Cors() Policy Options

You Must Learn!

Hasibul Islam
@devhasibulsilam

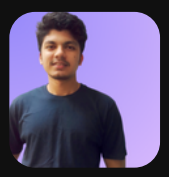
What is Cors()?

CORS (Cross-Origin Resource Sharing) is a security feature implemented by web browsers to control which web pages can make requests to a given domain. It is a crucial aspect of web security that helps prevent malicious websites from making requests to your server on behalf of a user. In Express.js, CORS is typically implemented using middleware.

Install `cors` dependency

CORS is a node.js package for providing a Connect/Express middleware that can be used to enable CORS with various options.

```
npm install cors || yarn add cors
```



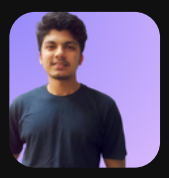
Hasibul Islam
@devhasibulsilam

Simple Usage (Enable All CORS Requests)

```
var express = require('express')
var cors = require('cors')
var app = express()

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Enable CORS for a Single Route

```
var express = require('express')
var cors = require('cors')
var app = express()

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

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```



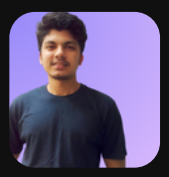
Hasibul Islam
@devhasibulsilam

Configuring CORS

```
var express = require('express')
var cors = require('cors')
var app = express()

var corsOptions = {
  origin: 'http://example.com',
  optionsSuccessStatus: 200 // some legacy browsers (IE11, various SmartTVs) choke on 204
}

app.get('/products/:id', cors(corsOptions), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for only example.com.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Configuring CORS w/ Dynamic Origin

This module supports validating the origin dynamically using a function provided to the origin option. This function will be passed a string that is the origin (or undefined if the request has no origin), and a callback with the signature callback(error, origin). The origin argument to the callback can be any value allowed for the origin option of the middleware, except a function. See the configuration options section for more information on all the possible value types.



Hasibul Islam
@devhasibulsilam

This function is designed to allow the dynamic loading of allowed origin(s) from a backing datasource, like a database.

```
var express = require('express')
var cors = require('cors')
var app = express()

var corsOptions = {
  origin: function (origin, callback) {
    // db.loadOrigins is an example call to load
    // a list of origins from a backing database
    db.loadOrigins(function (error, origins) {
      callback(error, origins)
    })
  }
}
```

```
app.get('/products/:id', cors(corsOptions), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for an allowed domain.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Enabling CORS Pre-Flight

Certain CORS requests are considered ‘complex’ and require an initial OPTIONS request (called the “pre-flight request”). An example of a ‘complex’ CORS request is one that uses an HTTP verb other than GET/HEAD/POST (such as DELETE) or that uses custom headers.

Hasibul Islam
@devhasibulsilam

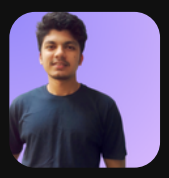
To enable pre-flighting, you must add a new OPTIONS handler for the route you want to support:

```
var express = require('express')
var cors = require('cors')
var app = express()

app.options('/products/:id', cors()) // enable pre-flight request for DELETE request
app.del('/products/:id', cors(), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

You can also enable pre-flight across-the-board like so:

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

NOTE: *When using this middleware as an application level middleware (for example, app.use(cors())), pre-flight requests are already handled for all routes.*



Hasibul Islam
@devhasibulsilam

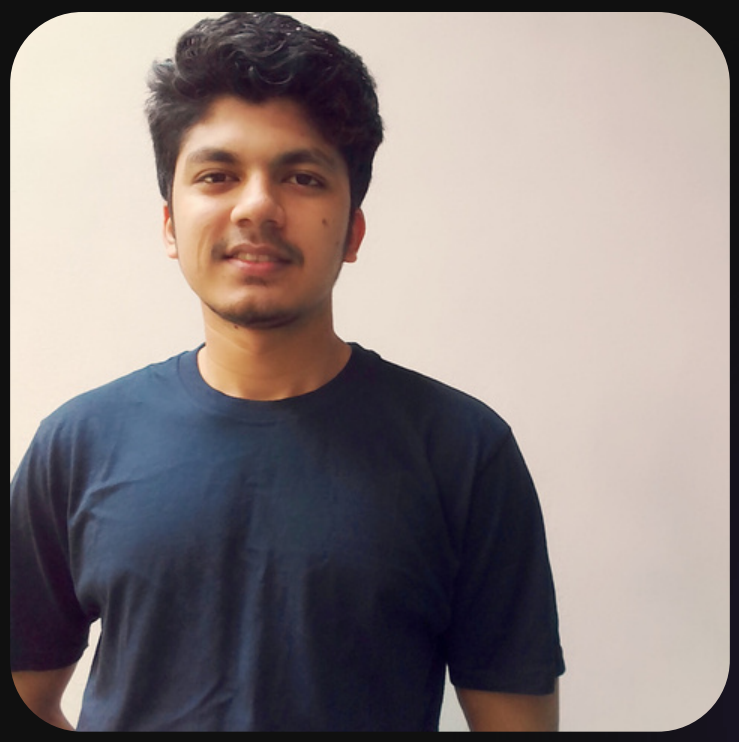
Configuring CORS w/ Dynamic Origin

```js
var express = require('express')
var cors = require('cors')
var app = express()

var allowlist = ['http://example1.com', 'http://example2.com']
var corsOptionsDelegate = function (req, callback) {
  var corsOptions;
  if (allowlist.indexOf(req.header('Origin')) !== -1) {
    corsOptions = { origin: true } // reflect (enable) the requested origin in the CORS response
  } else {
    corsOptions = { origin: false } // disable CORS for this request
  }
  callback(null, corsOptions) // callback expects two parameters: error and options
}

app.get('/products/:id', cors(corsOptionsDelegate), function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for an allowed domain.'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

Default Configuration

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

Remember *that CORS is primarily a browser security feature, so these restrictions are enforced by the browser. Always ensure your server also has appropriate security measures in place.*



Hasibul Islam

@devhasibulislam

Did you find it useful?

Share your thoughts.


Follow me on social
DM for Queries!
@devhasibulislam