# @jridgewell/gen-mapping

Generate source maps

gen-mapping allows you to generate a source map during transpilation or minification. With a source map, you're able to trace the original location in the source file, either in Chrome's DevTools or using a library like [@jridgewell/trace-mapping](https://github.com/jridgewell/trace-mapping).

You may already be familiar with the [source-map](https://www.npmjs.com/package/source-map) package's SourceMapGenerator. This provides the same addMapping and setSourceContent API.

## Installation

npm install @jridgewell/gen-mapping

## Usage

import { GenMapping, addMapping, setSourceContent, toEncodedMap, toDecodedMap } from '@jridgewell/gen-mapping';

const map = new GenMapping({

file: 'output.js',

sourceRoot: 'https://example.com/',

});

setSourceContent(map, 'input.js', `function foo() {}`);

addMapping(map, {

// Lines start at line 1, columns at column 0.

generated: { line: 1, column: 0 },

source: 'input.js',

original: { line: 1, column: 0 },

});

addMapping(map, {

generated: { line: 1, column: 9 },

source: 'input.js',

original: { line: 1, column: 9 },

name: 'foo',

});

assert.deepEqual(toDecodedMap(map), {

version: 3,

file: 'output.js',

names: ['foo'],

sourceRoot: 'https://example.com/',

sources: ['input.js'],

sourcesContent: ['function foo() {}'],

mappings: [

[ [0, 0, 0, 0], [9, 0, 0, 9, 0] ]

],

});

assert.deepEqual(toEncodedMap(map), {

version: 3,

file: 'output.js',

names: ['foo'],

sourceRoot: 'https://example.com/',

sources: ['input.js'],

sourcesContent: ['function foo() {}'],

mappings: 'AAAA,SAASA',

});

### Smaller Sourcemaps

Not everything needs to be added to a sourcemap, and needless markings can cause signficantly larger file sizes. gen-mapping exposes maybeAddSegment/maybeAddMapping APIs that will intelligently determine if this marking adds useful information. If not, the marking will be skipped.

import { maybeAddMapping } from '@jridgewell/gen-mapping';

const map = new GenMapping();

// Adding a sourceless marking at the beginning of a line isn't useful.

maybeAddMapping(map, {

generated: { line: 1, column: 0 },

});

// Adding a new source marking is useful.

maybeAddMapping(map, {

generated: { line: 1, column: 0 },

source: 'input.js',

original: { line: 1, column: 0 },

});

// But adding another marking pointing to the exact same original location isn't, even if the

// generated column changed.

maybeAddMapping(map, {

generated: { line: 1, column: 9 },

source: 'input.js',

original: { line: 1, column: 0 },

});

assert.deepEqual(toEncodedMap(map), {

version: 3,

names: [],

sources: ['input.js'],

sourcesContent: [null],

mappings: 'AAAA',

});

## Benchmarks

node v18.0.0

amp.js.map

Memory Usage:

gen-mapping: addSegment 5852872 bytes

gen-mapping: addMapping 7716042 bytes

source-map-js 6143250 bytes

source-map-0.6.1 6124102 bytes

source-map-0.8.0 6121173 bytes

Smallest memory usage is gen-mapping: addSegment

Adding speed:

gen-mapping: addSegment x 441 ops/sec ±2.07% (90 runs sampled)

gen-mapping: addMapping x 350 ops/sec ±2.40% (86 runs sampled)

source-map-js: addMapping x 169 ops/sec ±2.42% (80 runs sampled)

source-map-0.6.1: addMapping x 167 ops/sec ±2.56% (80 runs sampled)

source-map-0.8.0: addMapping x 168 ops/sec ±2.52% (80 runs sampled)

Fastest is gen-mapping: addSegment

Generate speed:

gen-mapping: decoded output x 150,824,370 ops/sec ±0.07% (102 runs sampled)

gen-mapping: encoded output x 663 ops/sec ±0.22% (98 runs sampled)

source-map-js: encoded output x 197 ops/sec ±0.45% (84 runs sampled)

source-map-0.6.1: encoded output x 198 ops/sec ±0.33% (85 runs sampled)

source-map-0.8.0: encoded output x 197 ops/sec ±0.06% (93 runs sampled)

Fastest is gen-mapping: decoded output

\*\*\*

babel.min.js.map

Memory Usage:

gen-mapping: addSegment 37578063 bytes

gen-mapping: addMapping 37212897 bytes

source-map-js 47638527 bytes

source-map-0.6.1 47690503 bytes

source-map-0.8.0 47470188 bytes

Smallest memory usage is gen-mapping: addMapping

Adding speed:

gen-mapping: addSegment x 31.05 ops/sec ±8.31% (43 runs sampled)

gen-mapping: addMapping x 29.83 ops/sec ±7.36% (51 runs sampled)

source-map-js: addMapping x 20.73 ops/sec ±6.22% (38 runs sampled)

source-map-0.6.1: addMapping x 20.03 ops/sec ±10.51% (38 runs sampled)

source-map-0.8.0: addMapping x 19.30 ops/sec ±8.27% (37 runs sampled)

Fastest is gen-mapping: addSegment

Generate speed:

gen-mapping: decoded output x 381,379,234 ops/sec ±0.29% (96 runs sampled)

gen-mapping: encoded output x 95.15 ops/sec ±2.98% (72 runs sampled)

source-map-js: encoded output x 15.20 ops/sec ±7.41% (33 runs sampled)

source-map-0.6.1: encoded output x 16.36 ops/sec ±10.46% (31 runs sampled)

source-map-0.8.0: encoded output x 16.06 ops/sec ±6.45% (31 runs sampled)

Fastest is gen-mapping: decoded output

\*\*\*

preact.js.map

Memory Usage:

gen-mapping: addSegment 416247 bytes

gen-mapping: addMapping 419824 bytes

source-map-js 1024619 bytes

source-map-0.6.1 1146004 bytes

source-map-0.8.0 1113250 bytes

Smallest memory usage is gen-mapping: addSegment

Adding speed:

gen-mapping: addSegment x 13,755 ops/sec ±0.15% (98 runs sampled)

gen-mapping: addMapping x 13,013 ops/sec ±0.11% (101 runs sampled)

source-map-js: addMapping x 4,564 ops/sec ±0.21% (98 runs sampled)

source-map-0.6.1: addMapping x 4,562 ops/sec ±0.11% (99 runs sampled)

source-map-0.8.0: addMapping x 4,593 ops/sec ±0.11% (100 runs sampled)

Fastest is gen-mapping: addSegment

Generate speed:

gen-mapping: decoded output x 379,864,020 ops/sec ±0.23% (93 runs sampled)

gen-mapping: encoded output x 14,368 ops/sec ±4.07% (82 runs sampled)

source-map-js: encoded output x 5,261 ops/sec ±0.21% (99 runs sampled)

source-map-0.6.1: encoded output x 5,124 ops/sec ±0.58% (99 runs sampled)

source-map-0.8.0: encoded output x 5,434 ops/sec ±0.33% (96 runs sampled)

Fastest is gen-mapping: decoded output

\*\*\*

react.js.map

Memory Usage:

gen-mapping: addSegment 975096 bytes

gen-mapping: addMapping 1102981 bytes

source-map-js 2918836 bytes

source-map-0.6.1 2885435 bytes

source-map-0.8.0 2874336 bytes

Smallest memory usage is gen-mapping: addSegment

Adding speed:

gen-mapping: addSegment x 4,772 ops/sec ±0.15% (100 runs sampled)

gen-mapping: addMapping x 4,456 ops/sec ±0.13% (97 runs sampled)

source-map-js: addMapping x 1,618 ops/sec ±0.24% (97 runs sampled)

source-map-0.6.1: addMapping x 1,622 ops/sec ±0.12% (99 runs sampled)

source-map-0.8.0: addMapping x 1,631 ops/sec ±0.12% (100 runs sampled)

Fastest is gen-mapping: addSegment

Generate speed:

gen-mapping: decoded output x 379,107,695 ops/sec ±0.07% (99 runs sampled)

gen-mapping: encoded output x 5,421 ops/sec ±1.60% (89 runs sampled)

source-map-js: encoded output x 2,113 ops/sec ±1.81% (98 runs sampled)

source-map-0.6.1: encoded output x 2,126 ops/sec ±0.10% (100 runs sampled)

source-map-0.8.0: encoded output x 2,176 ops/sec ±0.39% (98 runs sampled)

Fastest is gen-mapping: decoded output