webpack-stream

5.1.1 • Public • Published a month ago

Readme

9 Dependencies

274 Dependents

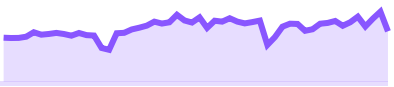
16 Versions

install

```
> npm i webpack-stream
```

weekly downloads

59,426



version	license
5.1.1	MIT
open issues	**pull requests**
43	5
homepage	**repository**
github.com	github
last publish	
a month ago	

collaborators

Test with RunKit | Report a vulnerability

webpack-stream build passing

Run webpack as a stream to conveniently integrate with gulp.

```
npm install webpack-stream
9 dependencies     version 5.1.1
205 dependents     updated a month ago
```

Installation

If you have `npm` run the following command in the console `npm install --save-dev webpack-stream`

Usage

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
gulp.task('default', function() {
  return gulp.src('src/entry.js')
    .pipe(webpack())
    .pipe(gulp.dest('dist/'));
});
```

The above will compile `src/entry.js` into assets with webpack into `dist/` with the output filename of `[hash].js` (webpack generated hash of the build).

You can pass webpack options in with the first argument, including `watch` which will greatly decrease compilation times:

```
return gulp.src('src/entry.js')
  .pipe(webpack({
    watch: true,
    module: {
      rules: [
        { test: /\.css$/, loader: 'style!css' },
      ],
    },
  }))
  .pipe(gulp.dest('dist/'));
```

Or just pass in your `webpack.config.js` :

```
return gulp.src('src/entry.js')
  .pipe(webpack( require('./webpack.config.js') ))
  .pipe(gulp.dest('dist/'));
```

If you would like to use a different version of webpack than the one this plugin uses, pass in an optional 2nd argument:

```
const gulp = require('gulp');
const webpack = require('webpack');
const gulpWebpack = require('webpack-stream');
gulp.task('default', function() {
  return gulp.src('src/entry.js')
    .pipe(gulpWebpack({}, webpack))
    .pipe(gulp.dest('dist/'));
});
```

Pass in 3rd argument if you want to access the stats outputted from webpack when the compilation is done:

```
const gulp = require('gulp');
```

```
const webpack = require('webpack-stream');
gulp.task('default', function() {
  return gulp.src('src/entry.js')
    .pipe(webpack({
      /* config */
    }, null, function(err, stats) {
      /* Use stats to do more things if needed */
    }))
    .pipe(gulp.dest('dist/'));
});
```

Usage with gulp watch

To use gulp `watch`, it's required that you explicitly pass webpack in the 2nd argument for a cached `compiler` instance to be used on subsequent runs.

Please note that gulp `watch` and webpack `watch` are mutually exclusive.

```
var gulp = require('gulp');
var compiler = require('webpack');
var webpack = require('webpack-stream');

gulp.task('build', function() {
  return gulp.src('src/entry.js')
    .pipe(webpack({
      /* config */
    }, compiler, function(err, stats) {
      /* Use stats to do more things if needed */
    }))
    .pipe(gulp.dest('dist/'));
});

gulp.task('default', ['build'], function() {
  gulp.watch(['src/**/*.js'], ['build']);
});
```

Multiple Entry Points

A common request is how to handle multiple entry points. You can continue to pass in an `entry` option in your typical webpack config like so:

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
gulp.task('default', function() {
  return gulp.src('src/entry.js')
    .pipe(webpack({
      entry: {
        app: 'src/app.js',
        test: 'test/test.js',
      },
      output: {
        filename: '[name].js',
      },
    }))
    .pipe(gulp.dest('dist/'));
});
```

Or pipe files through a stream that names the chunks. A convenient library for this is vinyl-named:

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
const named = require('vinyl-named');
gulp.task('default', function() {
  return gulp.src(['src/app.js', 'test/test.js'])
    .pipe(named())
    .pipe(webpack())
    .pipe(gulp.dest('dist/'));
```

```
});
```

The above `named()` stream will add a `.named` property to the vinyl files passing through. The `webpack()` stream will read those as entry points and even group entry points with common names together.

Source Maps

Source maps are built into webpack, specify a devtool:

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
const named = require('vinyl-named');
gulp.task('default', function() {
  return gulp.src(['src/app.js', 'test/test.js'])
    .pipe(named())
    .pipe(webpack({
      devtool: 'source-map'
    }))
    .pipe(gulp.dest('dist/'));
});
```

Now the appropriate `.map` files will be emitted. Or set to `inline-source-map` to inline the source maps into the files.

If you need further special handling of source maps, such as using with gulp-sourcemaps then just pipe to a stream and handle the source map files emitted by webpack:

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
const named = require('vinyl-named');
const through = require('through2');
const sourcemaps = require('gulp-sourcemaps');
gulp.task('default', function() {
  return gulp.src(['src/app.js', 'test/test.js'])
```

```
    .pipe(named())
    .pipe(webpack({
      devtool: 'source-map'
    }))
    .pipe(sourcemaps.init({loadMaps: true}))
    .pipe(through.obj(function (file, enc, cb) {
      // Dont pipe through any source map files as it will be ha
      // by gulp-sourcemaps
      const isSourceMap = /\.map$/.test(file.path);
      if (!isSourceMap) this.push(file);
      cb();
    }))
    .pipe(sourcemaps.write('.'))
    .pipe(gulp.dest('dist/'));
});
```

Multi-compiler support

Multiple compilers are supported, but instead of passing the webpack configuration directly, you have to wrap it in an object under the key 'config'.

```
const gulp = require('gulp');
const webpack = require('webpack-stream');
gulp.task('default', function() {
  return gulp.src('src/entry.js')
    .pipe(webpack({
      config : require('./webpack.config.js')
    }))
    .pipe(gulp.dest('dist/'));
});
```

Release History

- Please check the commit log in the future for release history.