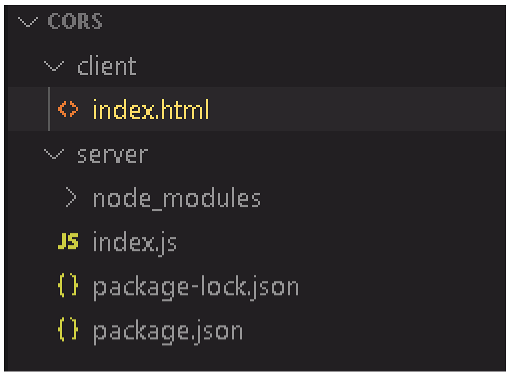
1 requests | 0 B transferred | 0 B resources



```
Access to fetch at 'https://www.google.com/search?q=sharvil+ajmani&rlz=1C1JJTC_enIN1017IN1017&o…YOTIGCAEQRRg8MgYIA index.html:1
hBFGD3SAQoxNzMzOWowajE1qAIAsAIA&sourceid=chrome&ie=UTF-8' from origin 'http://127.0.0.1:5500' has been blocked by CORS policy:
No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs, set the
request's mode to 'no-cors' to fetch the resource with CORS disabled.
▶GET https://www.google.com/search?q=sharvil+ajmani&rlz=1C1JJTC_enIN1017IN1017&o…YOTIGCAEQRRg8MgYIAhBFGD3SAQo index.html:15
xNzMzOWowajE1qAIAsAIA&sourceid=chrome&ie=UTF-8 net::ERR_FAILED 200 (OK)
▶Error: TypeError: Failed to fetch                                                                              index.html:24
    at fetchData (index.html:15:13)
    at HTMLButtonElement.onclick (index.html:10:35)
```

As we click on fetch data, the api gives us CORS error.

Example 2 - let us try to setup our own local client and server



```
CORS
  client
    index.html
  server
    > node_modules
    index.js
    package-lock.json
    package.json
```

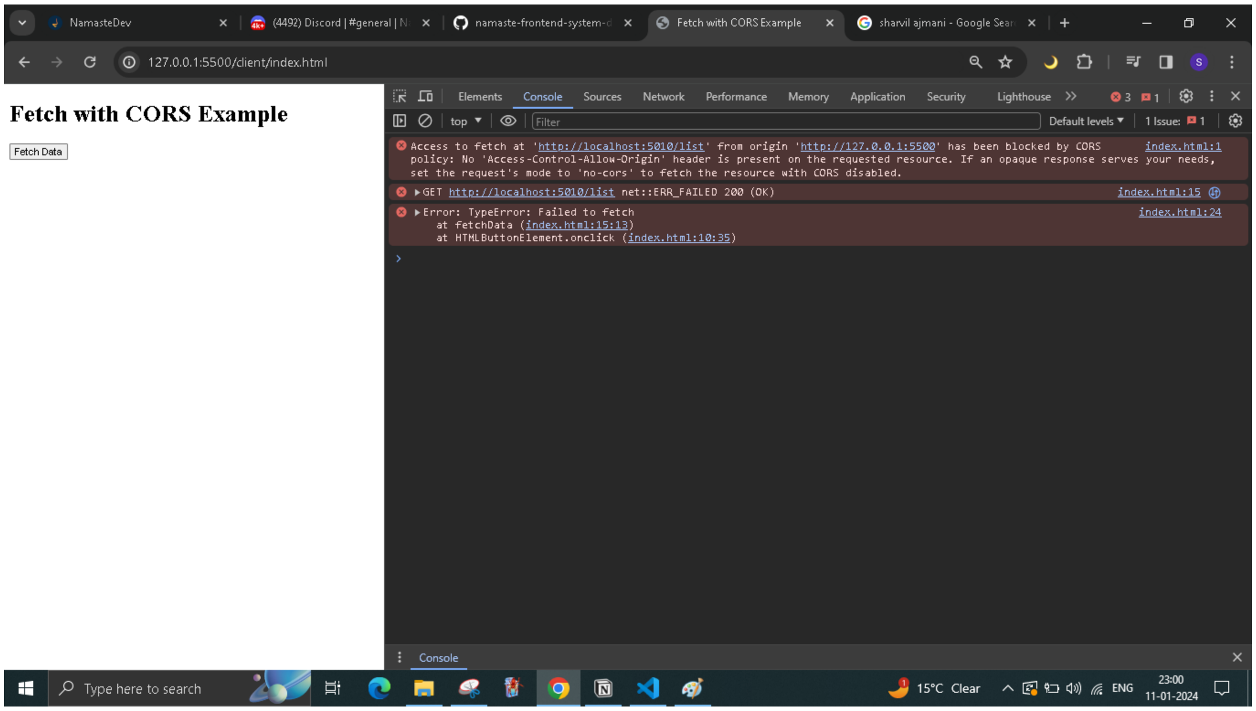
index html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initia
l-scale=1.0">
    <title>Fetch with CORS Example</title>
</head>
<body>
```

```
    <h1>Fetch with CORS Example</h1>
    <button onclick="fetchData()">Fetch Data</button>
    <div id="result"></div>

    <script>
        function fetchData() {
            fetch('http://localhost:5010/list', {
                method: 'GET',
            })
            .then(response => response.json())
            .then(data => {
                // Display the fetched data
                document.getElementById('result').innerText =
JSON.stringify(data, null, 2);
            })
            .catch(error => {
                console.error('Error:', error);
            });
        }
    </script>
</body>
</html>
```

index js

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

app.get('/list', (req, res) => {
  res.send([{
    id: 1,
    title: 'Namaste Frontend System Design'
  }])
})

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



Fetch with CORS Example

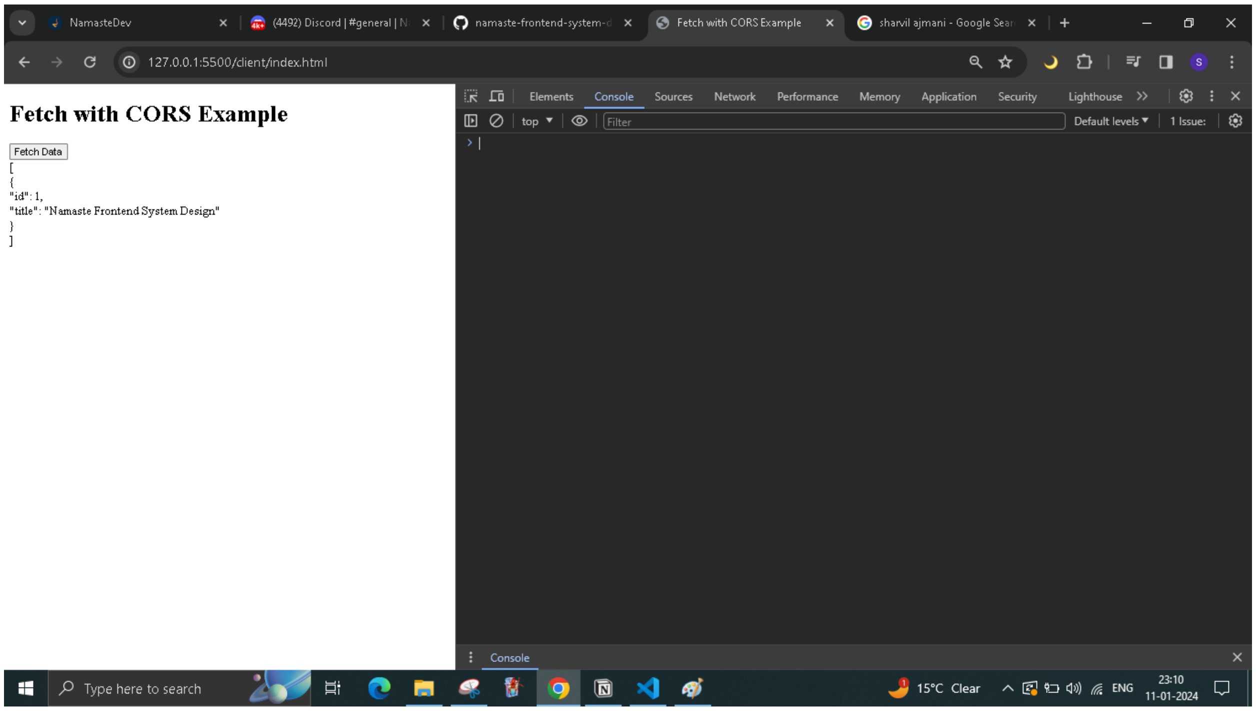
```
Access to fetch at 'http://localhost:5010/list' from origin 'http://127.0.0.1:5500' has been blocked by CORS      index.html:1
policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs,
set the request's mode to 'no-cors' to fetch the resource with CORS disabled.
▶GET http://localhost:5010/list net::ERR_FAILED 200 (OK)                                                       index.html:15
▶Error: TypeError: Failed to fetch                                                                              index.html:24
    at fetchData (index.html:15:13)
    at HTMLButtonElement.onclick (index.html:10:35)
```

We click on fetch data, again we get CORS error as this is default behavior of browser(same origin policy). But some people install CORS extension in browser and are able to access api from cross domain. In such cases we need to set CORS headers in our server to allow/not allow cross domains to access our resources.

We set it using CORS package. - https://www.npmjs.com/package/cors

index.js - whitelisting our cross domain(allowing) - our index html is running on 5500 port which we have allowed

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

var allowedOrigin = ['http://127.0.0.1:5500'];
const corsOptions = {
  origin: function(origin, callback) {
    if (allowedOrigin.indexOf(origin) !== -1 || !origin) {
      callback(null, true);
    } else {
      callback(new Error('CORS error'));
    }
  }
}

app.use(cors(corsOptions));

app.get('/list', (req, res) => {
  res.send([{
    id: 1,
    title: 'Namaste Frontend System Design'
  }])
})

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



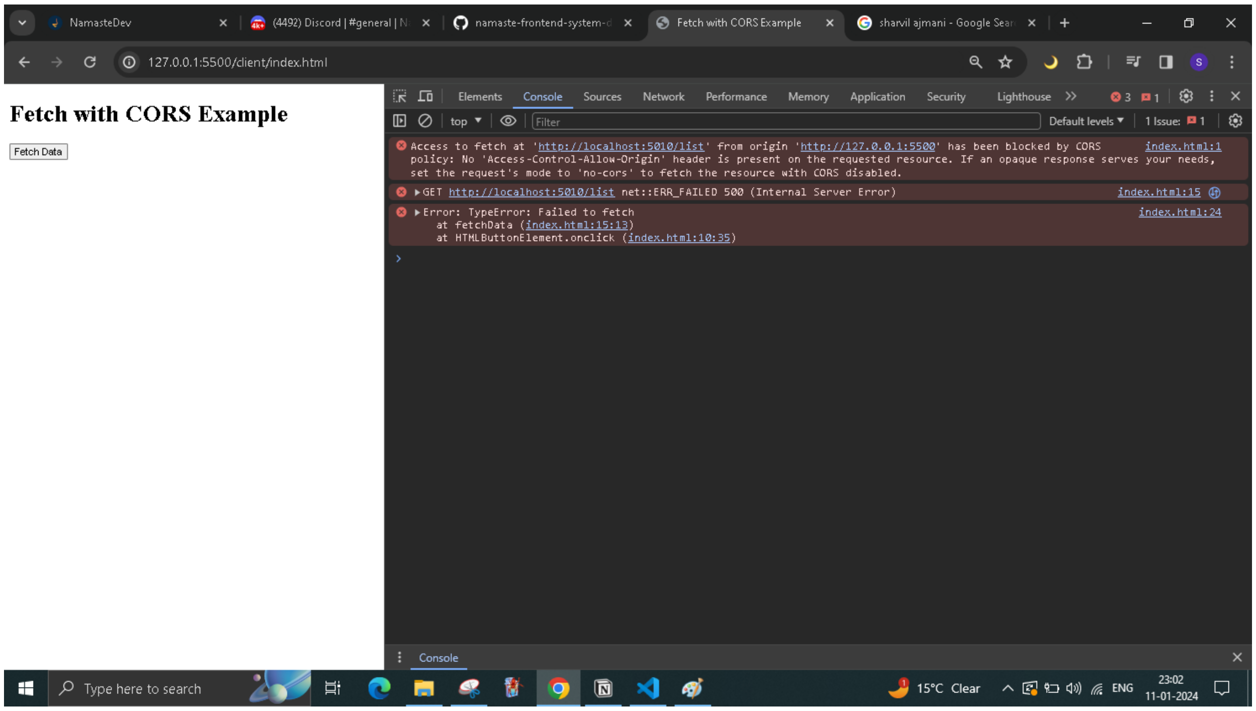
```
[
{
"id": 1,
"title": "Namaste Frontend System Design"
}
]
```

We click on fetch data and we get the data

index js - now we have allowed 5501 which is not our domain

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

var allowedOrigin = ['http://127.0.0.1:5501'];
const corsOptions = {
  origin: function(origin, callback) {
    if (allowedOrigin.indexOf(origin) !== -1 || !origin) {
      callback(null, true);
    } else {
      callback(new Error('CORS error'));
    }
  }
}
```

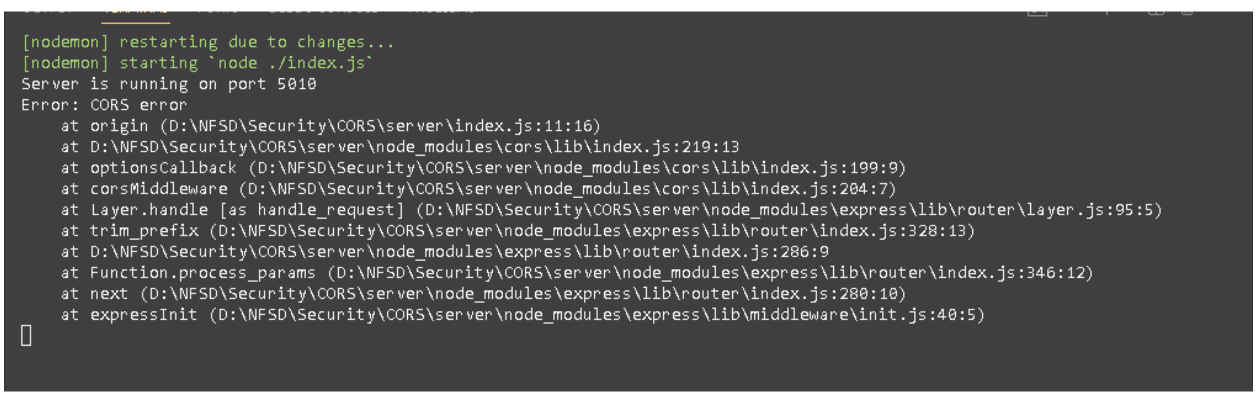
```
app.use(cors(corsOptions));

app.get('/list', (req, res) => {
  res.send([{
    id: 1,
    title: 'Namaste Frontend System Design'
  }])
})

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



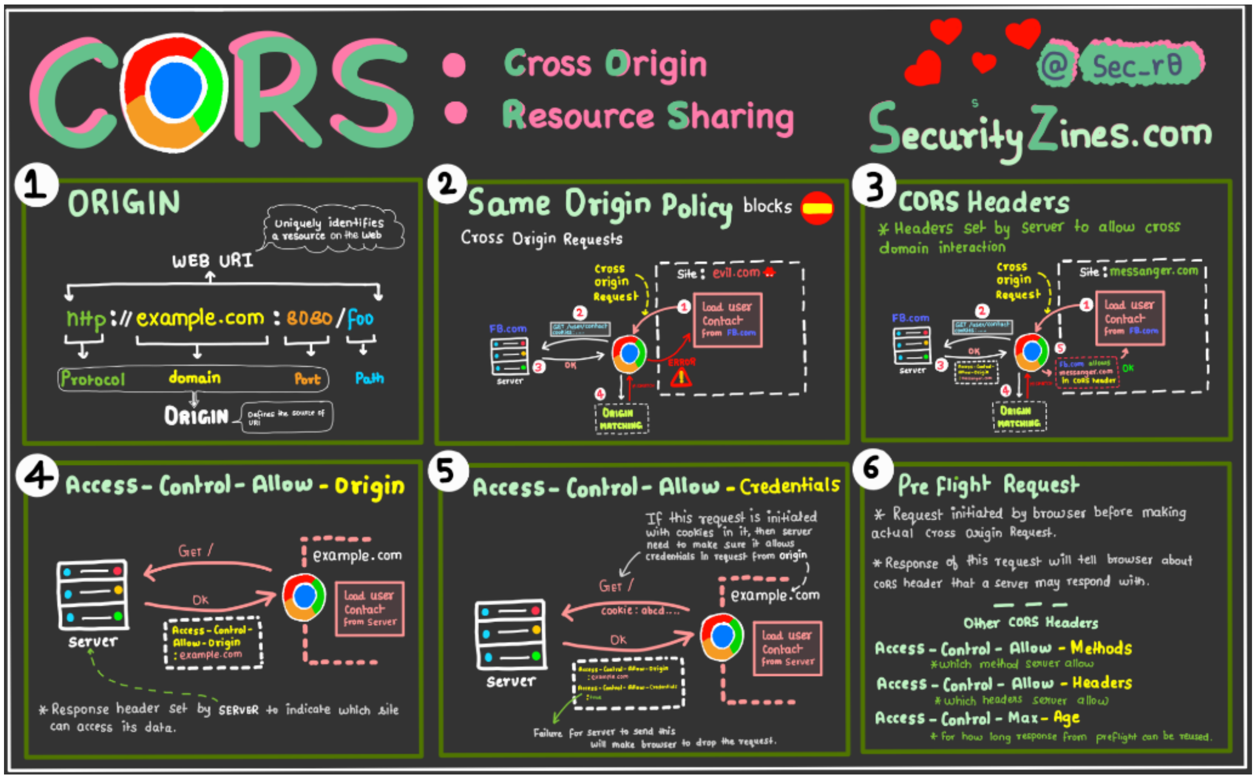
```
Access to fetch at 'http://localhost:5010/list' from origin 'http://127.0.0.1:5500' has been blocked by CORS    index.html:1
policy: No 'Access-Control-Allow-Origin' header is present on the requested resource. If an opaque response serves your needs,
set the request's mode to 'no-cors' to fetch the resource with CORS disabled.
▶GET http://localhost:5010/list net::ERR_FAILED 500 (Internal Server Error)    index.html:15
▶Error: TypeError: Failed to fetch    index.html:24
    at fetchData (index.html:15:13)
    at HTMLButtonElement.onclick (index.html:10:35)
```

We click on fetch data and we get CORS error again.

Our server throws error as we have mentioned in code



```
[nodemon] restarting due to changes...
[nodemon] starting `node ./index.js`
Server is running on port 5010
Error: CORS error
    at origin (D:\NFSD\Security\CORS\server\index.js:11:16)
    at D:\NFSD\Security\CORS\server\node_modules\cors\lib\index.js:219:13
    at optionsCallback (D:\NFSD\Security\CORS\server\node_modules\cors\lib\index.js:199:9)
    at corsMiddleware (D:\NFSD\Security\CORS\server\node_modules\cors\lib\index.js:204:7)
    at Layer.handle [as handle_request] (D:\NFSD\Security\CORS\server\node_modules\express\lib\router\layer.js:95:5)
    at trim_prefix (D:\NFSD\Security\CORS\server\node_modules\express\lib\router\index.js:328:13)
    at D:\NFSD\Security\CORS\server\node_modules\express\lib\router\index.js:286:9
    at Function.process_params (D:\NFSD\Security\CORS\server\node_modules\express\lib\router\index.js:346:12)
    at next (D:\NFSD\Security\CORS\server\node_modules\express\lib\router\index.js:280:10)
    at expressInit (D:\NFSD\Security\CORS\server\node_modules\express\lib\middleware\init.js:40:5)
▯
```


CORS : Cross Origin Resource Sharing
@Sec_rθ
SecurityZines.com
1 ORIGIN
Uniquely identifies a resource on the web
WEB URI
http://example.com:8080/foo
Protocol
domain
Port
Path
ORIGIN
Defines the source of URI
2 Same Origin Policy blocks Cross Origin Requests
Cross origin Request
Site: evil.com
Load user Contact from FB.com
FB.com
Server
OK
ERROR
Origin Matching
3 CORS Headers
* Headers set by server to allow cross domain interaction
Cross origin Request
Site: messanger.com
Load user Contact from FB.com
FB.com
Server
OK
Fb.com allows messanger.com in CORS header
OK
Origin Matching
4 Access-Control-Allow-Origin
Get /
OK
Server
example.com
Load user Contact from Server
Access-Control-Allow-Origin: example.com
* Response header set by SERVER to indicate which site can access its data.
5 Access-Control-Allow-Credentials
If this request is initiated with cookies in it, then server need to make sure it allows credentials in request from origin
Get /
cookie: abcd....
OK
Server
example.com
Load user Contact from Server
Access-Control-Allow-Origin: example.com
Access-Control-Allow-Credentials: true
Failure for server to send this will make browser to drop the request.
6 Pre Flight Request
* Request initiated by browser before making actual cross origin Request.
* Response of this request will tell browser about CORS header that a server may respond with.
Other CORS Headers
Access-Control-Allow-Methods
*which method server allow
Access-Control-Allow-Headers
*which headers server allow
Access-Control-Max-Age
* for how long response from preflight can be reused.