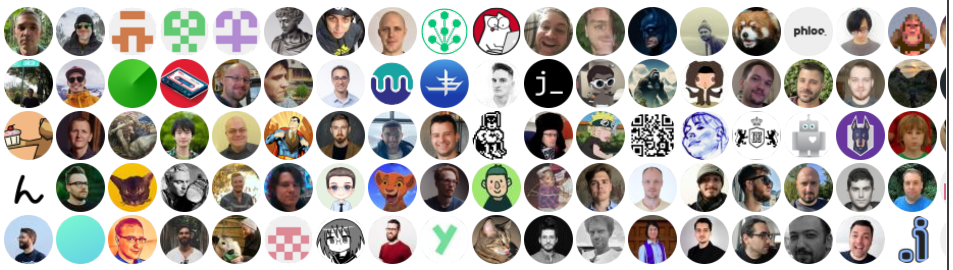
Deps

PostHTML

Maintainers

<img width="150 height="150" src="https://avatars0.githubusercontent.com/u/2789192?s=460&v=4">
Ivan Demidov <img width="150 height="150" src="https://avatars.githubusercontent.com/u/1510217?v=3&s=150">
Ivan Voischev

Contributors



Backers

Thank you to all our backers! 🙏 [Become a backer]

PostHTML is a tool for transforming HTML/XML with JS plugins. PostHTML itself is very small. It includes only a HTML parser, a HTML node tree API and a node tree stringifier.

All HTML transformations are made by plugins. And these plugins are just small plain JS functions, which receive a HTML node tree, transform it, and return a modified tree.

For more detailed information about PostHTML in general take a look at the docs.

Dependencies

Name	Status	Description
posthtml-parser		Parser HTML/XML to PostHTMLTree
posthtml-render		Render PostHTMLTree to HTML/XML

Create to your project

Install

```
    posthtml({
      parser: sugarml(),
      plugins: [include()],
      template: true  // only rollup-
        plugin-posthtml-template
    })
  ]
};
```

Parser

```
import pug from 'posthtml-pug'

posthtml().process(html, { parser:
        pug(options) }).then((result)
        => result.html)
```

Name	Status	Description
posthtml-pug		Pug Parser
sugarml		SugarML Parser

Plugins

In case you want to develop your own plugin, we recommend using posthtml-plugin-starter to get started.

posthtml-plugins

awesome-posthtml

```
  ]
}

export default config
```

Rollup

```
$ npm i rollup-plugin-posthtml -D
# or
$ npm i rollup-plugin-posthtml-template
        -D

import { join } from 'path';

import posthtml from 'rollup-plugin-
        posthtml-template';
// or
// import posthtml from 'rollup-plugin-
        posthtml';

import sugarml from 'posthtml-
        sugarml';  // npm i posthtml-
        sugarml -D
import include from 'posthtml-
        include';  // npm i posthtml-
        include -D

export default {
  entry: join(__dirname, 'main.js'),
  dest: join(__dirname, 'bundle.js'),
  format: 'iife',
  plugins: [
```


Usage

API

Sync

```
import posthtml from 'posthtml'

const html = `
  <component>
    <title>Super Title</title>
    <text>Awesome Text</text>
  </component>
`

const result = posthtml()
  .use(require('posthtml-custom-
        elements')())
  .process(html, { sync: true })
  .html

console.log(result)
```

```
  Super Title
  Awesome Text
```

:warning: Async Plugins can't be used in sync mode and will throw an Error. It's recommended to use PostHTML asynchronously whenever possible.

Async

```
import posthtml from 'posthtml'

const html = `
  <html>
    <body>
      <p class="wow">OMG</p>
    </body>
  </html>
`

posthtml(
  [
    require('posthtml-to-svg-tags')(),
    require('posthtml-extend-attrs')({
      attrsTree: {
        '.wow' : {
          id: 'wow_id',
          fill: '#4A83B4',
          'fill-rule': 'evenodd',
          'font-family': 'Verdana'
        }
      }
    })
  ])
  .process(html/*, options */)
  .then((result) =>
        console.log(result.html))

<svg xmlns="http://www.w3.org/2000/svg">
  <text
    class="wow"
    id="wow_id"
```



```
import { LoaderOptionsPlugin } from
        'webpack'

const config = {
  module: {
    rules: [
      {
        test: /\.html$/,
        use: [
          {
            loader: 'html-loader',
            options: { minimize: true }
          },
          {
            loader: 'posthtml-loader'
          }
        ]
      }
    ]
  },
  plugins: [
    new LoaderOptionsPlugin({
      options: {
        posthtml(ctx) {
          return {
            parser: require('posthtml-
        pug'),
            plugins: [
              require('posthtml-bem')()
            ]
          }
        }
      }
    })
```

Webpack

```
npm i -D html-loader posthtml-loader
```

v1.x

webpack.config.js

```
const config = {
  module: {
    loaders: [
      {
        test: /\.html$/,
        loader: 'html!posthtml'
      }
    ]
  },
  posthtml: (ctx) => ({
    parser: require('posthtml-pug'),
    plugins: [
      require('posthtml-bem')()
    ]
  })
}

export default config
```

v2.x

webpack.config.js


```
      fill="#4A83B4"
      fill-rule="evenodd" font-
          family="Verdana">
        OMG
    </text>
</svg>
```

Directives

```
import posthtml from 'posthtml'

const php = `
  <component>
    <title><?php echo $title; ?></title>
    <text><?php echo $article; ?></text>
  </component>
`

const result = posthtml()
  .use(require('posthtml-custom-
        elements')())
  .process(html, {
    directives: [
      { name: '?php', start: '<', end:
        '>' }
    ]
  })
  .html

console.log(result)


  <?php echo $title; ?>
  <?php echo $article; ?>
```

CLI

```
npm i posthtml-cli
```

```
"scripts": {
  "posthtml": "posthtml -o output.html
        -i input.html -c config.json"
}
```

```
npm run posthtml
```

Gulp

```
npm i -D gulp-posthtml
```

```
import tap from 'gulp-tap'
import posthtml from 'gulp-posthtml'
import { task, src, dest } from 'gulp'

task('html', () => {
  let path

  const plugins = [ require('posthtml-
        include')({ root: `${path}` }) ]
  const options = {}

  src('src/**/*.html')
    .pipe(tap((file) => path =
        file.path))
    .pipe(posthtml(plugins, options))
    .pipe(dest('build/'))
})
```

Check project-stub for an example with Gulp

Grunt

```
npm i -D grunt-posthtml
```

```
posthtml: {
  options: {
    use: [
      require('posthtml-doctype')({
        doctype: 'HTML 5' }),
      require('posthtml-include')({
        root: './', encoding: 'utf-8' })
    ]
  },
  build: {
    files: [
      {
        dot: true,
        cwd: 'html/',
        src: ['*.html'],
        dest: 'tmp/',
        expand: true,
      }
    ]
  }
}
```