You are reading webpack 5 documentation. Change here to:

webpack 4 documentation

This option controls if and how source maps are generated.

Use the `SourceMapDevToolPlugin` for a more fine grained configuration. See the `source-map-loader` to deal with existing source maps.

`devtool`

`string` `false`

Choose a style of source mapping to enhance the debugging process. These values can affect build and rebuild speed dramatically.

The webpack repository contains an example showing the effect of all `devtool` variants. Those examples will likely help you to understand the differences.

Instead of using the `devtool` option you can also use `SourceMapDevToolPlugin` / `EvalSourceMapDevToolPlugin` directly as it has more options. Never use both the `devtool` option and plugin together. The `devtool` option adds the plugin internally so you would end up with the plugin applied twice.

devtool	build	rebuild	production	quality
(none)	fastest	fastest	yes	bundled code
eval	fastest	fastest	no	generated code
eval-cheap-source-map	fast	faster	no	transformed code (lines only)
eval-cheap-module-source-map	slow	faster	no	original source (lines only)
eval-source-map	slowest	fast	no	original source
eval-nosources-source-map				
eval-nosources-cheap-source-map				
eval-nosources-cheap-module-source-map				
cheap-source-map	fast	slow	yes	transformed code (lines only)
cheap-module-source-map	slow	slower	yes	original source (lines only)
inline-cheap-source-map	fast	slow	no	transformed code (lines only)
inline-cheap-module-source-map	slow	slower	no	original sourc (lines only)

› Watch and WatchOptions
› Externals

devtool	build	rebuild	production	quality
inline-source-map	slowest	slowest	no	original source
inline-nosources-source-map				
inline-nosources-cheap-source-map				
inline-nosources-cheap-module-source-map				
source-map	slowest	slowest	yes	original source
hidden-source-map	slowest	slowest	yes	original source
hidden-nosources-source-map				
hidden-nosources-cheap-source-map				
hidden-nosources-cheap-module-source-map				
hidden-cheap-source-map				
hidden-cheap-module-source-map				
nosources-source-map	slowest	slowest	yes	without source content
nosources-cheap-source-map				
nosources-cheap-module-source-map				

We expect a certain pattern when validate devtool name, pay attention and dont mix up the sequence of devtool string. The pattern is: `[inline-|hidden-|eval-][nosources-][cheap-[module-]]source-map` *.*

Some of these values are suited for development and some for production. For development you typically want fast Source Maps at the cost of bundle size, but for production you want separate Source Maps that are accurate and support minimizing.

There are some issues with Source Maps in Chrome. We need your help!.

See `output.sourceMapFilename` *to customize the filenames of generated Source Maps.*

Qualities

`bundled code` - You see all generated code as a big blob of code. You don't see modules separated from each other.

`generated code` - You see each module separated from each other, annotated with module names. You see the code generated by webpack. Example: Instead of `import {test} from "module"; test();` you see something like `var module__WEBPACK_IMPORTED_MODULE_1__ = __webpack_require__(42); module__WEBPACK_IMPORTED_MODULE_1__.a();` .

`transformed code` - You see each module separated from each other, annotated with module names. You see the code before webpack transforms it, but after Loaders transpile it. Example: Instead of `import {test} from "module"; class A extends test {}` you see something like `import {test} from "module"; var A = function(_test) { ... }(test);`

`original source` - You see each module separated from each other, annotated with module names. You see the code before transpilation, as you authored it. This depends on Loader support.

`without source content` - Contents for the sources are not included in the Source Maps. Browsers usually try to load the source from the webserver or filesystem. You have to make sure to set `output.devtoolModuleFilenameTemplate` correctly to match source urls.

`(lines only)` - Source Maps are simplified to a single mapping per line. This usually means a single mapping per statement (assuming you author it this way). This prevents you from debugging execution on statement level and from settings breakpoints on columns of a line. Combining with minimizing is not possible as minimizers usually only emit a single line.

Development

The following options are ideal for development:

`eval` - Each module is executed with `eval()` and `//@ sourceURL`. This is pretty fast. The main disadvantage is that it doesn't display line numbers correctly since it gets mapped to transpiled code instead of the original code (No Source Maps from Loaders).

`eval-source-map` - Each module is executed with `eval()` and a SourceMap is added as a DataUrl to the `eval()`. Initially it is slow, but it provides fast rebuild speed and yields real files. Line numbers are correctly mapped since it gets mapped to the original code. It yields the best quality SourceMaps for development.

`eval-cheap-source-map` - Similar to `eval-source-map`, each module is executed with `eval()`. It is "cheap" because it doesn't have column mappings, it only maps line numbers. It ignores SourceMaps from Loaders and only display transpiled code similar to the `eval` devtool.

`eval-cheap-module-source-map` - Similar to `eval-cheap-source-map`, however, in this case Source Maps from Loaders are processed for better results. However Loader Source Maps are simplified to a single mapping per line.

Special cases

The following options are not ideal for development nor production. They are needed for some special cases, i. e. for some 3rd party tools.

`inline-source-map` - A SourceMap is added as a DataUrl to the bundle.

`cheap-source-map` - A SourceMap without column-mappings ignoring loader Source Maps.

`inline-cheap-source-map` - Similar to `cheap-source-map` but SourceMap is added as a DataUrl to the bundle.

`cheap-module-source-map` - A SourceMap without column-mappings that simplifies loader Source Maps to a single mapping per line.

`inline-cheap-module-source-map` - Similar to `cheap-module-source-map` but SourceMap is added as a DataUrl to the bundle.

Production

These options are typically used in production:

`(none)` (Omit the `devtool` option) - No SourceMap is emitted. This is a good option to start with.

`source-map` - A full SourceMap is emitted as a separate file. It adds a reference comment to the bundle so development tools know where to find it.

> *You should configure your server to disallow access to the Source Map file for normal users!*

`hidden-source-map` - Same as `source-map`, but doesn't add a reference comment to the bundle. Useful if you only want SourceMaps to map error stack traces from error reports, but don't want to expose your SourceMap for the browser development tools.

> *You should not deploy the Source Map file to the webserver. Instead only use it for error report tooling.*

`nosources-source-map` - A SourceMap is created without the `sourcesContent` in it. It can be used to map stack traces on the client without exposing all of the source code. You can deploy the Source Map file to the webserver.

> *It still exposes filenames and structure for decompiling, but it doesn't expose the original code.*

> *If the default webpack `minimizer` has been overridden (such as to customise the `terser-webpack-plugin` options), make sure to configure its replacement with `sourceMap: true` to enable SourceMap support.*

« Previous
DevServer

Next »
Target

Further Reading

- Enabling Source Maps
- webpack's Devtool Source Map

Contributors


sokra


skipjack


SpaceK33z


lricoy


madhavarshney


wizardofhogwarts


anikethsaha