Source Map Support

This module provides source map support for stack traces in node via the V8 stack trace API. It uses the source-map module to replace the paths and line numbers of source-mapped files with their original paths and line numbers. The output mimics node's stack trace format with the goal of making every compile-to-JS language more of a first-class citizen. Source maps are completely general (not specific to any one language) so you can use source maps with multiple compile-to-JS languages in the same node process.

Installation and Usage

Node support

```
$ npm install source-map-support
```

Source maps can be generated using libraries such as source-map-index-generator. Once you have a valid source map, place a source mapping comment somewhere in the file (usually done automatically or with an option by your transpiler):

```
//# sourceMappingURL=path/to/source.map
```

If multiple sourceMappingURL comments exist in one file, the last sourceMappingURL comment will be respected (e.g. if a file mentions the comment in code, or went through multiple transpilers). The path should either be absolute or relative to the compiled file.

From here you have two options.

CLI Usage

```
node -r source-map-support/register
        compiled.js
```

Programmatic Usage

Put the following line at the top of the compiled file.

```
require('source-map-support').install();
```

It is also possible to install the source map support directly by requiring the `register` module which can be handy with ES6:

```
import 'source-map-support/register'

// Instead of:
import sourceMapSupport from 'source-
        map-support'
sourceMapSupport.install()
```

Note: if you're using babel-register, it includes source-map-support already.

It is also very useful with Mocha:

Tests

This repo contains both automated tests for node and manual tests for the browser. The automated tests can be run using mocha (type `mocha` in the root directory). To run the manual tests:

- Build the tests using `build.js`
- Launch the HTTP server (`npm run serve-tests`) and visit
  - http://127.0.0.1:1336/amd-test
  - http://127.0.0.1:1336/browser-test
  - http://127.0.0.1:1336/browserify-test - **Currently not working** due to a bug with browserify (see pull request #66 for details).
- For `header-test`, run `server.js` inside that directory and visit http://127.0.0.1:1337/

License

This code is available under the MIT license.

Compile and run the file using the CoffeeScript compiler from the terminal:

```
$ npm install source-map-support
        coffeescript
$ node_modules/.bin/coffee --map --
        compile demo.coffee
$ node demo.js

demo.coffee:3
  bar = -> throw new Error 'this is a
        demo'
                    ^
Error: this is a demo
    at bar (demo.coffee:3:22)
    at foo (demo.coffee:4:3)
    at Object.<anonymous>
        (demo.coffee:5:1)
    at Object.<anonymous>
        (demo.coffee:1:1)
    at Module._compile
        (module.js:456:26)
    at Object.Module._extensions..js
        (module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load
        (module.js:312:12)
    at Function.Module.runMain
        (module.js:497:10)
    at startup (node.js:119:16)
```



```
$ mocha --require source-map-support/
register tests/
```

Browser support

This library also works in Chrome. While the DevTools console already supports source maps, the V8 engine doesn't and `Error.prototype.stack` will be incorrect without this library. Everything will just work if you deploy your source files using browserify. Just make sure to pass the `--debug` flag to the browserify command so your source maps are included in the bundled code.

This library also works if you use another build process or just include the source files directly. In this case, include the file `browser-source-map-support.js` in your page and call `sourceMapSupport.install()`. It contains the whole library already bundled for the browser using browserify.

```
<script src="browser-source-map-
        support.js"></script>
<script>sourceMapSupport.install();</
        script>
```

This library also works if you use AMD (Asynchronous Module Definition), which is used in tools like RequireJS. Just list `browser-source-map-support` as a dependency:

```
<script>
  define(['browser-source-map-
        support'],
        function(sourceMapSupport) {
    sourceMapSupport.install();
```

```
  });
</script>
```

Options

This module installs two things: a change to the `stack` property on `Error` objects and a handler for uncaught exceptions that mimics node's default exception handler (the handler can be seen in the demos below). You may want to disable the handler if you have your own uncaught exception handler. This can be done by passing an argument to the installer:

```
require('source-map-support').install({
  handleUncaughtExceptions: false
});
```

This module loads source maps from the filesystem by default. You can provide alternate loading behavior through a callback as shown below. For example, Meteor keeps all source maps cached in memory to avoid disk access.

```
require('source-map-support').install({
  retrieveSourceMap: function(source) {
    if (source === 'compiled.js') {
      return {
        url: 'original.js',
        map:
         fs.readFileSync('compiled.js.map',
         'utf8')
      };
```



```
$ node_modules/typescript/bin/tsc
-sourcemap demo.ts
$ node -r source-map-support/register
demo.js

demo.ts:5
  bar() { throw new Error('this is a
demo'); }
                ^
Error: this is a demo
    at Foo.bar (demo.ts:5:17)
    at new Foo (demo.ts:4:24)
    at Object.<anonymous> (demo.ts:7:1)
    at Module._compile
(module.js:456:26)
    at Object.Module._extensions..js
(module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load
(module.js:312:12)
    at Function.Module.runMain
(module.js:497:10)
    at startup (node.js:119:16)
    at node.js:901:3
```

CoffeeScript Demo

demo.coffee:

```
require('source-map-support').install()
foo = ->
  bar = -> throw new Error 'this is a
        demo'
  bar()
foo()
```

Compile and run the file using the TypeScript compiler from the terminal:

```
$ npm install source-map-support
typescript
$ node_modules/typescript/bin/tsc
-sourcemap demo.ts
$ node demo.js

demo.ts:5
  bar() { throw new Error('this is a
demo'); }
                ^
Error: this is a demo
    at Foo.bar (demo.ts:5:17)
    at new Foo (demo.ts:4:24)
    at Object.<anonymous> (demo.ts:7:1)
    at Module._compile
(module.js:456:26)
    at Object.Module._extensions..js
(module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load
(module.js:312:12)
    at Function.Module.runMain
(module.js:497:10)
    at startup (node.js:119:16)
    at node.js:901:3
```

There is also the option to use `-r source-map-support/register` with typescript, without the need add the `require('source-map-support').install()` in the code base:

```
$ npm install source-map-support
typescript
```



```
    }
    return null;
  }
});
```

The module will by default assume a browser environment if XMLHttpRequest and window are defined. If either of these do not exist it will instead assume a node environment. In some rare cases, e.g. when running a browser emulation and where both variables are also set, you can explictly specify the environment to be either ‘browser’ or ‘node’.

```
require('source-map-support').install({
  environment: 'node'
});
```

To support files with inline source maps, the `hookRequire` options can be specified, which will monitor all source files for inline source maps.

```
require('source-map-support').install({
  hookRequire: true
});
```

This monkey patches the `require` module loading chain, so is not enabled by default and is not recommended for any sort of production usage.

Demos

Basic Demo

original.js:

```
throw new Error('test'); // This is the
        original code
```

compiled.js:

```
require('source-map-support').install();

throw new Error('test'); // This is the
        compiled code
// The next line defines the
        sourceMapping.
//# sourceMappingURL=compiled.js.map
```

compiled.js.map:

```
{
  "version": 3,
  "file": "compiled.js",
  "sources": ["original.js"],
  "names": [],
  "mappings": ";;AAAA,MAAM,IAAI"
}
```

Run compiled.js using node (notice how the stack trace uses original.js instead of compiled.js):


```
$ node compiled.js

original.js:1
throw new Error('test'); // This is the
original code
      ^
Error: test
    at Object.<anonymous>
(original.js:1:7)
    at Module._compile
(module.js:456:26)
    at Object.Module._extensions..js
(module.js:474:10)
    at Module.load (module.js:356:32)
    at Function.Module._load
(module.js:312:12)
    at Function.Module.runMain
(module.js:497:10)
    at startup (node.js:119:16)
    at node.js:901:3
```

TypeScript Demo

demo.ts:

```
declare function require(name: string);
require('source-map-support').install();
class Foo {
  constructor() { this.bar(); }
  bar() { throw new Error('this is a
        demo'); }
}
new Foo();
```