# @jridgewell/trace-mapping

Trace the original position through a source map

trace-mapping allows you to take the line and column of an output file and trace it to the original location in the source file through a source map.

You may already be familiar with the [source-map](https://www.npmjs.com/package/source-map) package's SourceMapConsumer. This provides the same originalPositionFor and generatedPositionFor API, without requiring WASM.

## Installation

npm install @jridgewell/trace-mapping

## Usage

import {

TraceMap,

originalPositionFor,

generatedPositionFor,

sourceContentFor,

isIgnored,

} from '@jridgewell/trace-mapping';

const tracer = new TraceMap({

version: 3,

sources: ['input.js'],

sourcesContent: ['content of input.js'],

names: ['foo'],

mappings: 'KAyCIA',

ignoreList: [],

});

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

const traced = originalPositionFor(tracer, { line: 1, column: 5 });

assert.deepEqual(traced, {

source: 'input.js',

line: 42,

column: 4,

name: 'foo',

});

const content = sourceContentFor(tracer, traced.source);

assert.strictEqual(content, 'content for input.js');

const generated = generatedPositionFor(tracer, {

source: 'input.js',

line: 42,

column: 4,

});

assert.deepEqual(generated, {

line: 1,

column: 5,

});

const ignored = isIgnored(tracer, 'input.js');

assert.equal(ignored, false);

We also provide a lower level API to get the actual segment that matches our line and column. Unlike originalPositionFor, traceSegment uses a 0-base for line:

import { traceSegment } from '@jridgewell/trace-mapping';

// line is 0-base.

const traced = traceSegment(tracer, /\* line \*/ 0, /\* column \*/ 5);

// Segments are [outputColumn, sourcesIndex, sourceLine, sourceColumn, namesIndex]

// Again, line is 0-base and so is sourceLine

assert.deepEqual(traced, [5, 0, 41, 4, 0]);

### SectionedSourceMaps

The sourcemap spec defines a special sections field that's designed to handle concatenation of output code with associated sourcemaps. This type of sourcemap is rarely used (no major build tool produces it), but if you are hand coding a concatenation you may need it. We provide an AnyMap helper that can receive either a regular sourcemap or a SectionedSourceMap and returns a TraceMap instance:

import { AnyMap } from '@jridgewell/trace-mapping';

const fooOutput = 'foo';

const barOutput = 'bar';

const output = [fooOutput, barOutput].join('\n');

const sectioned = new AnyMap({

version: 3,

sections: [

{

// 0-base line and column

offset: { line: 0, column: 0 },

// fooOutput's sourcemap

map: {

version: 3,

sources: ['foo.js'],

names: ['foo'],

mappings: 'AAAAA',

},

},

{

// barOutput's sourcemap will not affect the first line, only the second

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

map: {

version: 3,

sources: ['bar.js'],

names: ['bar'],

mappings: 'AAAAA',

},

},

],

});

const traced = originalPositionFor(sectioned, {

line: 2,

column: 0,

});

assert.deepEqual(traced, {

source: 'bar.js',

line: 1,

column: 0,

name: 'bar',

});

## Benchmarks

node v18.0.0

amp.js.map - 45120 segments

Memory Usage:

trace-mapping decoded 562400 bytes

trace-mapping encoded 5706544 bytes

source-map-js 10717664 bytes

source-map-0.6.1 17446384 bytes

source-map-0.8.0 9701757 bytes

Smallest memory usage is trace-mapping decoded

Init speed:

trace-mapping: decoded JSON input x 180 ops/sec ±0.34% (85 runs sampled)

trace-mapping: encoded JSON input x 364 ops/sec ±1.77% (89 runs sampled)

trace-mapping: decoded Object input x 3,116 ops/sec ±0.50% (96 runs sampled)

trace-mapping: encoded Object input x 410 ops/sec ±2.62% (85 runs sampled)

source-map-js: encoded Object input x 84.23 ops/sec ±0.91% (73 runs sampled)

source-map-0.6.1: encoded Object input x 37.21 ops/sec ±2.08% (51 runs sampled)

Fastest is trace-mapping: decoded Object input

Trace speed:

trace-mapping: decoded originalPositionFor x 3,952,212 ops/sec ±0.17% (98 runs sampled)

trace-mapping: encoded originalPositionFor x 3,487,468 ops/sec ±1.58% (90 runs sampled)

source-map-js: encoded originalPositionFor x 827,730 ops/sec ±0.78% (97 runs sampled)

source-map-0.6.1: encoded originalPositionFor x 748,991 ops/sec ±0.53% (94 runs sampled)

source-map-0.8.0: encoded originalPositionFor x 2,532,894 ops/sec ±0.57% (95 runs sampled)

Fastest is trace-mapping: decoded originalPositionFor

\*\*\*

babel.min.js.map - 347793 segments

Memory Usage:

trace-mapping decoded 89832 bytes

trace-mapping encoded 35474640 bytes

source-map-js 51257176 bytes

source-map-0.6.1 63515664 bytes

source-map-0.8.0 42933752 bytes

Smallest memory usage is trace-mapping decoded

Init speed:

trace-mapping: decoded JSON input x 15.41 ops/sec ±8.65% (34 runs sampled)

trace-mapping: encoded JSON input x 28.20 ops/sec ±12.87% (42 runs sampled)

trace-mapping: decoded Object input x 964 ops/sec ±0.36% (99 runs sampled)

trace-mapping: encoded Object input x 31.77 ops/sec ±13.79% (45 runs sampled)

source-map-js: encoded Object input x 6.45 ops/sec ±5.16% (21 runs sampled)

source-map-0.6.1: encoded Object input x 4.07 ops/sec ±5.24% (15 runs sampled)

Fastest is trace-mapping: decoded Object input

Trace speed:

trace-mapping: decoded originalPositionFor x 7,183,038 ops/sec ±0.58% (95 runs sampled)

trace-mapping: encoded originalPositionFor x 5,192,185 ops/sec ±0.41% (100 runs sampled)

source-map-js: encoded originalPositionFor x 4,259,489 ops/sec ±0.79% (94 runs sampled)

source-map-0.6.1: encoded originalPositionFor x 3,742,629 ops/sec ±0.71% (95 runs sampled)

source-map-0.8.0: encoded originalPositionFor x 6,270,211 ops/sec ±0.64% (94 runs sampled)

Fastest is trace-mapping: decoded originalPositionFor

\*\*\*

preact.js.map - 1992 segments

Memory Usage:

trace-mapping decoded 37128 bytes

trace-mapping encoded 247280 bytes

source-map-js 1143536 bytes

source-map-0.6.1 1290992 bytes

source-map-0.8.0 96544 bytes

Smallest memory usage is trace-mapping decoded

Init speed:

trace-mapping: decoded JSON input x 3,483 ops/sec ±0.30% (98 runs sampled)

trace-mapping: encoded JSON input x 6,092 ops/sec ±0.18% (97 runs sampled)

trace-mapping: decoded Object input x 249,076 ops/sec ±0.24% (98 runs sampled)

trace-mapping: encoded Object input x 14,555 ops/sec ±0.48% (100 runs sampled)

source-map-js: encoded Object input x 2,447 ops/sec ±0.36% (99 runs sampled)

source-map-0.6.1: encoded Object input x 1,201 ops/sec ±0.57% (96 runs sampled)

Fastest is trace-mapping: decoded Object input

Trace speed:

trace-mapping: decoded originalPositionFor x 7,620,192 ops/sec ±0.09% (99 runs sampled)

trace-mapping: encoded originalPositionFor x 6,872,554 ops/sec ±0.30% (97 runs sampled)

source-map-js: encoded originalPositionFor x 2,489,570 ops/sec ±0.35% (94 runs sampled)

source-map-0.6.1: encoded originalPositionFor x 1,698,633 ops/sec ±0.28% (98 runs sampled)

source-map-0.8.0: encoded originalPositionFor x 4,015,644 ops/sec ±0.22% (98 runs sampled)

Fastest is trace-mapping: decoded originalPositionFor

\*\*\*

react.js.map - 5726 segments

Memory Usage:

trace-mapping decoded 16176 bytes

trace-mapping encoded 681552 bytes

source-map-js 2418352 bytes

source-map-0.6.1 2443672 bytes

source-map-0.8.0 111768 bytes

Smallest memory usage is trace-mapping decoded

Init speed:

trace-mapping: decoded JSON input x 1,720 ops/sec ±0.34% (98 runs sampled)

trace-mapping: encoded JSON input x 4,406 ops/sec ±0.35% (100 runs sampled)

trace-mapping: decoded Object input x 92,122 ops/sec ±0.10% (99 runs sampled)

trace-mapping: encoded Object input x 5,385 ops/sec ±0.37% (99 runs sampled)

source-map-js: encoded Object input x 794 ops/sec ±0.40% (98 runs sampled)

source-map-0.6.1: encoded Object input x 416 ops/sec ±0.54% (91 runs sampled)

Fastest is trace-mapping: decoded Object input

Trace speed:

trace-mapping: decoded originalPositionFor x 32,759,519 ops/sec ±0.33% (100 runs sampled)

trace-mapping: encoded originalPositionFor x 31,116,306 ops/sec ±0.33% (97 runs sampled)

source-map-js: encoded originalPositionFor x 17,458,435 ops/sec ±0.44% (97 runs sampled)

source-map-0.6.1: encoded originalPositionFor x 12,687,097 ops/sec ±0.43% (95 runs sampled)

source-map-0.8.0: encoded originalPositionFor x 23,538,275 ops/sec ±0.38% (95 runs sampled)

Fastest is trace-mapping: decoded originalPositionFor