Linting Typescript Projects

This page will discuss common topics around maintaining the code quality of Typescript codebases using the Mill build tool

Linting and AutoFormatting with Eslint and Prettier

Eslint and Prettier are tools that analyzes your Typescript source code, performing intelligent analyses and code quality checks, and is often able to automatically fix the issues that it discovers. It can also perform automated refactoring.

build.mill (download, browse)

```
package build

import mill._, javascriptlib._

object foo extends TypeScriptModule
```

Mill supports code linting via `eslint` https://eslint.org, `typescript-eslint` https://typescript-eslint.io/getting-started and `prettier` https://prettier.io/docs/en out of the box. You can lint your projects code by providing a configuration for your preferred linter and running `mill _.checkFormatAll`.

If both configuration files are present, the command `mill _.checkFormatAll` will default to eslint. If neither files are present, the command will exit with an error, you must include at least one configuration file. You can lint via a specificied linter via the commands `mill _.checkFormatEslint` to lint with eslint and `mill _.checkFormatPrettier` to lint with prettier.

When using prettier you can specify the path to lint via command line argument, `mill _.checkFormatAll "//.ts"` just as you would when running `prettier --check` if no path is provided mill will default to using "//.ts". Also if a `.prettierignore` is not provided, mill will generate one ignoring "node_modules" and ".git".

You can define `npmLintDeps` field to add dependencies specific to linting to your module. The task `npmLintDeps` works the same way as `npmDeps` or `npmDevDeps`.

```
> cat foo/src/foo.ts # initial poorly formatted source code
export class Foo{
static main(
args: string[
])
```

```
{console.log("Hello World!")
}
}

> mill foo.checkFormatAll # run linter - since both eslint and prettier
configurations are present, mill will opt to use eslint.
...
foo/src/foo.ts
  2:1   error  Expected indentation of 2 spaces but found 0
indent
  3:1   error  Expected indentation of 4 spaces but found 0
indent
  5:1   error  Opening curly brace does not appear on the same line as
controlling statement  brace-style
  5:1   error  Statement inside of curly braces should be on next line
brace-style
  5:1   error  Requires a space after '{'
block-spacing
  5:1   error  Expected indentation of 2 spaces but found 0
indent
  5:14  error  Strings must use singlequote
quotes
  5:29  error  Missing semicolon
semi
  6:1   error  Expected indentation of 2 spaces but found 0
indent
  7:2   error  Newline required at end of file but not found
eol-last
...
...10 problems (10 errors, 0 warnings)
...10 errors and 0 warnings potentially fixable with running foo.reformatAll.

> rm -rf eslint.config.mjs # since there is no eslint config file
`eslint.config.(js|mjs|cjs)`, mill will use prettier if a `.prettierrc` conf
file is available.

> mill foo.checkFormatAll # run lint with prettier configuration.
Checking formatting...
[warn] foo/src/foo.ts
[warn] Code style issues found. Run foo.reformatAll to fix.
```

build.mill (download, browse)

```
package build

import mill._, javascriptlib._

object foo extends TypeScriptModule
```

Mill supports code formatting via `eslint` https://eslint.org, `typescript-eslint` https://typescript-eslint.io/getting-started and `prettier` https://prettier.io/docs/en out of the box. You can reformat your projects code by providing a configuration for your preferred linter and running `mill _.reformatAll`.

If both configurations files are present, the command `mill _.reformatAll` will default to eslint. You can format via a specificied linter via the commands `mill _.reformatEslint` to format with eslint and `mill _.reformatPrettier` to format with prettier.

When using prettier you can specify the path to reformat via command line argument, `mill _.reformatAll "//.ts"` just as you would when running `prettier --write` if no path is provided mill will default to using "//.ts". Also if a `.prettierignore` is not provided, mill will generate one ignoring "node_modules" and ".git".

BASH

```
> cat foo/src/foo.ts # initial poorly formatted source code
export class Foo{
static main(
args: string[
])
{console.log("Hello World!")
}
}

> mill foo.checkFormatAll # run linter - since both eslint and prettier
configurations are present, mill will opt to use eslint.
...
foo/src/foo.ts
  2:1   error  Expected indentation of 2 spaces but found 0
indent
  3:1   error  Expected indentation of 4 spaces but found 0
indent
  5:1   error  Opening curly brace does not appear on the same line as
controlling statement  brace-style
  5:1   error  Statement inside of curly braces should be on next line
brace-style
  5:1   error  Requires a space after '{'
block-spacing
  5:1   error  Expected indentation of 2 spaces but found 0
indent
  5:14  error  Strings must use singlequote
quotes
  5:29  error  Missing semicolon
semi
  6:1   error  Expected indentation of 2 spaces but found 0
indent
  7:2   error  Newline required at end of file but not found
eol-last
...
...10 problems (10 errors, 0 warnings)
```

```
...10 errors and 0 warnings potentially fixable with running foo.reformatAll.

> mill foo.reformatAll
...
All matched files have been reformatted!

> cat foo/src/foo.ts # code formatted with eslint configuration.
export class Foo{
  static main(
    args: string[
]) {
    console.log('Hello World!');
  }
}

> rm -rf eslint.config.mjs # since there is no eslint config file
`eslint.config.(js|mjs|cjs)`, mill will use the prettier configuration
available.

> mill foo.checkFormatAll # run lint with prettier configuration.
Checking formatting...
[warn] foo/src/foo.ts
[warn] Code style issues found. Run foo.reformatAll to fix.

> mill foo.reformatAll
...
All matched files have been reformatted!

> cat foo/src/foo.ts # code formatted with prettier configuration.
export class Foo {
  static main(args: string[]) {
    console.log('Hello World!');
  }
}
```

Code Coverage with Jest, Mocha, Vitest and Jasmine

Mill supports code coverage analysis with multiple Typescript testing frameworks.

build.mill (download, browse)

```
package build

import mill._, javascriptlib._

object foo extends TypeScriptModule {
  object test extends TypeScriptTests with TestModule.Jest
}

object bar extends TypeScriptModule {
```

```
  object test extends TypeScriptTests with TestModule.Mocha
}

object baz extends TypeScriptModule {
  object test extends TypeScriptTests with TestModule.Vitest
}

object qux extends TypeScriptModule {
  object test extends TypeScriptTests with TestModule.Jasmine
}
```

Mill supports code coverage with `Jest`, `Mocha`, `Vitest` and `Jasmine` out of the box. To run a test with coverage, run the command `mill _.test.coverage`.

The path to generated coverage data can be retrieved with `mill _.test.coverageFiles`, coverage data can also be displayed in a web brower via the task `mill _.test.htmlReport`.

```
> mill foo.test.coverage
...Calculator
...
---------------|---------|----------|---------|---------|-------------------
File...| % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s...
---------------|---------|----------|---------|---------|-------------------
...All files...|...100 |...100 |...100 |...100 |...
...calculator.ts...|...100 |...100 |...100 |...100 |...
---------------|---------|----------|---------|---------|-------------------
...
Test Suites:...1 passed, 1 total...
Tests:...4 passed, 4 total...
...

> mill bar.test.coverage
...
...4 passing...
...
---------------|---------|----------|---------|---------|-------------------
File...| % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s...
---------------|---------|----------|---------|---------|-------------------
...All files...|...100 |...100 |...100 |...100 |...
...calculator.ts...|...100 |...100 |...100 |...100 |...
---------------|---------|----------|---------|---------|-------------------

> mill baz.test.coverage
.../calculator.test.ts...
...Test Files  1 passed...
...Tests  4 passed...
...
...Coverage report from v8
---------------|---------|----------|---------|---------|-------------------
File...| % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s...
```

```
---------------|---------|----------|---------|---------|-------------------
...All files...|...100 |...100 |...100 |...100 |...
...calculator.ts...|...100 |...100 |...100 |...100 |...
---------------|---------|----------|---------|---------|-------------------

> mill qux.test.coverage
...
4 specs, 0 failures
...
---------------|---------|----------|---------|---------|-------------------
File...| % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s...
---------------|---------|----------|---------|---------|-------------------
...All files...|...100 |...100 |...100 |...100 |...
...calculator.ts...|...100 |...100 |...100 |...100 |...
---------------|---------|----------|---------|---------|-------------------
```