TypeScript vs Babel

JShelf

Transpiler or not?

- CoffeeScript / ClojureScript
  - Syntactic sugar with some new features. Still writing front-end code. Just not JavaScript.
- Dart / GWT / GopherJS / …
  - Does not feel writing front-end code anymore.
- TypeScript / Babel
  - Supports the future (ES6).
  - Still JavaScript.

- **Objective: Write the future JavaScript. Choose TypeScript or Babel.**

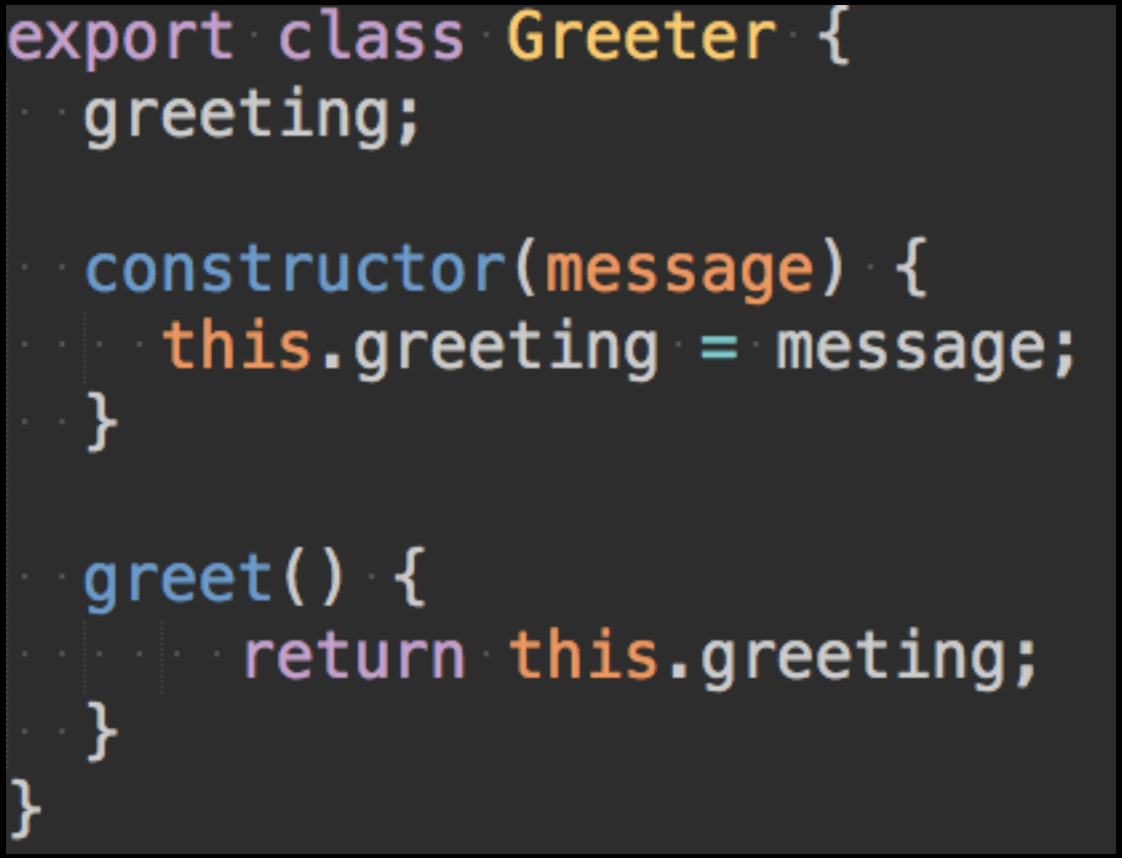
- https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS

ES6 support

- Babel > TypeScript
  - https://kangax.github.io/compat-table/es6/

Transpiled code readability - ES6 Original

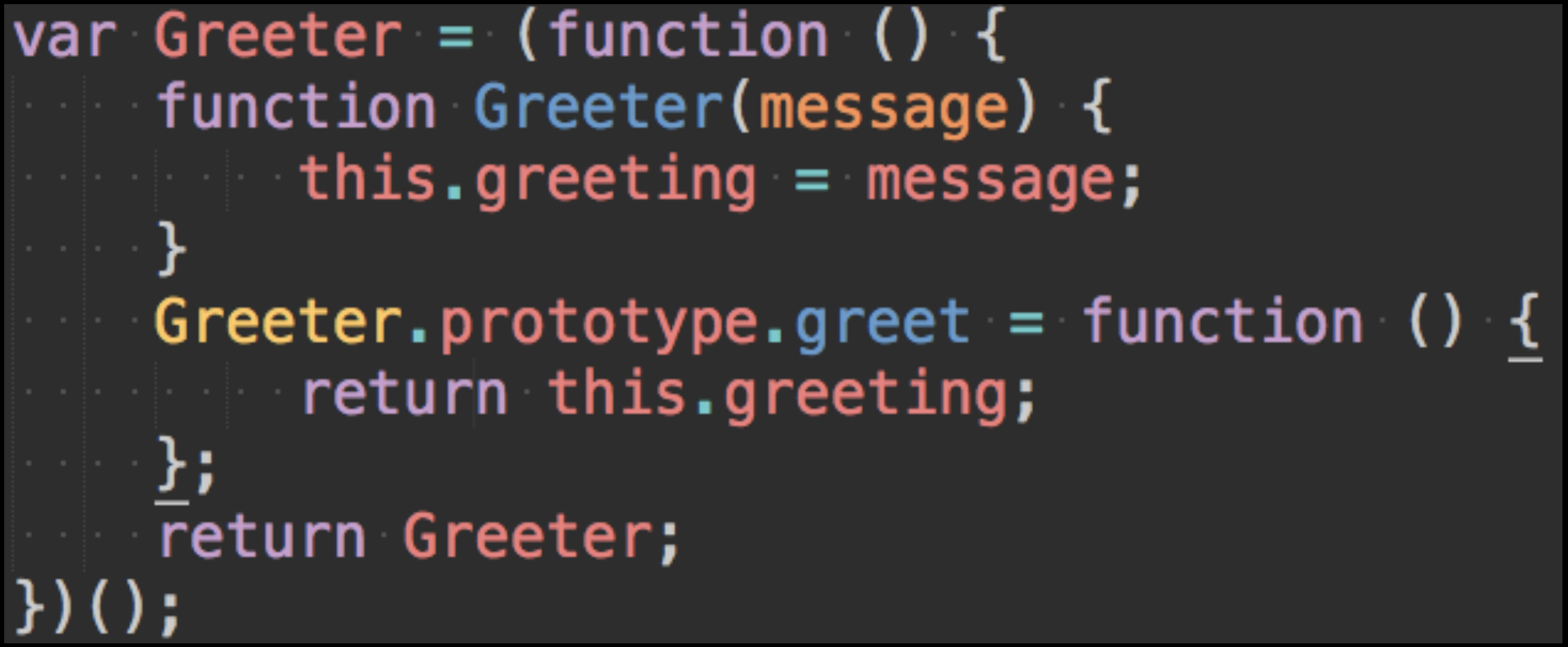
- Class Definition in ES6



```
export class Greeter {
  greeting;

  constructor(message) {
    this.greeting = message;
  }

  greet() {
      return this.greeting;
  }
}
```

Transpiled code readability - TypeScript

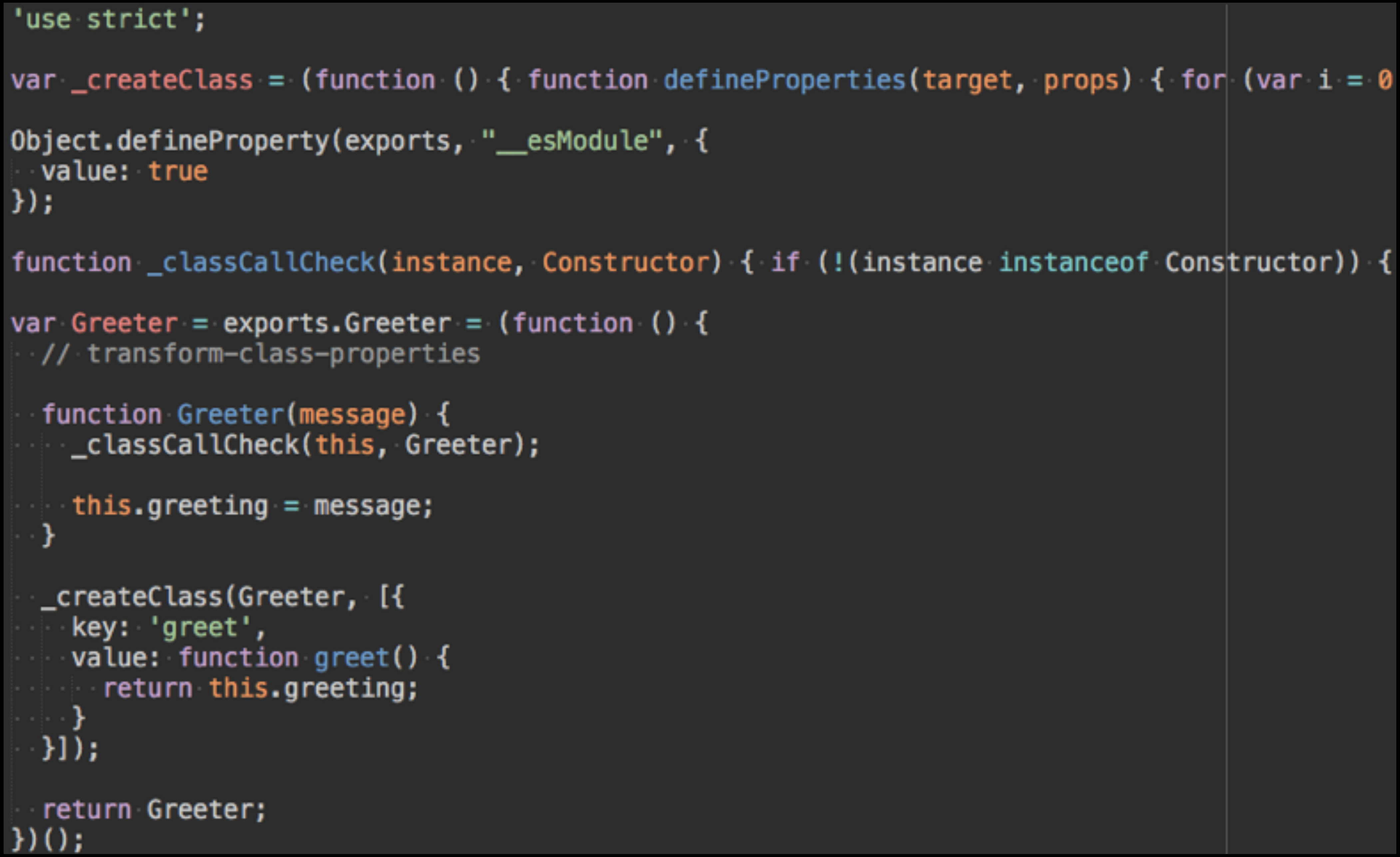
- TypeScript compiled code



```
var Greeter = (function () {
    function Greeter(message) {
        this.greeting = message;
    }
    Greeter.prototype.greet = function () {
        return this.greeting;
    };
    return Greeter;
})();
```

Transpiled code readability - Babel

- Babel compiled code



```
'use strict';

var _createClass = (function () { function defineProperties(target, props) { for (var i = 0

Object.defineProperty(exports, "__esModule", {
  value: true
});

function _classCallCheck(instance, Constructor) { if (!(instance instanceof Constructor)) {

var Greeter = exports.Greeter = (function () {
  // transform-class-properties

  function Greeter(message) {
    _classCallCheck(this, Greeter);

    this.greeting = message;
  }

  _createClass(Greeter, [{
    key: 'greet',
    value: function greet() {
      return this.greeting;
    }
  }]);

  return Greeter;
})();
```

Module - Babel

- Babel
  - Exact ES6 syntax (import / export)
  - One file per module

Module - TypeScript

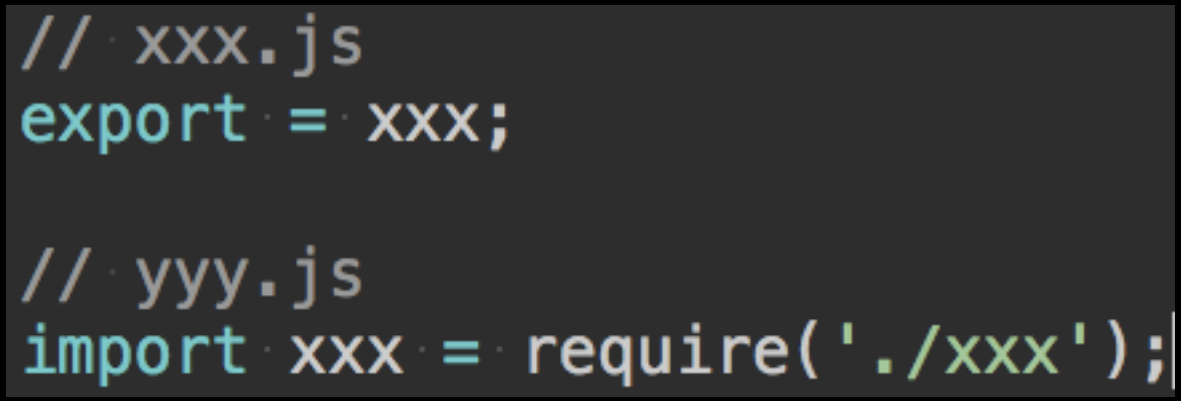
- TypeScript
  - Internal (namespace)
  - External

- DON'T MIX these usages.
- NO CHOICE BUT USE EXTERNAL (Unless you don't need any other 3rd lib)

- http://www.typescriptlang.org/Handbook#modules
- https://www.stevefenton.co.uk/2015/05/Stop-Mixing-TypeScript-Internal-And-External-Modules/

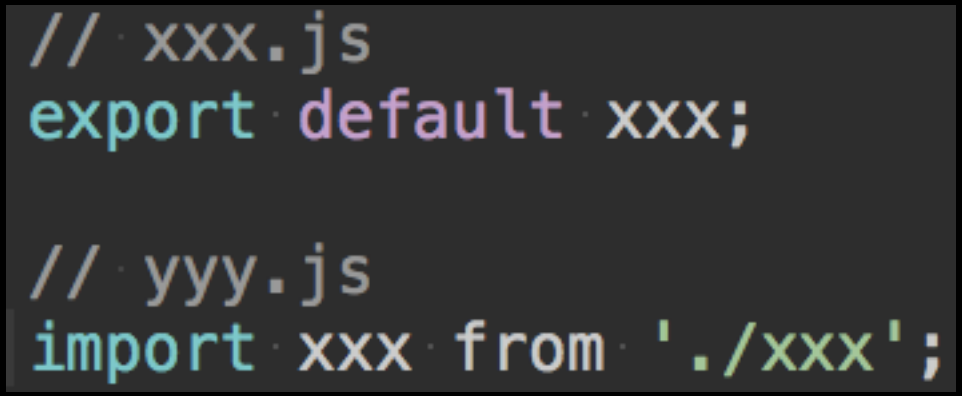
Module - TypeScript

- Simple, backwards compatible with AMD, CommonJS. Not suggested as legacy soon



```
// xxx.js
export = xxx;

// yyy.js
import xxx = require('./xxx');
```

- ES6 style alternatively



```
// xxx.js
export default xxx;

// yyy.js
import xxx from './xxx';
```

Type and IDE support - Babel

- Babel
  - Flow: http://flowtype.org/
  - IDE Support: Sublime, Nuclide (*http://nuclide.io/ Atom package)*

- Disadvantages:
  - Not Support Windows
  - Atom is slow
  - Sublime plugin not smart enough

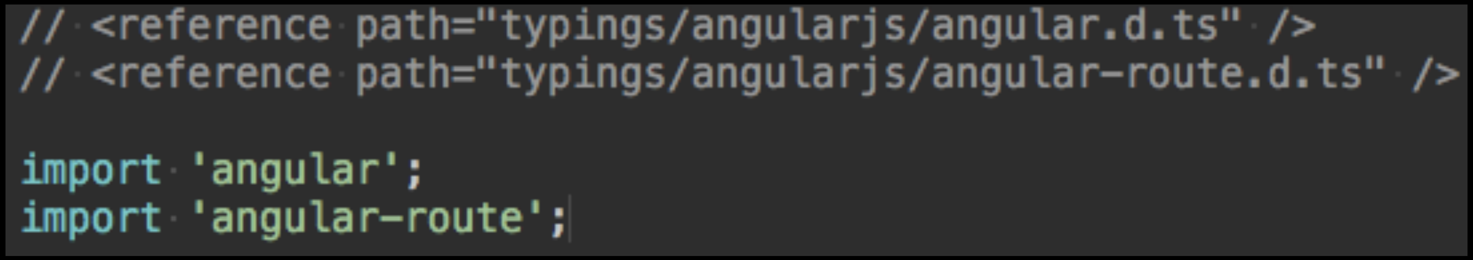
Type and IDE support - TypeScript

- TypeScript (typed supperset of JavaScript)
  - DefinitelyTyped project provides extensive 3rd party lib
  - IDE Support: Sublime, Atom, Webstorm, Visual Studio

- DefinitelyTyped: http://definitelytyped.org/

TypeScript Usage

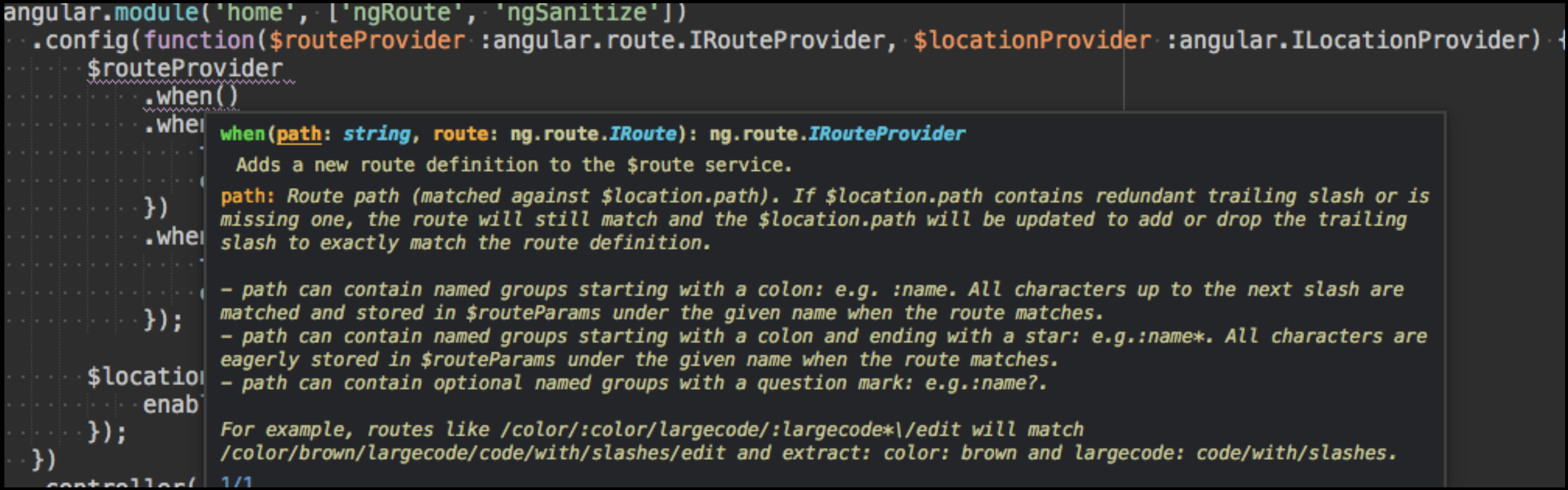
- Install TypeScript & Definition manager
  - npm install -g typescript tsd
  - tsd install angular angular-route
- Reference in TypeScript



```
// <reference path="typings/angularjs/angular.d.ts" />
// <reference path="typings/angularjs/angular-route.d.ts" />

import 'angular';
import 'angular-route';
```

TypeScript Usage

- Code Tips and auto-complete



```
angular.module('home', ['ngRoute', 'ngSanitize'])
  .config(function($routeProvider :angular.route.IRouteProvider, $locationProvider :angular.ILocationProvider) {
    $routeProvider
      .when()
      .whe
      })
      .whe
      });
    $locatio
      enab
    });
  })
  .controller(
```

when(path: string, route: ng.route.IRoute): ng.route.IRouteProvider

Adds a new route definition to the $route service.

path: *Route path (matched against $location.path). If $location.path contains redundant trailing slash or is missing one, the route will still match and the $location.path will be updated to add or drop the trailing slash to exactly match the route definition.*

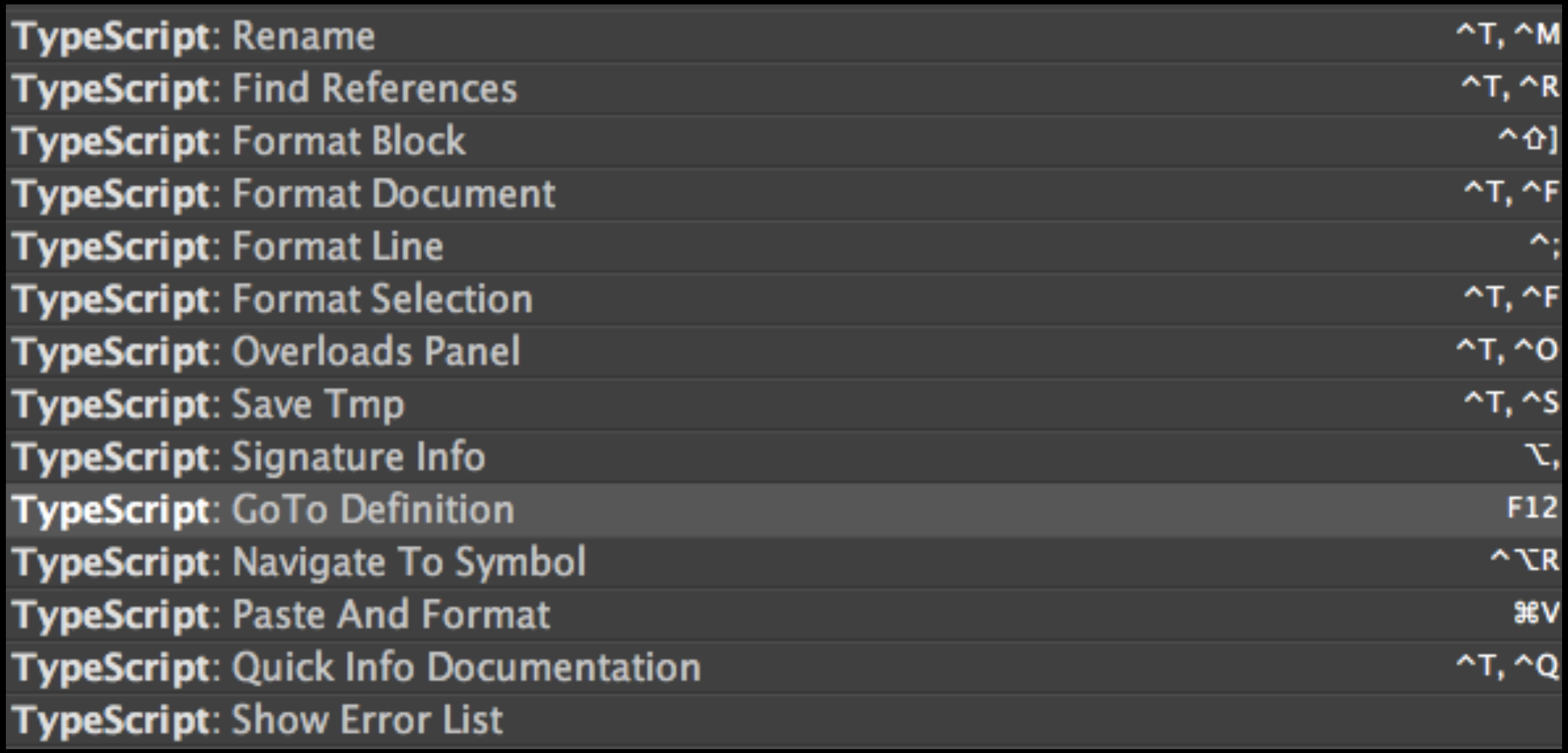
- path can contain named groups starting with a colon: e.g. :name. All characters up to the next slash are matched and stored in $routeParams under the given name when the route matches.
- path can contain named groups starting with a colon and ending with a star: e.g.:name. All characters are eagerly stored in $routeParams under the given name when the route matches.*
- path can contain optional named groups with a question mark: e.g.:name?.

For example, routes like /color/:color/largecode/:largecode\/edit will match /color/brown/largecode/code/with/slashes/edit and extract: color: brown and largecode: code/with/slashes.*

1/1

TypeScript Usage

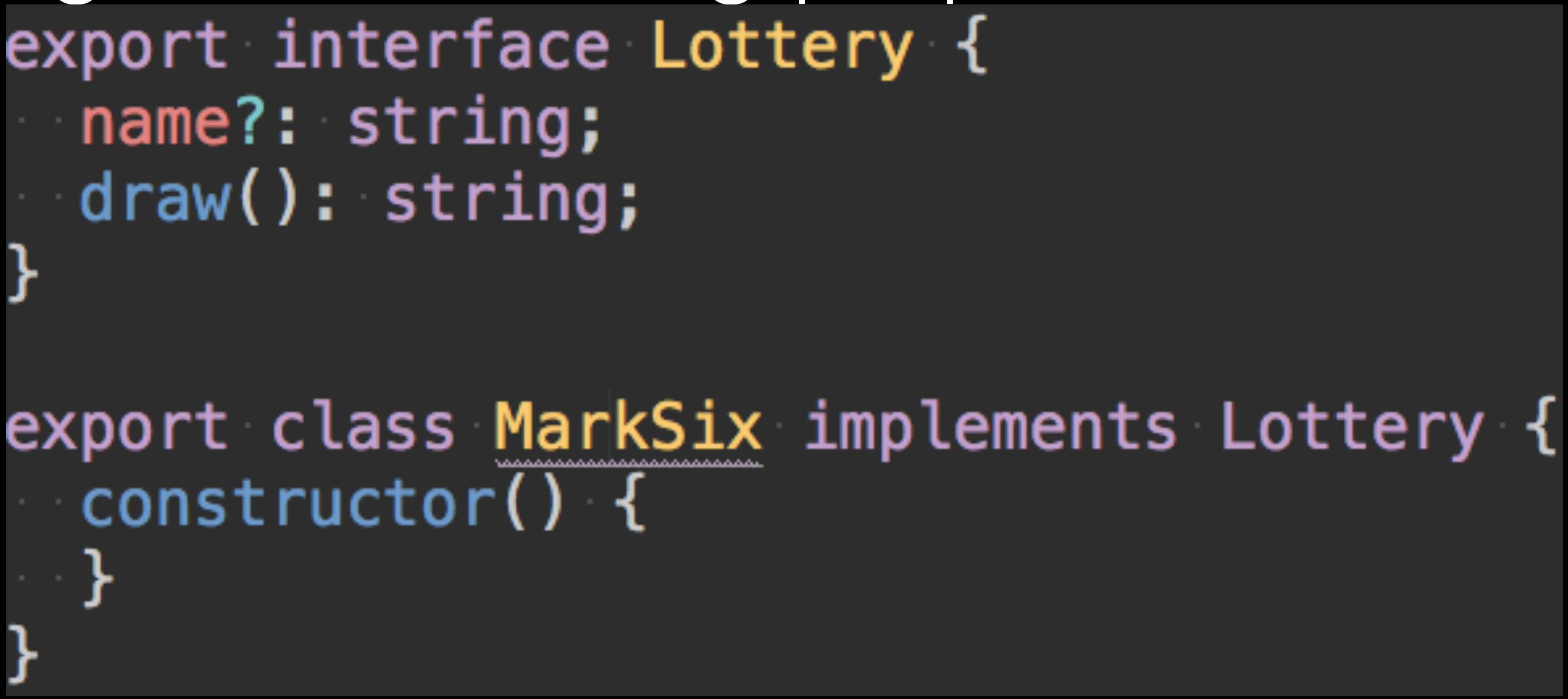
- Jump to definition and many support shortcuts



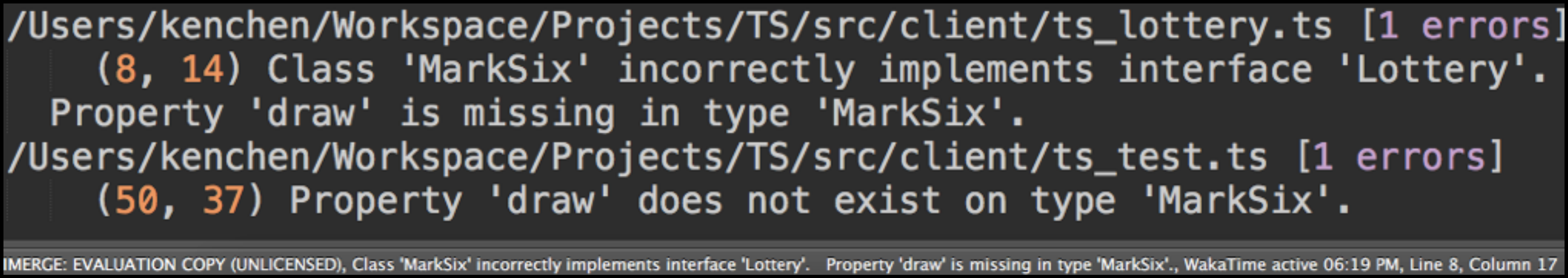
TypeScript: Rename	^T, ^M
TypeScript: Find References	^T, ^R
TypeScript: Format Block	^⇧]
TypeScript: Format Document	^T, ^F
TypeScript: Format Line	^;
TypeScript: Format Selection	^T, ^F
TypeScript: Overloads Panel	^T, ^O
TypeScript: Save Tmp	^T, ^S
TypeScript: Signature Info	⌥,
TypeScript: GoTo Definition	F12
TypeScript: Navigate To Symbol	^⌥R
TypeScript: Paste And Format	⌘V
TypeScript: Quick Info Documentation	^T, ^Q
TypeScript: Show Error List	

TypeScript Usage

- Warning about missing properties



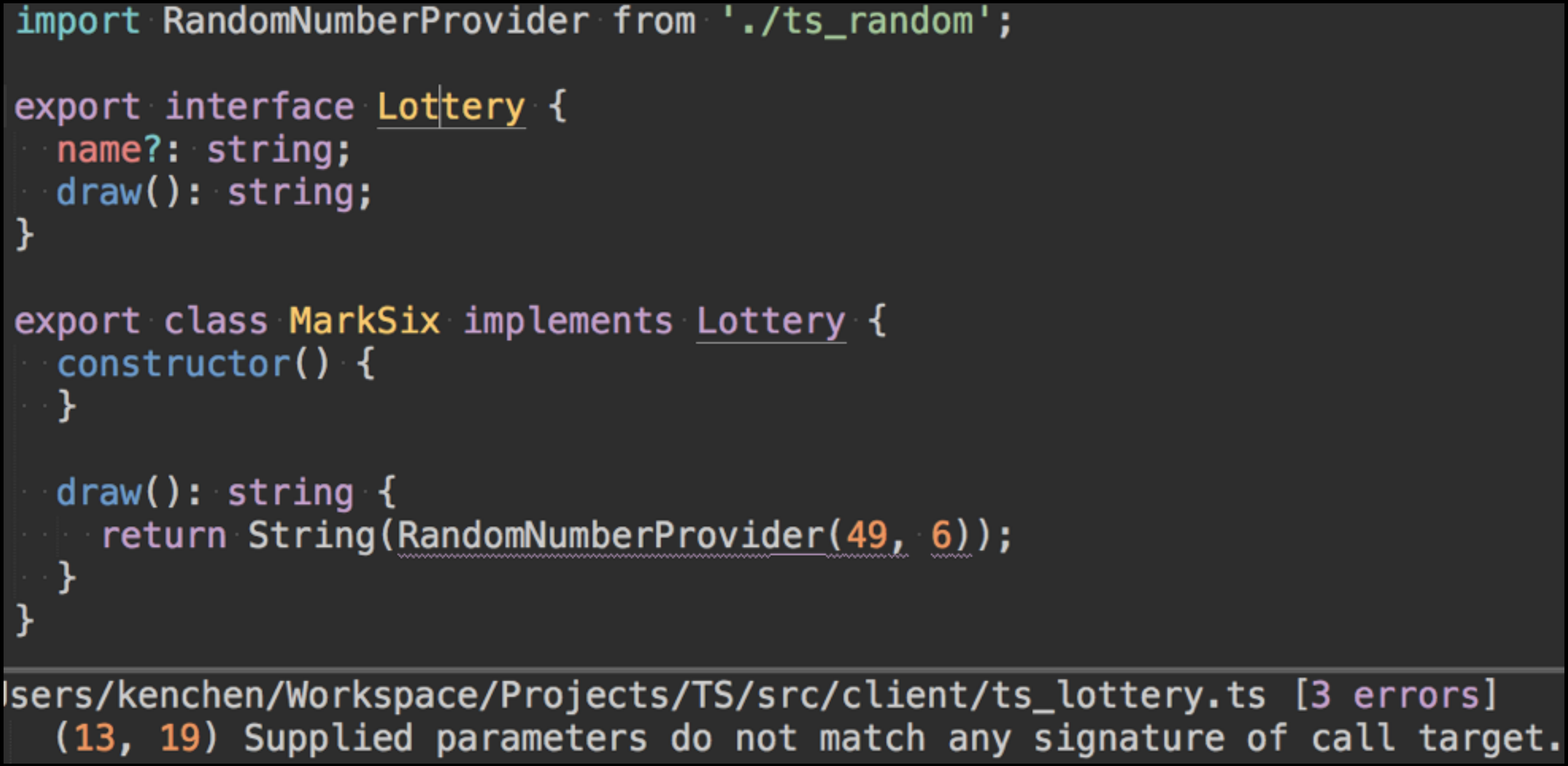
```typescript
export interface Lottery {
  name?: string;
  draw(): string;
}

export class MarkSix implements Lottery {
  constructor() {
  }
}
```



```
/Users/kenchen/Workspace/Projects/TS/src/client/ts_lottery.ts [1 errors]
    (8, 14) Class 'MarkSix' incorrectly implements interface 'Lottery'.
  Property 'draw' is missing in type 'MarkSix'.
/Users/kenchen/Workspace/Projects/TS/src/client/ts_test.ts [1 errors]
    (50, 37) Property 'draw' does not exist on type 'MarkSix'.

IMERGE: EVALUATION COPY (UNLICENSED), Class 'MarkSix' incorrectly implements interface 'Lottery'.   Property 'draw' is missing in type 'MarkSix'., WakaTime active 06:19 PM, Line 8, Column 17
```

TypeScript Usage

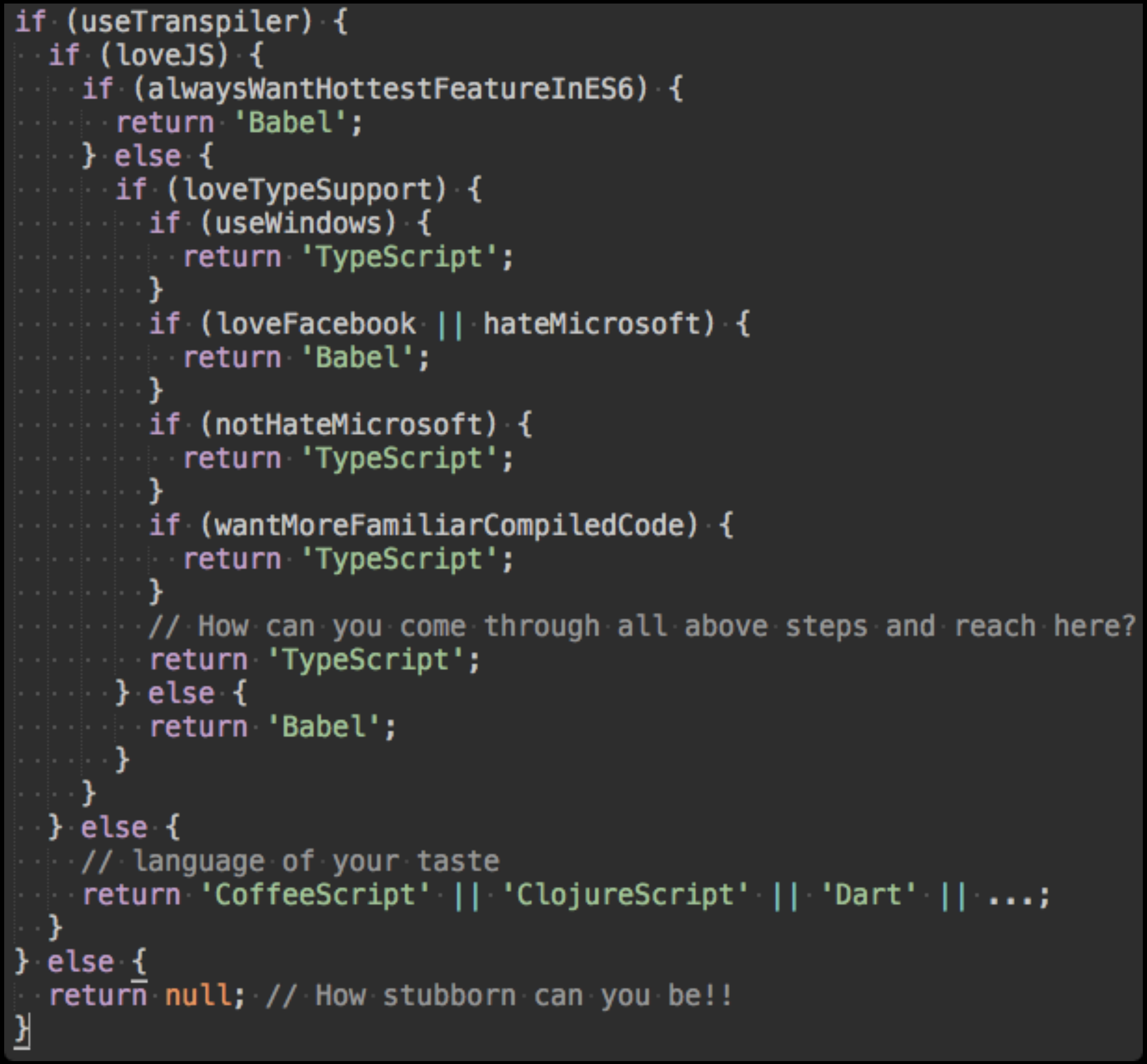
- Warning about incorrect function signature



```
import RandomNumberProvider from './ts_random';

export interface Lottery {
  name?: string;
  draw(): string;
}

export class MarkSix implements Lottery {
  constructor() {
  }

  draw(): string {
    return String(RandomNumberProvider(49, 6));
  }
}
```

```
sers/kenchen/Workspace/Projects/TS/src/client/ts_lottery.ts [3 errors]
  (13, 19) Supplied parameters do not match any signature of call target.
```

Which one to choose?



```
if (useTranspiler) {
  if (loveJS) {
    if (alwaysWantHottestFeatureInES6) {
      return 'Babel';
    } else {
      if (loveTypeSupport) {
        if (useWindows) {
          return 'TypeScript';
        }
        if (loveFacebook || hateMicrosoft) {
          return 'Babel';
        }
        if (notHateMicrosoft) {
          return 'TypeScript';
        }
        if (wantMoreFamiliarCompiledCode) {
          return 'TypeScript';
        }
        // How can you come through all above steps and reach here?
        return 'TypeScript';
      } else {
        return 'Babel';
      }
    }
  } else {
    // language of your taste
    return 'CoffeeScript' || 'ClojureScript' || 'Dart' || ...;
  }
} else {
  return null; // How stubborn can you be!!
}
```

Others to consider

- How it work with existing codebase?
- How it work with Node.js
- How it work with Webpack

How it work with existing JS?

- Can directly rename .ts to .js without error in most of the case.
- Type declaration is optional. You can add it if you want the benefit of type checking.

How it work with Node.js

1. Simply compile all files before use and restart
2. Interop with TypeScript `require` extension https://github.com/theblacksmith/typescript-require

   During process bootstrap, require `typescript-require` once like:

   require('typescript-require');

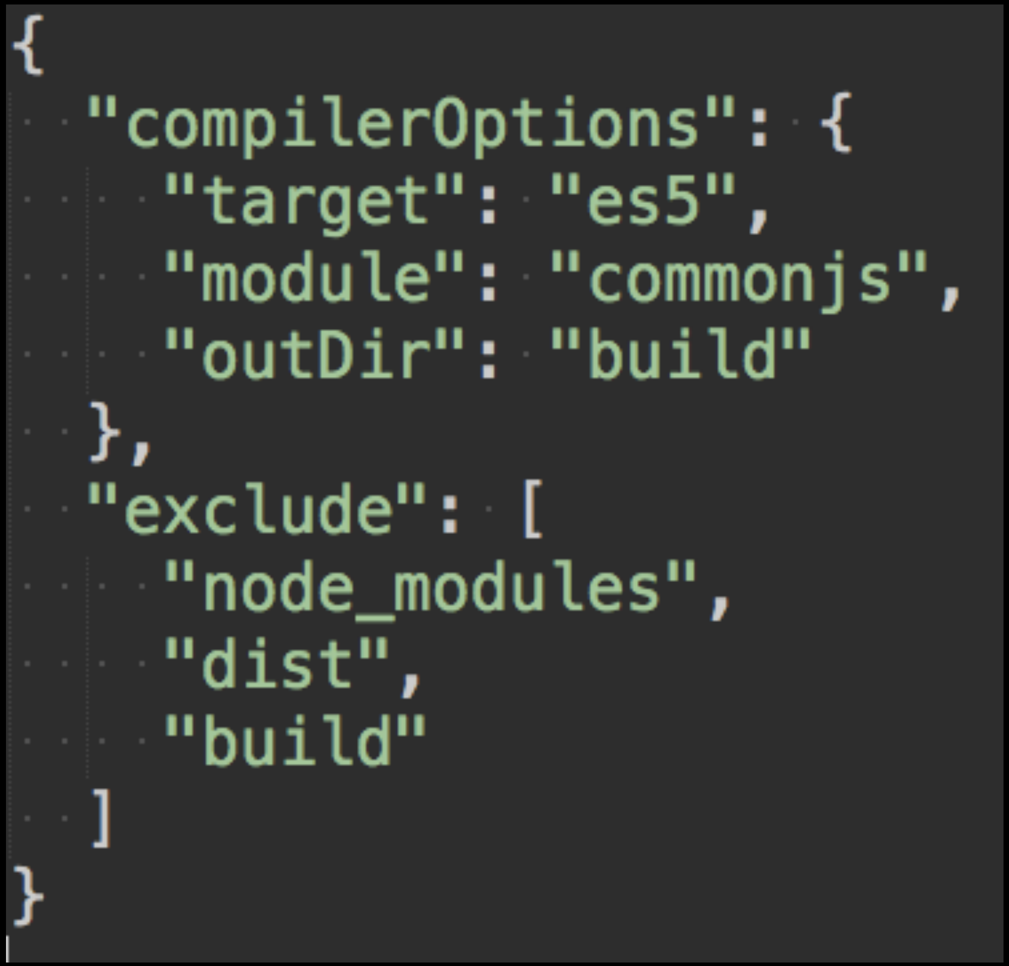
   After that, `.ts` module can be required just like `.js` module.

My choice: Option #1

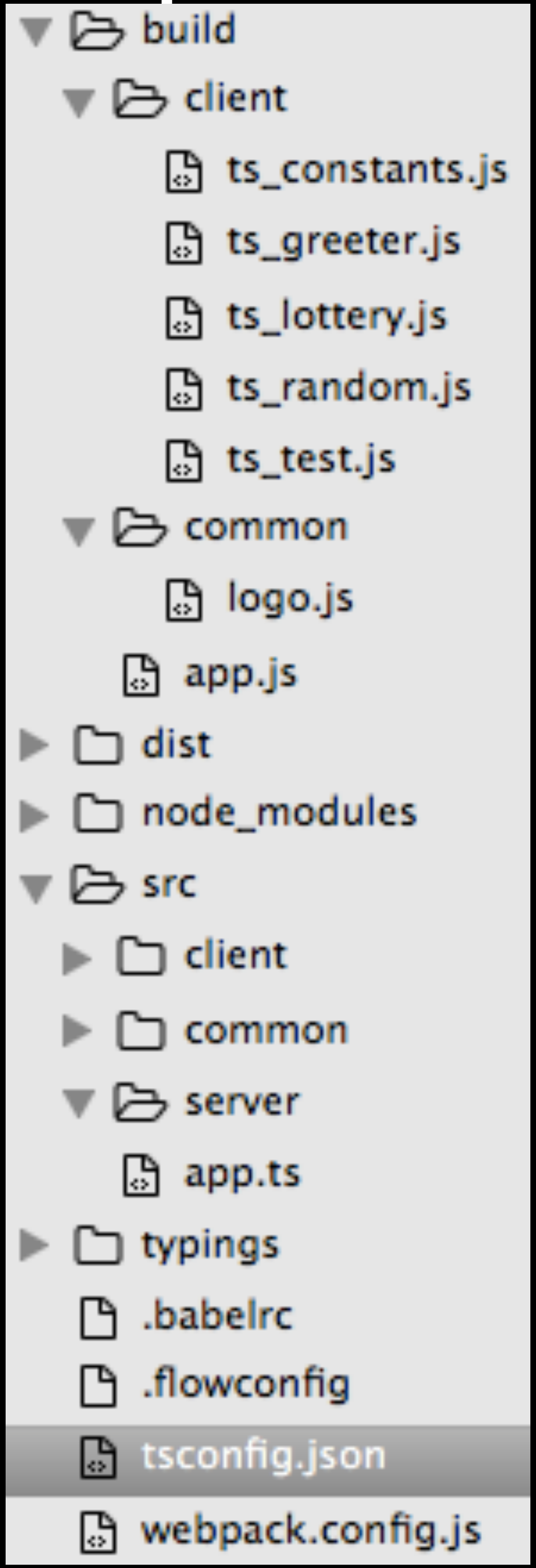
- Extension requires special syntax although very minimum.
- Extension is not official and might have unexpected result

How it works with Webpack

- TypeScript config: tsconfig.json

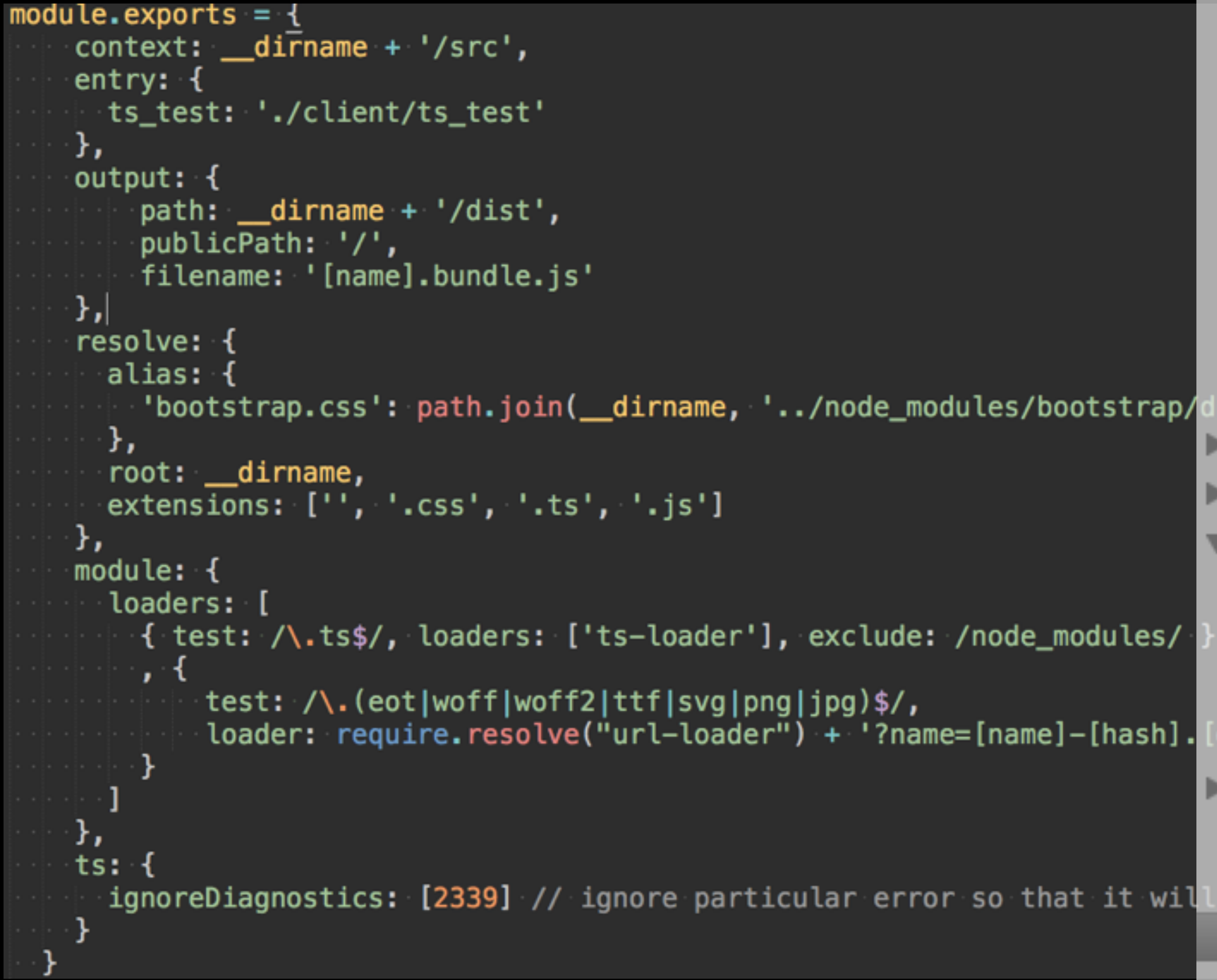


```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "outDir": "build"
  },
  "exclude": [
    "node_modules",
    "dist",
    "build"
  ]
}
```

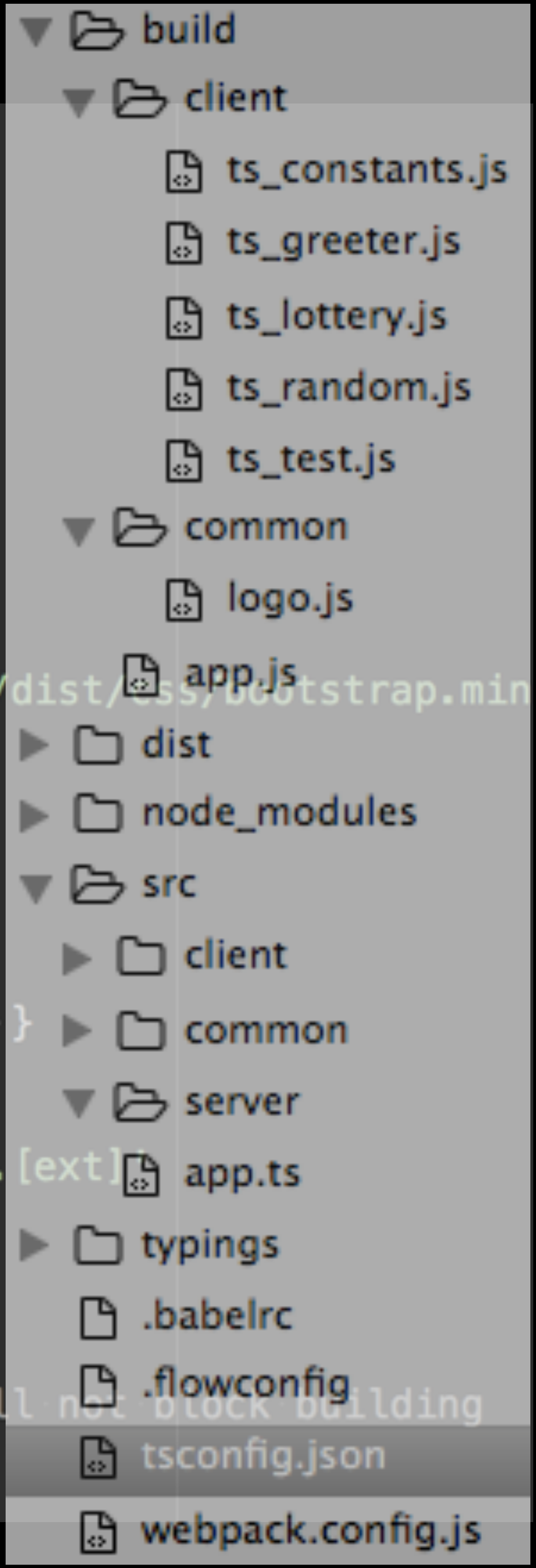


```
▼ build
  ▼ client
      ts_constants.js
      ts_greeter.js
      ts_lottery.js
      ts_random.js
      ts_test.js
  ▼ common
      logo.js
    app.js
▶ dist
▶ node_modules
▼ src
  ▶ client
  ▶ common
  ▼ server
      app.ts
▶ typings
  .babelrc
  .flowconfig
  tsconfig.json
  webpack.config.js
```

How it works with Webpack - ts-loader



```
module.exports = {
    context: __dirname + '/src',
    entry: {
      ts_test: './client/ts_test'
    },
    output: {
        path: __dirname + '/dist',
        publicPath: '/',
        filename: '[name].bundle.js'
    },
    resolve: {
      alias: {
        'bootstrap.css': path.join(__dirname, '../node_modules/bootstrap/dist/css/bootstrap.min
      },
      root: __dirname,
      extensions: ['', '.css', '.ts', '.js']
    },
    module: {
      loaders: [
        { test: /\.ts$/, loaders: ['ts-loader'], exclude: /node_modules/ }
        , {
          test: /\.(eot|woff|woff2|ttf|svg|png|jpg)$/,
          loader: require.resolve("url-loader") + '?name=[name]-[hash].[ext]
        }
      ]
    },
    ts: {
      ignoreDiagnostics: [2339] // ignore particular error so that it will not block building
    }
}
```



```
▼ build
  ▼ client
      ts_constants.js
      ts_greeter.js
      ts_lottery.js
      ts_random.js
      ts_test.js
  ▼ common
      logo.js
    app.js
▶ dist
▶ node_modules
▼ src
  ▶ client
  ▶ common
  ▼ server
      app.ts
▶ typings
  .babelrc
  .flowconfig
  tsconfig.json
  webpack.config.js
```

How it works with Webpack - ts-loader

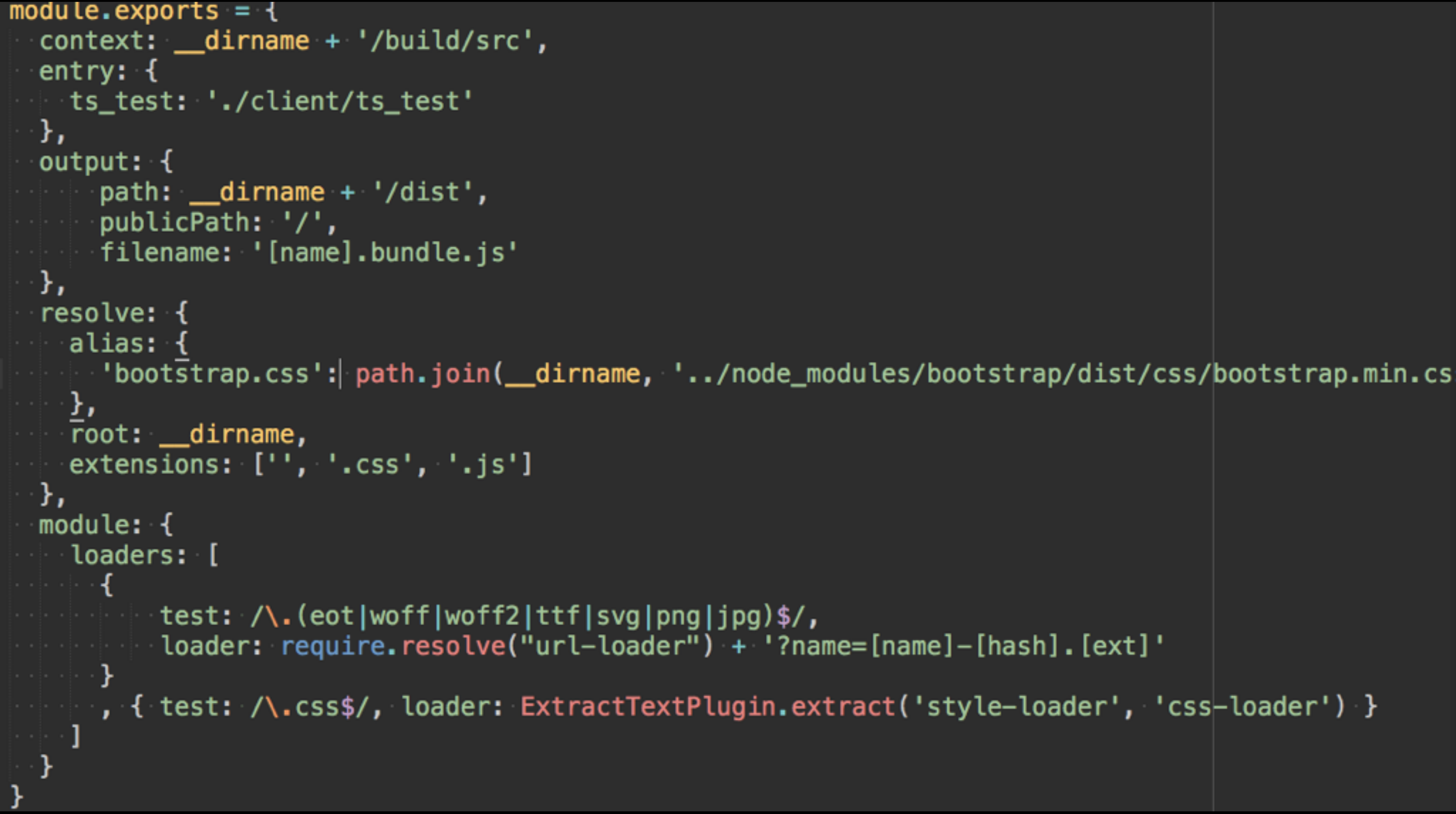
- Disadvantages
  - ts-loader cannot output the compiled file to outDir set in tsconfig.json.
  - Webpack only support different bundle name but not different folder. If we want to compile server side code, the result is in same folder and as a whole bundle.
  - TypeScript error will block the Webpack build process unless explicitly set ignoreDiagnostics in ts config.

How it works with Webpack - gulp

- Build the TypeScript (client & server) into a separate folder with exact structure of original folder

- Webpack source the TypeScript build folder instead of original folder

How it works with Webpack - gulp

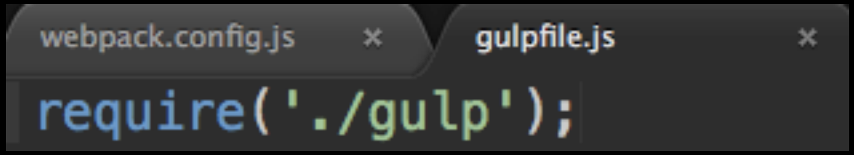


```
module.exports = {
  context: __dirname + '/build/src',
  entry: {
    ts_test: './client/ts_test'
  },
  output: {
      path: __dirname + '/dist',
      publicPath: '/',
      filename: '[name].bundle.js'
  },
  resolve: {
    alias: {
      'bootstrap.css': path.join(__dirname, '../node_modules/bootstrap/dist/css/bootstrap.min.cs
    },
    root: __dirname,
    extensions: ['', '.css', '.js']
  },
  module: {
    loaders: [
      {
          test: /\.(eot|woff|woff2|ttf|svg|png|jpg)$/,
          loader: require.resolve("url-loader") + '?name=[name]-[hash].[ext]'
      }
      , { test: /\.css$/, loader: ExtractTextPlugin.extract('style-loader', 'css-loader') }
    ]
  }
}
```

How it works with Webpack - gulp

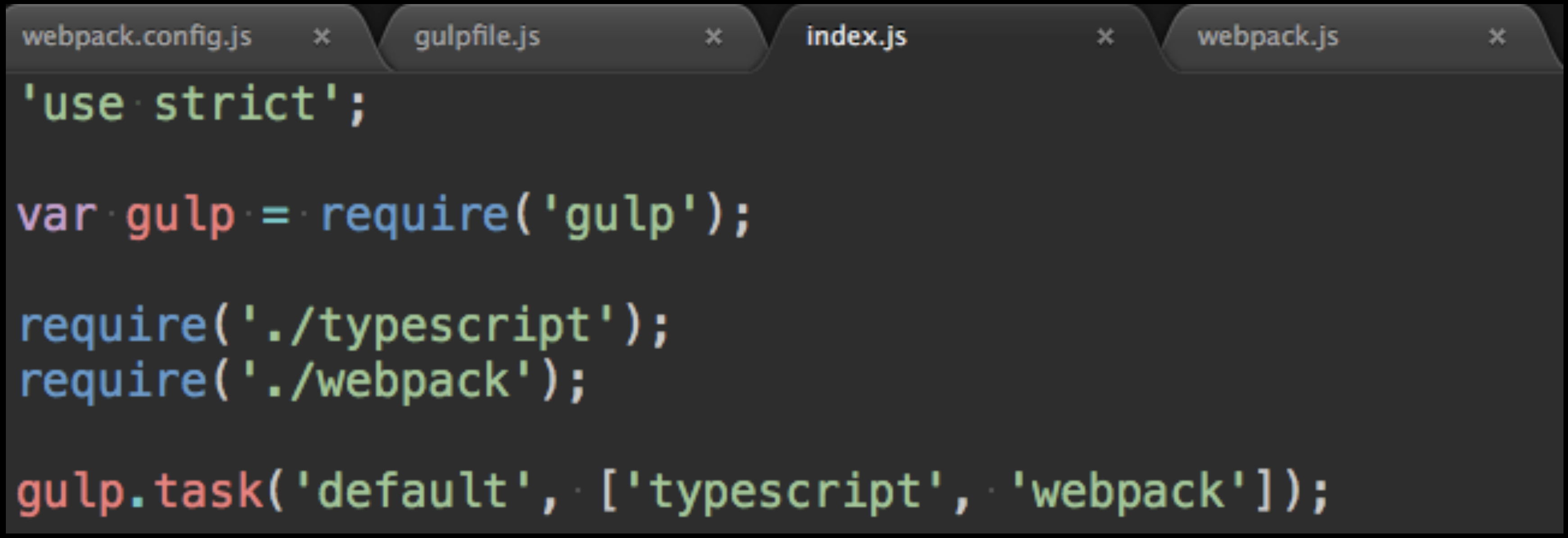
- Advantages
  - All TypeScript can be built exactly as the config
  - Folder structure is exactly retained.
  - TypeScript error can be safely ignored and not blocking build process

How it works with Webpack - gulp



webpack.config.js × gulpfile.js ×

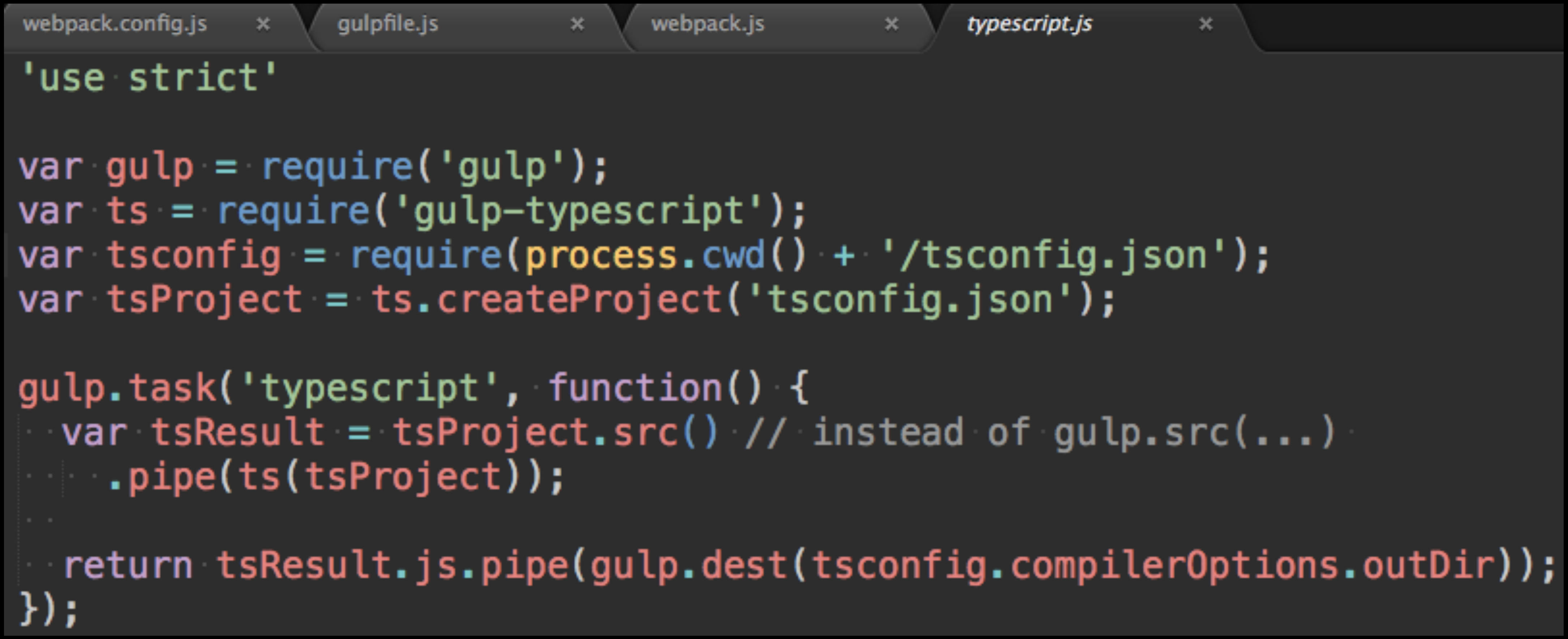
```
require('./gulp');
```



webpack.config.js × gulpfile.js × index.js × webpack.js ×

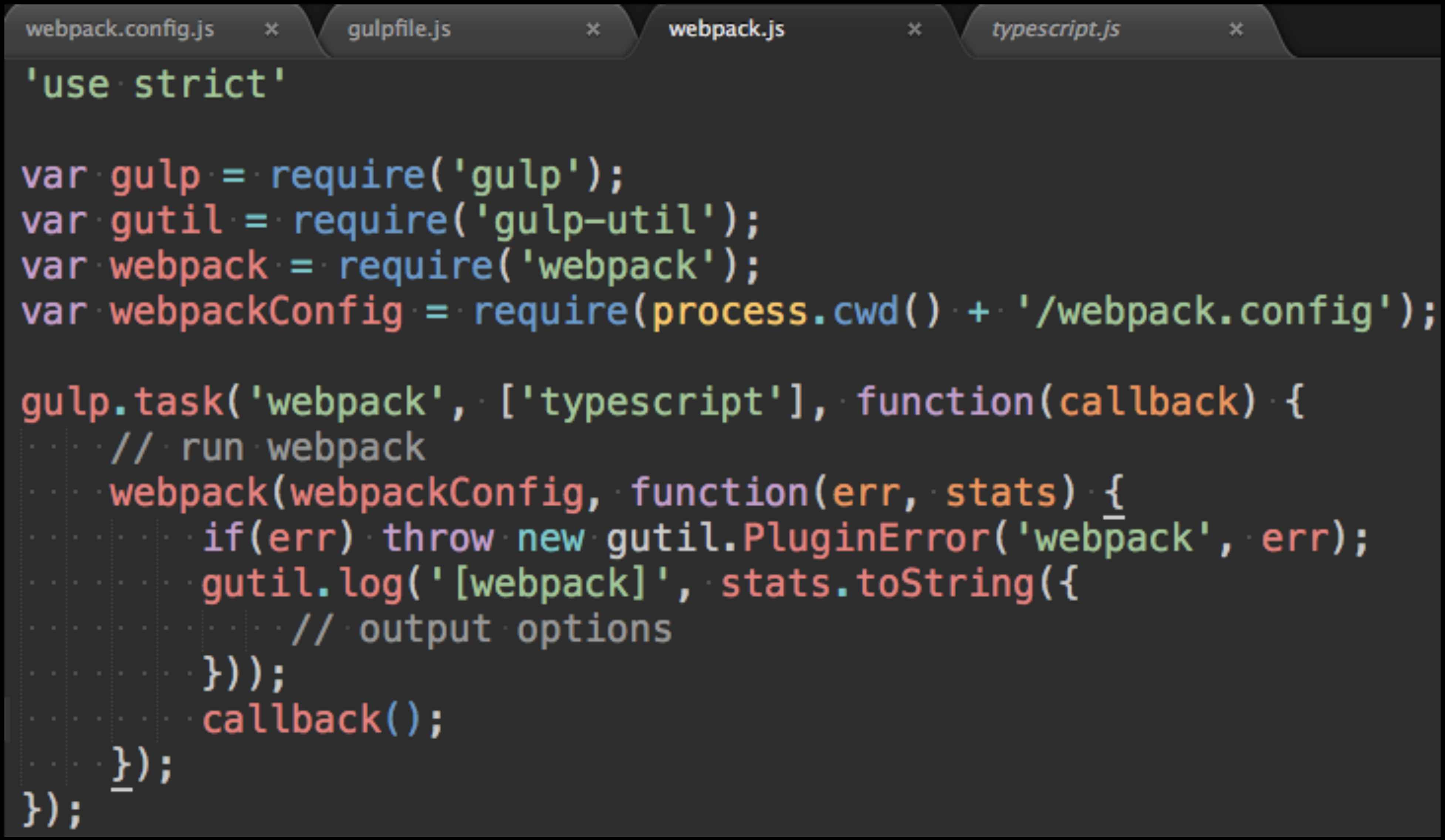
```
'use strict';

var gulp = require('gulp');

require('./typescript');
require('./webpack');

gulp.task('default', ['typescript', 'webpack']);
```

How it works with Webpack - gulp



```javascript
'use strict'

var gulp = require('gulp');
var ts = require('gulp-typescript');
var tsconfig = require(process.cwd() + '/tsconfig.json');
var tsProject = ts.createProject('tsconfig.json');

gulp.task('typescript', function() {
  var tsResult = tsProject.src() // instead of gulp.src(...)
      .pipe(ts(tsProject));

  return tsResult.js.pipe(gulp.dest(tsconfig.compilerOptions.outDir));
});
```

How it works with Webpack - gulp



```javascript
'use strict'

var gulp = require('gulp');
var gutil = require('gulp-util');
var webpack = require('webpack');
var webpackConfig = require(process.cwd() + '/webpack.config');

gulp.task('webpack', ['typescript'], function(callback) {
    // run webpack
    webpack(webpackConfig, function(err, stats) {
        if(err) throw new gutil.PluginError('webpack', err);
        gutil.log('[webpack]', stats.toString({
            // output options
        }));
        callback();
    });
});
```