Command Line Interface

For proper usage and easier distribution of this configuration, webpack can be configured with `webpack.config.js`. Any parameters sent to the CLI will map to a corresponding parameter in the configuration file.

Read the installation guide if you don't already have webpack and CLI installed.

warning

`webpack-cli` v5.0.0+ requires node >= v14.15.0, `webpack >= v5.0.0`, and `webpack-dev-server >= v4.0.0`.

warning

If you want to run webpack using `npx` please make sure you have `webpack-cli` installed.

Commands

webpack-cli offers a variety of commands to make working with webpack easier. By default webpack ships with

Command	Usage	Description
`build`	`build|bundle|b [entries...] [options]`	Run webpack (default command, can be omitted).
`configtest`	`configtest|t [config-path]`	Validate a webpack configuration.
`help`	`help|h [command] [option]`	Display help for commands and options.
`info`	`info|i [options]`	Outputs information about your system.

`init`	`init|create|c|new|n [generation-path] [options]`	Initialize a new webpack project.
`loader`	`loader|l [output-path] [options]`	Scaffold a loader.
`plugin`	`plugin|p [output-path] [options]`	Scaffold a plugin.
`serve`	`serve|server|s [options]`	Run the `webpack-dev-server`.
`version`	`version|v [commands...]`	Output the version number of `webpack`, `webpack-cli` and `webpack-dev-server`.
`watch`	`watch|w [entries...] [options]`	Run webpack and watch for files changes.

Build

Run webpack (default command, can be omitted).

```
npx webpack build [options]
```

example

```
npx webpack build --config ./webpack.config.js --stats verbose
```

Init

Used to initialize a new webpack project.

```
npx webpack init [generation-path] [options]
```

example

```
npx webpack init ./my-app --force --template=default
```

Generation Path

Location of where to generate the configuration. Defaults to `process.cwd()`.

Options

`-t, --template`

`string = 'default'`

Name of template to generate.

`-f, --force`

`boolean`

To generate a project without questions. When enabled, the default answer for each question will be used.

tip

See the full documentation of `webpack init` command.

Loader

Scaffold a loader.

```
npx webpack loader [output-path] [options]
```

example

```
npx webpack loader ./my-loader --template=default
```

Output Path

Path to the output directory, e.g. `./loader-name`.

Options

`-t, --template`

```
string = 'default'
```

Type of template.

Plugin

Scaffold a plugin.

```
npx webpack plugin [output-path] [options]
```

example

```
npx webpack plugin ./my-plugin --template=default
```

Output Path

Path to the output directory, e.g. `./plugin-name`.

Options

`-t, --template`

```
string = 'default'
```

Type of template.

Info

Outputs information about your system.

```
npx webpack info [options]
```

example

```
npx webpack info --output json --addition-package postcss
```

Options for info

-a, --additional-package

```
string
```

Adds additional packages to the output.

example

```
npx webpack info --additional-package postcss
```

-o, --output

```
string : 'json' | 'markdown'
```

To get the output in a specified format.

example

```
npx webpack info --output markdown
```

Configtest

Validate a webpack configuration.

```
npx webpack configtest [config-path]
```

example

```
npx webpack configtest ./webpack.config.js
```

Config Path

Path to your webpack configuration file. Defaults to `./webpack.config.js`.

Serve

Run the webpack dev server.

```
npx webpack serve [options]
```

example

```
npx webpack serve --static --open
```

tip

See the full list of options for `webpack serve` command and related documentation for webpack-dev-server.

Watch

Run webpack and watch for files changes.

```
npx webpack watch [options]
```

example

```
npx webpack watch --mode development
```

Flags

By default webpack ships with the following flags:

Flag / Alias	Type	Description
`--entry`	string[]	The entry point(s) of your application e.g. `./src/main.js`
`--config, -c`	string[]	Provide path to a webpack configuration file e.g. `./webpack.config.js`
`--config-name`	string[]	Name of the configuration to use
`--name`	string	Name of the configuration. Used when loading multiple configurations
`--color`	boolean	Enable colors on console
`--merge, -m`	boolean	Merge two or more configurations using `webpack-merge`
`--env`	string[]	Environment passed to the configuration when it is a function
`--define-process-env-node-env`	string	Set `process.env.NODE_ENV` to the specified value

`--progress`	boolean, string	Print compilation progress during build
`--help`	boolean	Outputs list of supported flags and commands
`--output-path, -o`	string	Output location of the file generated by webpack e.g. `./dist`
`--target, -t`	string[]	Sets the build target
`--watch, -w`	boolean	Watch for file changes
`--watch-options-stdin`	boolean	Stop watching when stdin stream has ended
`--devtool, -d`	string	Controls if and how source maps are generated.
`--json, -j`	boolean, string	Prints result as JSON or store it in a file
`--mode`	string	Defines the mode to pass to webpack
`--version, -v`	boolean	Get current version
`--stats`	boolean, string	It instructs webpack on how to treat the stats
`--disable-interpret`	boolean	Disable interpret for loading the config file.
`--fail-on-warnings`	boolean	Stop webpack-cli process with non-zero exit code on warnings from webpack
`--analyze`	boolean	It invokes `webpack-bundle-analyzer` plugin to get bundle information
`--extends, -e`	string[]	Extend an existing configuration

Negated Flags

Flag	Description
`--no-color`	Disables any color on the console
`--no-hot`	Disables hot reloading if you have it enabled via your config
`--no-stats`	Disables any compilation stats emitted by webpack
`--no-watch`	Do not watch for file changes

`--no-devtool`	Do not generate source maps
`--no-watch-options-stdin`	Do not stop watching when stdin stream has ended

Core Flags

Starting CLI v4 and webpack v5, CLI imports the entire configuration schema from webpack core to allow tuning almost every configuration option from the command line.

Here's the list of all the core flags supported by webpack v5 with CLI v4 - link

For example if you want to enable performance hints in your project you'd use this option in configuration, with core flags you can do -

```
npx webpack --performance-hints warning
```

Usage

With configuration file

```
npx webpack [--config webpack.config.js]
```

See configuration for the options in the configuration file.

Without configuration file

```
npx webpack --entry <entry> --output-path <output-path>
```

example

```
npx webpack --entry ./first.js --entry ./second.js --output-path /build
```

entry

A filename or a set of named filenames which act as the entry point to build your project. You can pass multiple entries (every entry is loaded on startup). Following are the multiple ways of specifying entry file(s) via CLI -

```
npx webpack --entry-reset ./first-entry.js
```

```
npx webpack --entry-reset --entry ./first-entry.js
```

```
npx webpack --entry-reset ./first-entry.js ./other-entry.js
```

```
npx webpack --entry-reset --entry ./first-entry.js ./other-entry.js
```

warning

The `--entry-reset` option is required to replace the existing entry option, without it the `--entry` option will add another entry to the existing entries.

tip

Use `webpack [command] --entry-reset [entries...] [option]` syntax because some options can accept multiple values so `webpack --target node ./entry.js` means `target: ['node', './entry.js']`

output-path

A path for the bundled file to be saved in. It will be mapped to the configuration options `output.path`.

Example

If your project structure is as follows -

```
.
├── dist
├── index.html
└── src
    ├── index.js
    ├── index2.js
    └── others.js
```

```
npx webpack ./src/index.js --output-path dist
```

This will bundle your source code with entry as `index.js`, and the output bundle file will have a path of `dist`.

```
asset main.js 142 bytes [compared for emit] [minimized] (name: main)
./src/index.js 30 bytes [built] [code generated]
./src/others.js 1 bytes [built] [code generated]
webpack 5.1.0 compiled successfully in 187 ms
```

```
npx webpack ./src/index.js ./src/others2.js --output-path dist/
```

This will form the bundle with both the files as separate entry points.

```
asset main.js 142 bytes [compared for emit] [minimized] (name: main)
./src/index.js 30 bytes [built] [code generated]
```

```
./src/others2.js 1 bytes [built] [code generated]
./src/others.js 1 bytes [built] [code generated]
webpack 5.1.0 compiled successfully in 198 ms
```

Default Configurations

CLI will look for some default configurations in the path of your project, here are the config files picked up by CLI.

This is the lookup priority in increasing order

> *example - config file lookup will be in order of .webpack/webpackfile > .webpack/webpack.config.js > webpack.config.js*

```
'webpack.config',
'.webpack/webpack.config',
'.webpack/webpackfile',
```

Common Options

warning

Note that Command Line Interface has a higher precedence for the arguments you use it with than your configuration file. For instance, if you pass `--mode="production"` to webpack CLI and your configuration file uses `development`, `production` will be used.

help

List basic commands and flags available on the cli

Both `webpack help [command] [option]` and `webpack [command] --help` are

valid to get help:

```
npx webpack --help
```

```
npx webpack help
```

List all supported commands and flags by cli

```
npx webpack --help=verbose
```

See help for a specific command or option

```
npx webpack help --mode
```

version

Show version of installed packages and sub-packages

To inspect the version of `webpack` and `webpack-cli` you are using, run the command:

```
npx webpack --version
```

```
npx webpack version
```

This will output the following result:

```
webpack 5.31.2
webpack-cli 4.6.0
```

It will output the version of `webpack-dev-server` as well if you have it installed:

```
webpack 5.31.2
webpack-cli 4.6.0
webpack-dev-server 3.11.2
```

To inspect the version of any `webpack-cli` sub-package (like `@webpack-cli/info`), run command similar to the following:

```
npx webpack info --version
```

This will output the following result:

```
@webpack-cli/info 1.2.3
webpack 5.31.2
webpack-cli 4.6.0
webpack-dev-server 3.11.2
```

config

Build source using a configuration file

Specify a different configuration file other than `webpack.config.js`, which is one of the defaults.

```
npx webpack --config example.config.js
```

config-name

In case your configuration file exports multiple configurations, you can use `--config-name` to specify which configuration to run.

Consider the following `webpack.config.js`:

```
module.exports = [
  {
    output: {
      filename: './dist-first.js',
    },
    name: 'first',
    entry: './src/first.js',
    mode: 'development',
  },
  {
    output: {
      filename: './dist-second.js',
    },
    name: 'second',
    entry: './src/second.js',
    mode: 'development',
  },
  {
    output: {
      filename: './dist-third.js',
    },
    name: 'third',
    entry: './src/third.js',
    mode: 'none',
```

```
    stats: 'verbose',
  },
];
```

To run only the `second` configuration:

```
npx webpack --config-name second
```

You can also pass multiple values:

```
npx webpack --config-name first --config-name second
```

merge

You can merge two or more different webpack configurations with the help of `--merge`:

```
npx webpack --config ./first.js --config ./second.js --merge
```

extends

webpack-cli v5.1.0+

You can extend existing webpack configurations with the help of `--extends`:

```
npx webpack --extends ./base.webpack.config.js
```

Read more about it in extending configurations.

json

Print result of webpack as JSON

```
npx webpack --json
```

If you want to store stats as json instead of printing it, you can use -

```
npx webpack --json stats.json
```

In every other case, webpack prints out a set of stats showing bundle, chunk and timing details. Using this option, the output can be a JSON object. This response is accepted by webpack's analyse tool, or chrisbateman's webpack-visualizer, or th0r's webpack-bundle-analyzer. The analyse tool will take in the JSON and provide all the details of the build in graphical form.

tip

See the stats data api to read more about the stats generated here.

Environment Options

When the webpack configuration exports a function, an "environment" may be passed to it.

env

```
npx webpack --env production
```

The `--env` argument accepts multiple values:

Invocation	Resulting environment
`npx webpack --env prod`	`{ prod: true }`
`npx webpack --env prod --env min`	`{ prod: true, min: true }`
`npx webpack --env platform=app --env production`	`{ platform: "app", production: true }`
`npx webpack --env foo=bar=app`	`{ foo: "bar=app"}`
`npx webpack --env app.platform="staging" --env app.name="test"`	`{ app: { platform: "staging", name: "test" }`

tip

If you want to explicitly set a variable to empty string (""), you may need to escape characters on terminal like `npx webpack --env foo=\"\"`

tip

See the environment variables guide for more information on its usage.

In addition to the customized `env` showed above, there are some built-in ones under `env` to be used inside your webpack configuration:

Environment Variable	Description
`WEBPACK_SERVE`	`true` if `serve|server|s` is being used.
`WEBPACK_BUILD`	`true` if `build|bundle|b` is being used.
`WEBPACK_WATCH`	`true` if `--watch|watch|w` is being used.

Note that you can not access those built-in environment variables inside the bundled code.

```
module.exports = (env, argv) => {
  return {
    mode: env.WEBPACK_SERVE ? 'development' : 'production',
  };
```

```
};
```

node-env

You can use `--node-env` option to set `process.env.NODE_ENV`, which is available to both user code and webpack configuration:

```
npx webpack --node-env production
```

warning

This option is deprecated and removed in webpack-cli v5 in favor of the `--define-process-env-node-env` option.

define-process-env-node-env

For `webpack-cli v5+`.

An alias for `--node-env` to set `process.env.NODE_ENV`:

```
npx webpack --define-process-env-node-env production
```

When the `mode` option is not specified in the configuration, you can use the `--define-process-env-node-env` option to set the `mode`. For example, using `--define-process-env-node-env production` will set both `process.env.NODE_ENV` and `mode` to `'production'`.

If your configuration exports a function, the value of `--define-process-env-node-env` is assigned to mode after the function returns. This means that

`mode` will not be available in the function arguments (`env` and `argv`). However, the value of `--define-process-env-node-env` is accessible as `argv.nodeEnv` within the function and can be used accordingly.

```
module.exports = (env, argv) => {
  console.log(argv.defineProcessEnvNodeEnv);
  return {

  };
};
```

Configuration Options

Parameter	Explanation	Input type	Default
`--config`	Path to the configuration file	string[]	Default Configs
`--config-name`	Name of the configuration to use	string[]	-
`--env`	Environment passed to the configuration, when it is a function	string[]	-

Analyzing Bundle

You can also use `webpack-bundle-analyzer` to analyze your output bundles emitted by webpack. You can use `--analyze` flag to invoke it via CLI.

```
npx webpack --analyze
```

warning

Make sure you have `webpack-bundle-analyzer` installed in your project else CLI will prompt you to install it.

Progress

To check the progress of any webpack compilation you can use the `--progress` flag.

```
npx webpack --progress
```

To collect profile data for progress steps you can pass `profile` as value to `--progress` flag.

```
npx webpack --progress=profile
```

Pass CLI arguments to Node.js

To pass arguments directly to Node.js process, you can use the `NODE_OPTIONS` option.

For example, to increase the memory limit of Node.js process to 4 GB

```
NODE_OPTIONS="--max-old-space-size=4096" webpack
```

Also, you can pass multiple options to Node.js process

```
NODE_OPTIONS="--max-old-space-size=4096 -r /path/to/preload/file.js" webpack
```

Exit codes and their meanings

Exit Code	Description
`0`	Success
`1`	Errors from webpack
`2`	Configuration/options problem or an internal error

CLI Environment Variables

Environment Variable	Description
`WEBPACK_CLI_SKIP_IMPORT_LOCAL`	when `true` it will skip using the local instance of `webpack-cli`.
`WEBPACK_CLI_FORCE_LOAD_ESM_CONFIG`	when `true` it will force load the ESM config.
`WEBPACK_PACKAGE`	Use a custom webpack version in CLI.
`WEBPACK_DEV_SERVER_PACKAGE`	Use a custom webpack-dev-server version in CLI.
`WEBPACK_CLI_HELP_WIDTH`	Use a custom width for help output.

```
WEBPACK_CLI_FORCE_LOAD_ESM_CONFIG=true npx webpack --config ./webpack.config.esm
```

WEBPACK_PACKAGE

Use a custom webpack version in CLI. Considering the following content in your `package.json`:

```
{
  "webpack": "^4.0.0",
  "webpack-5": "npm:webpack@^5.32.0",
  "webpack-cli": "^4.5.0"
}
```

To use `webpack v4.0.0`:

```
npx webpack
```

To use `webpack v5.32.0`:

```
WEBPACK_PACKAGE=webpack-5 npx webpack
```

Troubleshooting

TypeError [ERR_UNKNOWN_FILE_EXTENSION]: Unknown file extension ".ts" for ./webpack.config.ts

You might encounter this error in the case of using native ESM in TypeScript (i.e. `type: "module"` in `package.json`).

`webpack-cli` supports configuration in both `CommonJS` and `ESM` format, at first it tries to load a configuration using `require()`, once it fails with an error code of `'ERR_REQUIRE_ESM'` (a special code for this case) it would try to load the configuration using `import()`. However, the `import()` method won't work with `ts-node` without loader hooks enabled (described at `TypeStrong/ts-node#1007`).

To fix the error above use the following command:

```
NODE_OPTIONS="--loader ts-node/esm" npx webpack --entry ./src/index.js --mode pr
```

For more information, see our documentation on writing a webpack configuration in `TypeScript`.