`FeedHandler`). The old names are still available when requiring `htmlparser2`, your code should work as expected.

Security contact information

To report a security vulnerability, please use the Tidelift security contact. Tidelift will coordinate the fix and disclosure.

`htmlparser2` for enterprise

Available as part of the Tidelift Subscription

The maintainers of `htmlparser2` and thousands of other packages are working with Tidelift to deliver commercial support and maintenance for the open source dependencies you use to build your applications. Save time, reduce risk, and improve code health, while paying the maintainers of the exact dependencies you use. Learn more.

htmlparser2

The fast & forgiving HTML/XML parser.

Installation

```
npm install htmlparser2
```

A live demo of `htmlparser2` is available here.

Ecosystem

Name	Description
htmlparser2	Fast & forgiving HTML/XML parser
domhandler	Handler for htmlparser2 that turns documents into a DOM
domutils	Utilities for working with domhandler's DOM
css-select	

Name	Description
	CSS selector engine, compatible with domhandler's DOM
cheerio	The jQuery API for domhandler's DOM
dom-serializer	Serializer for domhandler's DOM

Usage

`htmlparser2` itself provides a callback interface that allows consumption of documents with minimal allocations. For a more ergonomic experience, read Getting a DOM below.

```
const htmlparser2 =
        require("htmlparser2");
const parser = new htmlparser2.Parser({
    onopentag(name, attributes) {
        /*
         * This fires when a new tag is
        opened.
         *
         * If you don't need an
        aggregated `attributes` object,
         * have a look at the
        `onopentagname` and
        `onattribute` events.
         */
```



```
html-dom-parser     : 3.56804 ms/file ±
5.15621
libxmljs            : 4.07490 ms/file ±
2.99869
htmljs-parser       : 6.15812 ms/file ±
7.52497
parse5              : 9.70406 ms/file ±
6.74872
htmlparser          : 15.0596 ms/file ±
89.0826
html-parser         : 28.6282 ms/file ±
22.6652
saxes               : 45.7921 ms/file ±
128.691
html5               : 120.844 ms/file ±
153.944
```

How does this module differ from node-htmlparser?

In 2011, this module started as a fork of the `htmlparser` module. `htmlparser2` was rewritten multiple times and, while it maintains an API that's mostly compatible with `htmlparser` in most cases, the projects don't share any code anymore.

The parser now provides a callback interface inspired by sax.js (originally targeted at readabilitySAX). As a result, old handlers won't work anymore.

The `DefaultHandler` and the `RssHandler` were renamed to clarify their purpose (to `DomHandler` and

Parsing RSS/RDF/Atom Feeds

```
const feed =
        htmlparser2.parseFeed(content,
        options);
```

Note: While the provided feed handler works for most feeds, you might want to use danmactough/node-feedparser, which is much better tested and actively maintained.

Performance

After having some artificial benchmarks for some time, **@AndreasMadsen** published his `htmlparser-benchmark`, which benchmarks HTML parses based on real-world websites.

At the time of writing, the latest versions of all supported parsers show the following performance characteristics on GitHub Actions (sourced from here):

```
htmlparser2        : 2.17215 ms/file ±
3.81587
node-html-parser   : 2.35983 ms/file ±
1.54487
html5parser        : 2.43468 ms/file ±
2.81501
neutron-html5parser: 2.61356 ms/file ±
1.70324
htmlparser2-dom    : 3.09034 ms/file ±
4.77033
```



```
    if (name === "script" &&
    attributes.type === "text/
    javascript") {
        console.log("JS! Hooray!");
    }
},
ontext(text) {
    /*
     * Fires whenever a section of
    text was processed.
     *
     * Note that this can fire at
    any point within text and you
    might
     * have to stich together
    multiple pieces.
     */
    console.log("-->", text);
},
onclosetag(tagname) {
    /*
     * Fires when a tag is closed.
     *
     * You can rely on this event
    only firing when you have
    received an
     * equivalent opening tag
    before. Closing tags without
    corresponding
     * opening tags will be ignored.
     */
    if (tagname === "script") {
        console.log("That's it?!");
    }
```

```
    },
});
parser.write(
    "Xyz <script type='text/
        javascript'>const foo =
        '<<bar>>';</ script>"
);
parser.end();
```

Output (with multiple text events combined):

```
--> Xyz
JS! Hooray!
--> const foo = '<<bar>>';
That's it?!
```

This example only shows three of the possible events. Read more about the parser, its events and options in the wiki.

Usage with streams

While the `Parser` interface closely resembles Node.js streams, it's not a 100% match. Use the `WritableStream` interface to process a streaming input:

```
const { WritableStream } =
        require("htmlparser2/lib/
        WritableStream");
const parserStream = new
        WritableStream({
    ontext(text) {
        console.log("Streaming:", text);
```



```
    },
});

const htmlStream =
        fs.createReadStream("./my-
        file.html");
htmlStream.pipe(parserStream).on("finish",
        () => console.log("done"));
```

Getting a DOM

The `DomHandler` produces a DOM (document object model) that can be manipulated using the `DomUtils` helper.

```
const htmlparser2 =
        require("htmlparser2");

const dom =
        htmlparser2.parseDocument(htmlString);
```

The `DomHandler`, while still bundled with this module, was moved to its own module. Have a look at that for further information.