mime-db

npm v1.54.0 downloads 826.7M/month node >= 0.6
ci success coverage 100%

This is a large database of mime types and information about them. It consists of a single, public JSON file and does not include any logic, allowing it to remain as un-opinionated as possible with an API. It aggregates data from the following sources:

- https://www.iana.org/assignments/media-types/media-types.xhtml
- https://svn.apache.org/repos/asf/httpd/httpd/trunk/docs/conf/mime.types
- https://hg.nginx.org/nginx/raw-file/default/conf/mime.types

Installation

```
npm install mime-db
```

Database Download

If you intend to use this in a web browser, you can conveniently access the JSON file via jsDelivr, a popular CDN (Content Delivery Network). To ensure stability and compatibility, it is advisable to specify a release tag instead of using the ‘master’ branch. This is because the JSON file’s format might change in future updates, and relying on a specific release tag will prevent potential issues arising from these changes.

```
https://cdn.jsdelivr.net/gh/jshttp/mime-
db@master/db.json
```

Usage

```
var db = require('mime-db')

// grab data on .js files
var data = db['application/javascript']
```

Data Structure

The JSON file is a map lookup for lowercased mime types. Each mime type has the following properties:

- `.source` - where the mime type is defined. If not set, it’s probably a custom media type.
  - `apache` - Apache common media types
  - `iana` - IANA-defined media types
  - `nginx` - nginx media types
- `.extensions[]` - known extensions associated with this mime type.

- `.compressible` - whether a file of this type can be gzipped.
- `.charset` - the default charset associated with this type, if any.

If unknown, every property could be `undefined`.

Note on MIME Type Data and Semver

This package considers the programmatic api as the semver compatibility. This means the MIME type resolution is *not* considered in the semver bumps. This means that if you want to pin your `mime-db` data you will need to do it in your application. While this expectation was not set in docs until now, it is how the pacakge operated, so we do not feel this is a breaking change.

Contributing

The primary way to contribute to this database is by updating the data in one of the upstream sources. The database is updated from the upstreams periodically and will pull in any changes.

Registering Media Types

The best way to get new media types included in this library is to register them with the IANA. The community registration procedure is outlined in RFC 6838 section 5. Types registered with the IANA are automatically pulled into this library.

Direct Inclusion

If that is not possible / feasible, they can be added directly here as a “custom” type. To do this, it is required to have a primary source that definitively lists the media type. If an extension is going to be listed as associated with this media type, the source must definitively link the media type and extension as well.

To edit the database, only make PRs against `src/custom-types.json` or `src/custom-suffix.json`.

The `src/custom-types.json` file is a JSON object with the MIME type as the keys and the values being an object with the following keys:

- `compressible` - leave out if you don’t know, otherwise `true`/`false` to indicate whether the data represented by the type is typically compressible.
- `extensions` - include an array of file extensions that are associated with the type.
- `notes` - human-readable notes about the type, typically what the type is.

- `sources` - include an array of URLs of where the MIME type and the associated extensions are sourced from. This needs to be a primary source; links to type aggregating sites and Wikipedia are *not acceptable*.

To update the build, run `npm run build`.