# type-is

[NPM Version](https://npmjs.org/package/type-is) [NPM Downloads](https://npmjs.org/package/type-is) [Node.js Version](https://nodejs.org/en/download) [Build Status](https://travis-ci.org/jshttp/type-is) [Test Coverage](https://coveralls.io/r/jshttp/type-is?branch=master)

Infer the content-type of a request.

### Install

This is a [Node.js](https://nodejs.org/en/) module available through the [npm registry](https://www.npmjs.com/). Installation is done using the [npm install command](https://docs.npmjs.com/getting-started/installing-npm-packages-locally):

$ npm install type-is

## API

var http = require('http')

var typeis = require('type-is')

http.createServer(function (req, res) {

var istext = typeis(req, ['text/\*'])

res.end('you ' + (istext ? 'sent' : 'did not send') + ' me text')

})

### typeis(request, types)

Checks if the request is one of the types. If the request has no body, even if there is a Content-Type header, then null is returned. If the Content-Type header is invalid or does not matches any of the types, then false is returned. Otherwise, a string of the type that matched is returned.

The request argument is expected to be a Node.js HTTP request. The types argument is an array of type strings.

Each type in the types array can be one of the following:

* A file extension name such as json. This name will be returned if matched.
* A mime type such as application/json.
* A mime type with a wildcard such as \*/\* or \*/json or application/\*. The full mime type will be returned if matched.
* A suffix such as +json. This can be combined with a wildcard such as \*/vnd+json or application/\*+json. The full mime type will be returned if matched.

Some examples to illustrate the inputs and returned value:

// req.headers.content-type = 'application/json'

typeis(req, ['json']) // => 'json'

typeis(req, ['html', 'json']) // => 'json'

typeis(req, ['application/\*']) // => 'application/json'

typeis(req, ['application/json']) // => 'application/json'

typeis(req, ['html']) // => false

### typeis.hasBody(request)

Returns a Boolean if the given request has a body, regardless of the Content-Type header.

Having a body has no relation to how large the body is (it may be 0 bytes). This is similar to how file existence works. If a body does exist, then this indicates that there is data to read from the Node.js request stream.

if (typeis.hasBody(req)) {

// read the body, since there is one

req.on('data', function (chunk) {

// ...

})

}

### typeis.is(mediaType, types)

Checks if the mediaType is one of the types. If the mediaType is invalid or does not matches any of the types, then false is returned. Otherwise, a string of the type that matched is returned.

The mediaType argument is expected to be a [media type](https://tools.ietf.org/html/rfc6838) string. The types argument is an array of type strings.

Each type in the types array can be one of the following:

* A file extension name such as json. This name will be returned if matched.
* A mime type such as application/json.
* A mime type with a wildcard such as \*/\* or \*/json or application/\*. The full mime type will be returned if matched.
* A suffix such as +json. This can be combined with a wildcard such as \*/vnd+json or application/\*+json. The full mime type will be returned if matched.

Some examples to illustrate the inputs and returned value:

var mediaType = 'application/json'

typeis.is(mediaType, ['json']) // => 'json'

typeis.is(mediaType, ['html', 'json']) // => 'json'

typeis.is(mediaType, ['application/\*']) // => 'application/json'

typeis.is(mediaType, ['application/json']) // => 'application/json'

typeis.is(mediaType, ['html']) // => false

## Examples

### Example body parser

var express = require('express')

var typeis = require('type-is')

var app = express()

app.use(function bodyParser (req, res, next) {

if (!typeis.hasBody(req)) {

return next()

}

switch (typeis(req, ['urlencoded', 'json', 'multipart'])) {

case 'urlencoded':

// parse urlencoded body

throw new Error('implement urlencoded body parsing')

case 'json':

// parse json body

throw new Error('implement json body parsing')

case 'multipart':

// parse multipart body

throw new Error('implement multipart body parsing')

default:

// 415 error code

res.statusCode = 415

res.end()

break

}

})

## License

[MIT](http://license)