**axios**

Promise based HTTP client for the browser and node.js

For reference - <https://github.com/axios/axios>

## Features

* Make [XMLHttpRequests](https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest) from the browser
* Make [http](http://nodejs.org/api/http.html) requests from node.js
* Supports the [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) API
* Intercept request and response
* Transform request and response data
* Cancel requests
* Automatic transforms for JSON data
* Client side support for protecting against [XSRF](http://en.wikipedia.org/wiki/Cross-site_request_forgery)

## Browser Support

| **[Chrome](https://camo.githubusercontent.com/26846e979600799e9f4273d38bd9e5cb7bb8d6d0/68747470733a2f2f7261772e6769746875622e636f6d2f616c7272612f62726f777365722d6c6f676f732f6d61737465722f7372632f6368726f6d652f6368726f6d655f34387834382e706e67)** | **[Firefox](https://camo.githubusercontent.com/6087557f69ec6585eb7f8d7bd7d9ecb6b7f51ba1/68747470733a2f2f7261772e6769746875622e636f6d2f616c7272612f62726f777365722d6c6f676f732f6d61737465722f7372632f66697265666f782f66697265666f785f34387834382e706e67)** | **[Safari](https://camo.githubusercontent.com/6fbaeb334b99e74ddd89190a42766ea3b4600d2c/68747470733a2f2f7261772e6769746875622e636f6d2f616c7272612f62726f777365722d6c6f676f732f6d61737465722f7372632f7361666172692f7361666172695f34387834382e706e67)** | **[Opera](https://camo.githubusercontent.com/96d2405a936da1fb8988db0c1d304d3db04b8a52/68747470733a2f2f7261772e6769746875622e636f6d2f616c7272612f62726f777365722d6c6f676f732f6d61737465722f7372632f6f706572612f6f706572615f34387834382e706e67)** | **[Edge](https://camo.githubusercontent.com/826b3030243b09465bf14cf420704344f5eee991/68747470733a2f2f7261772e6769746875622e636f6d2f616c7272612f62726f777365722d6c6f676f732f6d61737465722f7372632f656467652f656467655f34387834382e706e67)** |  |
| --- | --- | --- | --- | --- | --- |
| Latest ✔ | Latest ✔ | Latest ✔ | Latest ✔ | Latest ✔ | 11 ✔ |

## Installing

Using npm:

$ npm install axios

Using bower:

$ bower install axios

Using cdn:

<script src="https://unpkg.com/axios/dist/axios.min.js"></script>

## Example

Performing a GET request

const axios = require('axios');

// Make a request for a user with a given ID

axios.get('/user?ID=12345')

.then(function (response) {

// handle success

console.log(response);

})

.catch(function (error) {

// handle error

console.log(error);

})

.then(function () {

// always executed

});

// Optionally the request above could also be done as

axios.get('/user', {

params: {

ID: 12345

}

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

})

.then(function () {

// always executed

});

// Want to use async/await? Add the `async` keyword to your outer function/method.

async function getUser() {

try {

const response = await axios.get('/user?ID=12345');

console.log(response);

} catch (error) {

console.error(error);

}

}

**NOTE:** async/await is part of ECMAScript 2017 and is not supported in Internet Explorer and older browsers, so use with caution.

Performing a POST request

axios.post('/user', {

firstName: 'Fred',

lastName: 'Flintstone'

})

.then(function (response) {

console.log(response);

})

.catch(function (error) {

console.log(error);

});

Performing multiple concurrent requests

function getUserAccount() {

return axios.get('/user/12345');

}

function getUserPermissions() {

return axios.get('/user/12345/permissions');

}

axios.all([getUserAccount(), getUserPermissions()])

.then(axios.spread(function (acct, perms) {

// Both requests are now complete

}));

## Interceptors

You can intercept requests or responses before they are handled by then or catch.

// Add a request interceptor

axios.interceptors.request.use(function (config) {

// Do something before request is sent

return config;

}, function (error) {

// Do something with request error

return Promise.reject(error);

});

// Add a response interceptor

axios.interceptors.response.use(function (response) {

// Do something with response data

return response;

}, function (error) {

// Do something with response error

return Promise.reject(error);

});

If you may need to remove an interceptor later you can.

const myInterceptor = axios.interceptors.request.use(function () {/\*...\*/});

axios.interceptors.request.eject(myInterceptor);

You can add interceptors to a custom instance of axios.

const instance = axios.create();

instance.interceptors.request.use(function () {/\*...\*/});

## Handling Errors

axios.get('/user/12345')

.catch(function (error) {

if (error.response) {

// The request was made and the server responded with a status code

// that falls out of the range of 2xx

console.log(error.response.data);

console.log(error.response.status);

console.log(error.response.headers);

} else if (error.request) {

// The request was made but no response was received

// `error.request` is an instance of XMLHttpRequest in the browser and an instance of

// http.ClientRequest in node.js

console.log(error.request);

} else {

// Something happened in setting up the request that triggered an Error

console.log('Error', error.message);

}

console.log(error.config);

});

You can define a custom HTTP status code error range using the validateStatus config option.

axios.get('/user/12345', {

validateStatus: function (status) {

return status < 500; // Reject only if the status code is greater than or equal to 500

}

})