

net

Issue HTTP/HTTPS requests using Chromium's native networking library

Process: [Main](#)

The `net` module is a client-side API for issuing HTTP(S) requests. It is similar to the [HTTP](#) and [HTTPS](#) modules of Node.js but uses Chromium's native networking library instead of the Node.js implementation, offering better support for web proxies. It also supports checking network status.

The following is a non-exhaustive list of why you may consider using the `net` module instead of the native Node.js modules:

- Automatic management of system proxy configuration, support of the wpad protocol and proxy pac configuration files.
- Automatic tunneling of HTTPS requests.
- Support for authenticating proxies using basic, digest, NTLM, Kerberos or negotiate authentication schemes.
- Support for traffic monitoring proxies: Fiddler-like proxies used for access control and monitoring.

The API components (including classes, methods, properties and event names) are similar to those used in Node.js.

Example usage:

```
const { app } = require('electron')
app.whenReady().then(() => {
  const { net } = require('electron')
  const request = net.request('https://github.com')
  request.on('response', (response) => {
    console.log(`STATUS: ${response.statusCode}`)
    console.log(`HEADERS: ${JSON.stringify(response.headers)}`)
    response.on('data', (chunk) => {
      console.log(`BODY: ${chunk}`)
    })
    response.on('end', () => {
      console.log('No more data in response.')
    })
  })
  request.end()
})
```

The `net` API can be used only after the application emits the `ready` event. Trying to use the module before the `ready` event will throw an error.

Methods

The `net` module has the following methods:

`net.request(options)`

- `options` (`ClientRequestConstructorOptions` | `string`) - The `ClientRequest` constructor options.

Returns [ClientRequest](#)

Creates a `ClientRequest` instance using the provided `options` which are directly forwarded to the `ClientRequest` constructor. The `net.request` method would be used to issue both secure and insecure HTTP requests according to the specified protocol scheme in the `options` object.

`net.isOnline()`

Returns `boolean` - Whether there is currently internet connection.

A return value of `false` is a pretty strong indicator that the user won't be able to connect to remote sites. However, a return value of `true` is inconclusive; even if some link is up, it is uncertain whether a particular connection attempt to a particular remote site will be successful.

Properties

`net.online` *Readonly*

A `boolean` property. Whether there is currently internet connection.

A return value of `false` is a pretty strong indicator that the user won't be able to connect to remote sites. However, a return value of `true` is inconclusive; even if some link is up, it is uncertain whether a particular connection attempt to a particular remote site will be successful.