

Developing with Node.js and Visual Studio Code



Agenda

- Introduction to Node.JS and NPM
- Installing and Updating Packages in Visual Studio Code
- Adding TypeScript Support to a Node.js Project
- Configuring Node.js with Server-side Debugging Support
- Using Gulp to Automate Running Development Tasks
- Developing Projects using Webpack



Cross-platform Toolchain

- Node.js
- Node Package Manager (npm)
- TypeScript
- Gulp
- Webpack



Installing node.js

- <https://nodejs.org/en/download/>



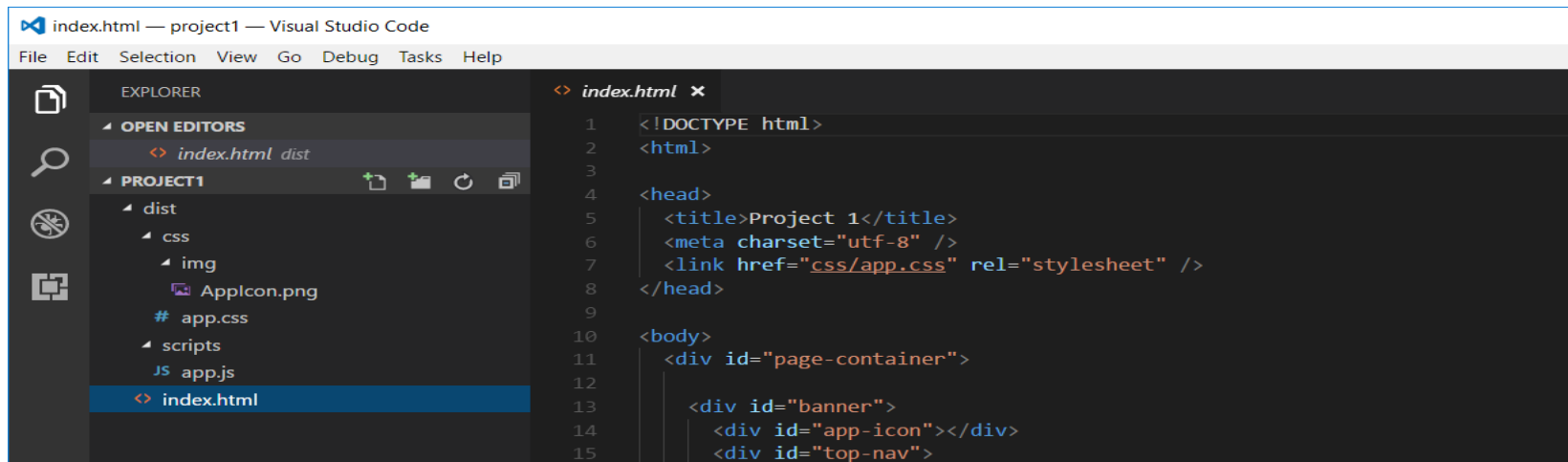
Install Visual Studio Code

- <http://code.visualstudio.com/>



Developing with Visual Studio Code

- Node.js is agnostic when it comes to developer IDE
 - There are many different IDEs that people use with Node.js
 - This course will be using Visual Studio Code

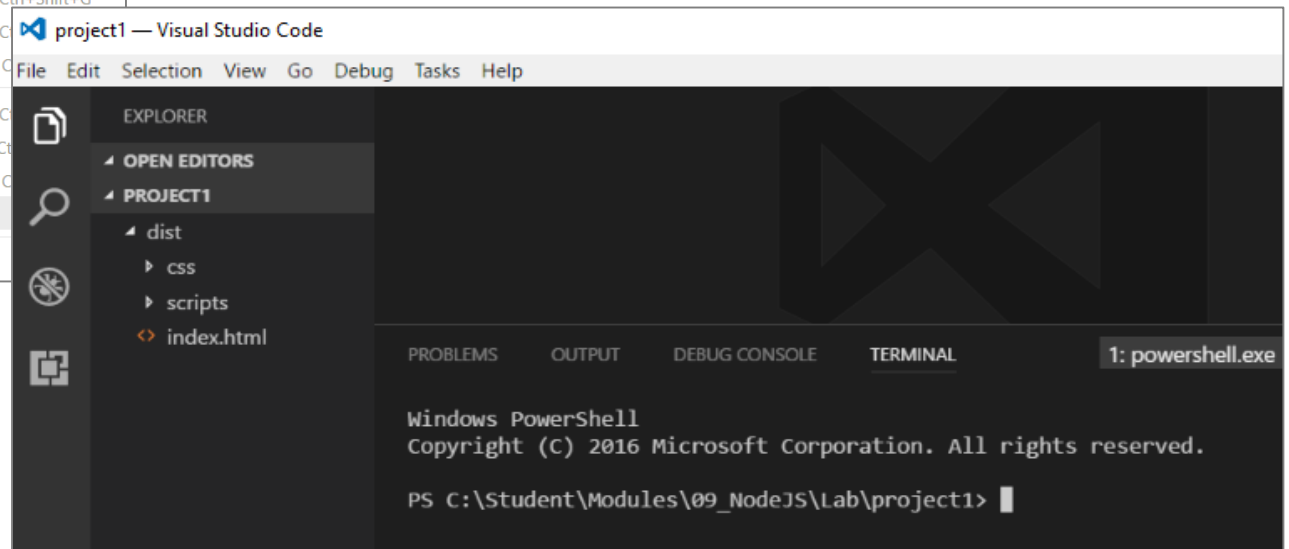
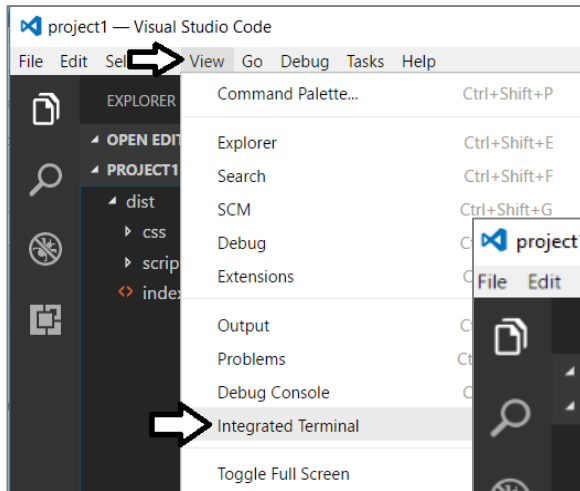


- Visual Studio is not a good fit for Node.js development
 - Visual Studio solution & project files incompatible with Node.js



Integrated Terminal

- Use the Integrated Terminal to execute `npm` command



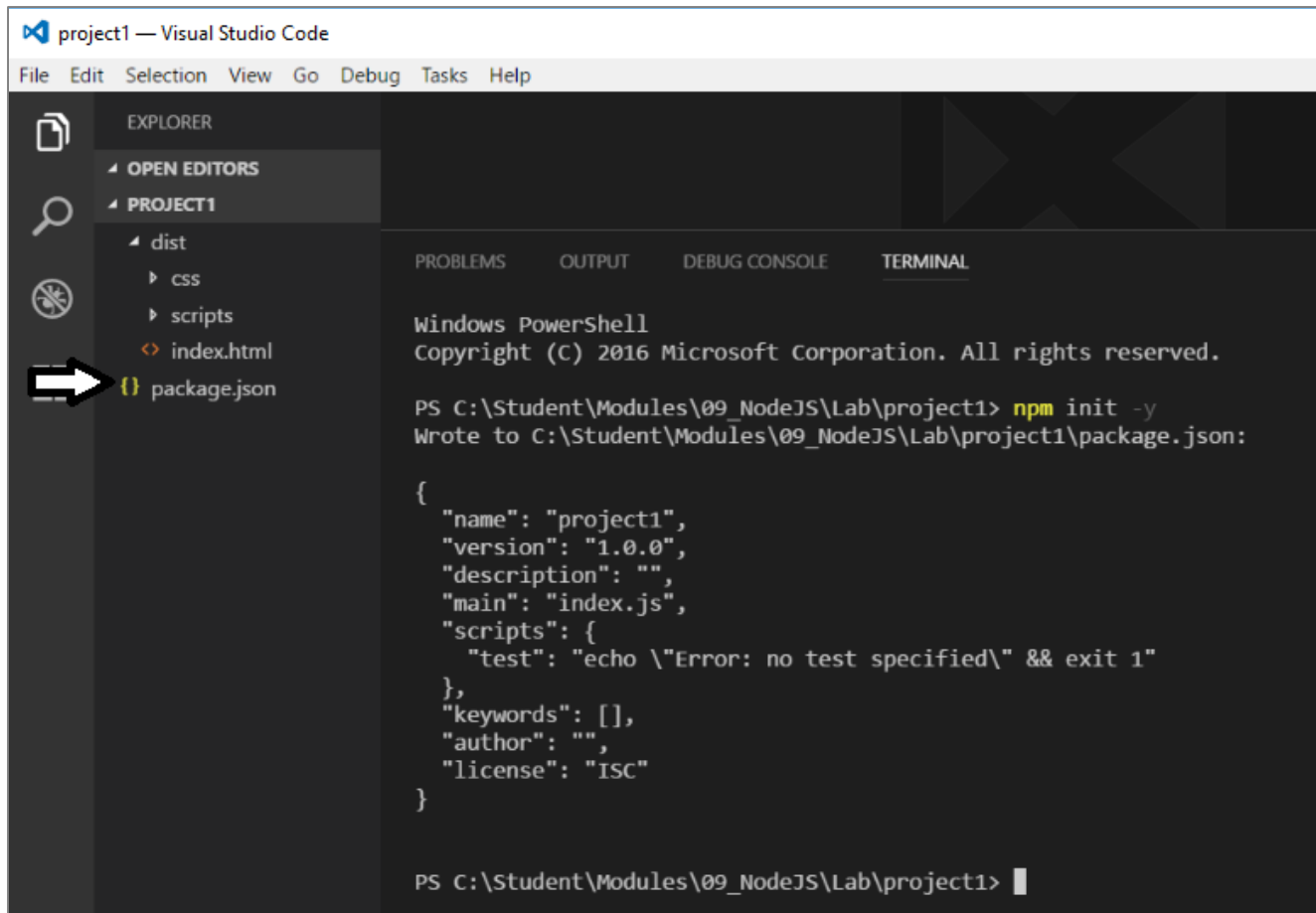
Agenda

- ✓ Introduction to Node.JS and Visual Studio Code
- Installing and Updating NPM packages
 - Configuring Server-side Debugging Support
 - Node.JS Development with TypeScript
 - Using Gulp to Automate Running Tasks
 - Bundling the Source Files using WebPack



npm init

- Node.js projects initialized with `npm init` command
- This command created the **package.json** file



The screenshot shows the Visual Studio Code interface for a project named 'project1'. The Explorer sidebar on the left displays the file structure: 'dist' (containing 'css' and 'scripts'), 'index.html', and 'package.json'. A white arrow points to the 'package.json' file, which is highlighted with a yellow icon. The TERMINAL panel on the right shows the output of the 'npm init -y' command, indicating that a 'package.json' file was written to the project directory. The content of the generated 'package.json' file is displayed in the terminal, showing default values for name, version, description, main, scripts, keywords, author, and license.

```
project1 — Visual Studio Code
File Edit Selection View Go Debug Tasks Help

EXPLORER
├─ OPEN EDITORS
├─ PROJECT1
│   └─ dist
│       ├── css
│       └── scripts
│   ├── index.html
│   └── package.json
└─ ...

TERMINAL
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Student\Modules\09_NodeJS\Lab\project1> npm init -y
Wrote to C:\Student\Modules\09_NodeJS\Lab\project1\package.json:

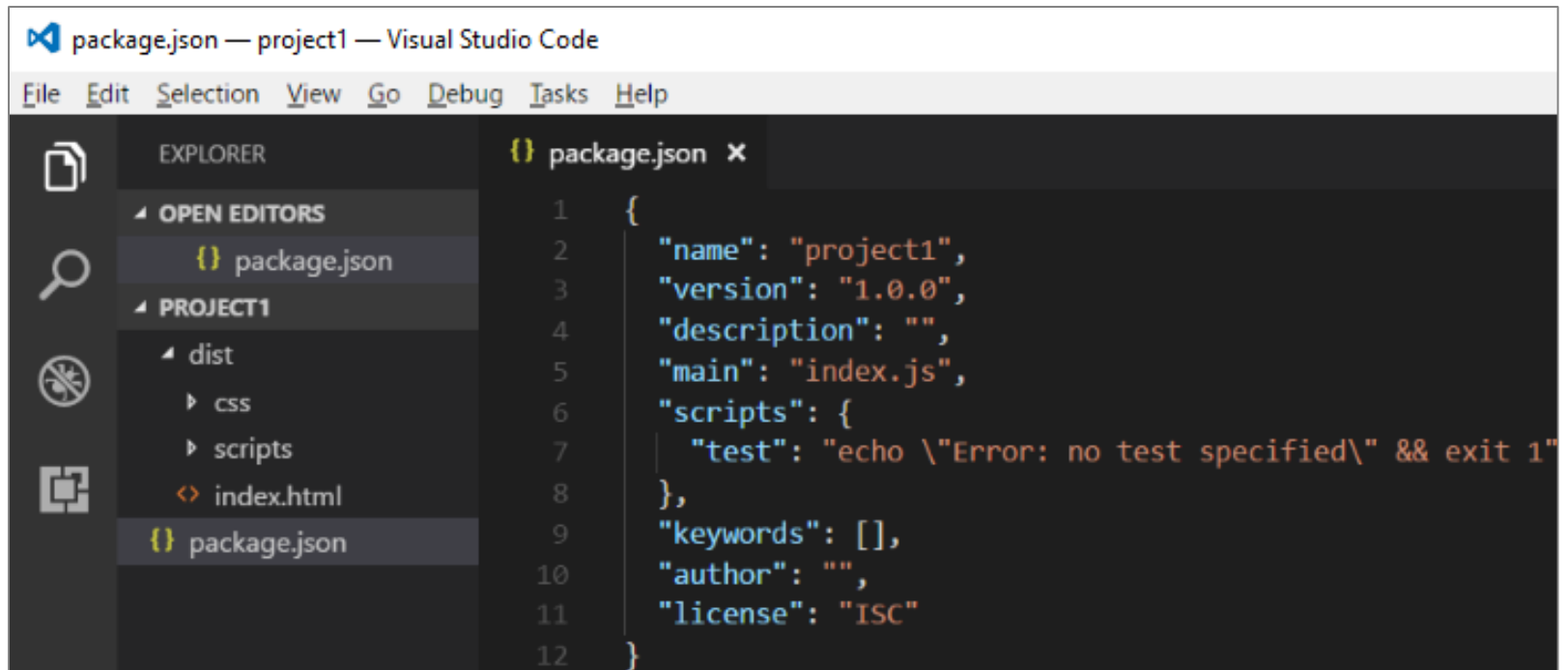
{
  "name": "project1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

PS C:\Student\Modules\09_NodeJS\Lab\project1>
```



package.json

- **package.json** serves as project manifest file
 - Tracks project name and version number
 - Tracks installed package dependencies



The screenshot shows the Visual Studio Code interface with a file named `package.json` open in the editor. The Explorer sidebar on the left shows the project structure, including a `dist` folder with `css` and `scripts` subfolders, and an `index.html` file. The `package.json` file is selected in the Explorer. The editor displays the following JSON content:

```
1 {  
2   "name": "project1",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC"  
12 }
```



Installing Packages

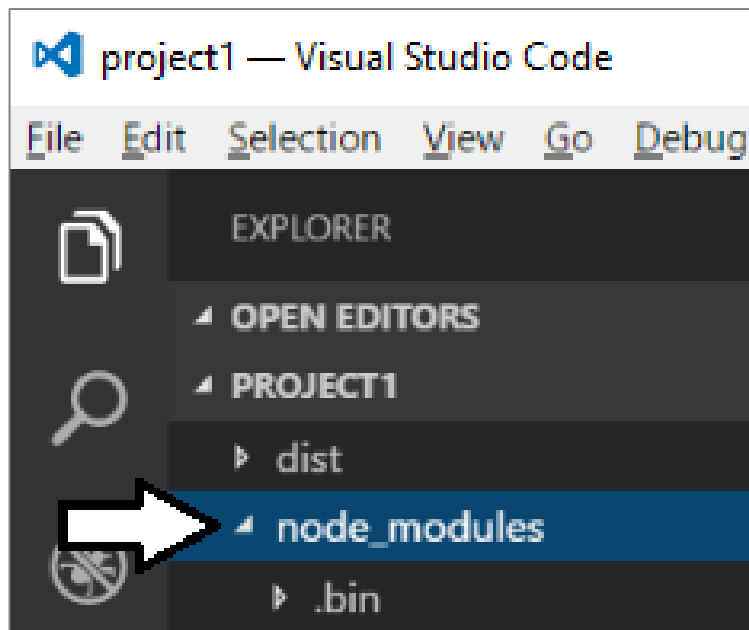
```
npm install browser-sync --save-dev
```

```
"devDependencies": {  
  "browser-sync": "^2.18.12"  
}
```



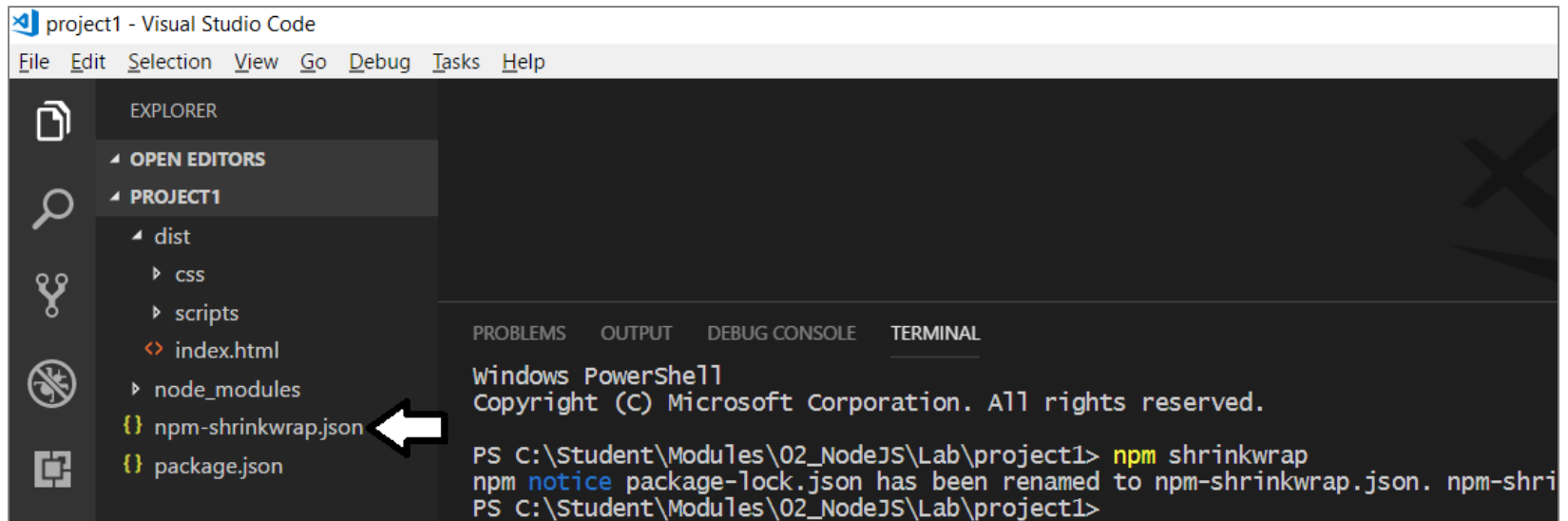
node_modules folder

- Package files copied into **node_modules** folder
 - This folder often contain 100s of packages for a project
 - Contents of folder not saved into source control
 - Contents can be restored with **npm install** command



package.lock.json vs npm.shrinkwrap.json

- **npm** generates files to track changes to **node-modules**
 - **package.lock.json** file initially created when installing packages.
 - **package.lock.json** file should not be checked into source control
- Running **npm shrinkwrap** generates **npm.shrinkwrap.json**
 - **npm.shrinkwrap.json** file can be checked into source control



The screenshot shows the Visual Studio Code interface for a project named 'project1'. The Explorer sidebar on the left displays the file structure, including 'dist', 'css', 'scripts', 'index.html', 'node_modules', 'npm-shrinkwrap.json', and 'package.json'. A white arrow points to 'npm-shrinkwrap.json'. The bottom panel shows the 'TERMINAL' tab with a Windows PowerShell prompt. The terminal output shows the command 'npm shrinkwrap' being executed, followed by a message: 'npm notice package-lock.json has been renamed to npm-shrinkwrap.json. npm-shri'. The prompt is then followed by 'PS C:\Student\Modules\02_NodeJS\Lab\project1>'.

```
project1 - Visual Studio Code
File Edit Selection View Go Debug Tasks Help

EXPLORER
├─ OPEN EDITORS
├─ PROJECT1
│  ├─ dist
│  │  └─ css
│  │  └─ scripts
│  │  └─ index.html
│  └─ node_modules
│     └─ npm-shrinkwrap.json
│     └─ package.json
└─ ...

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Student\Modules\02_NodeJS\Lab\project1> npm shrinkwrap
npm notice package-lock.json has been renamed to npm-shrinkwrap.json. npm-shri
PS C:\Student\Modules\02_NodeJS\Lab\project1>
```

Project Install vs Global Install

- Project installation adds package into project folder
 - Packages installed without -g parameter
`npm install [package_name] --save-dev`
 - **npx** command used to run CLIs from local **node_modules** folder
`npx package_cli`
- Global installation adds into shared package cache
 - Packages installed with -g parameter
`npm install -g [package_name]`
 - Package CLIs can be called directly from command line
`package_cli`



Agenda

- ✓ Introduction to Node.JS and Visual Studio Code
- ✓ Installing and Updating NPM packages
- Configuring Server-side Debugging Support
 - Node.JS Development with TypeScript
 - Using Gulp to Automate Running Tasks
 - Bundling the Source Files using WebPack



Configuring a Server-side Web Server

- Node.js does not provide its own web server
 - Instead, you must install a npm package to provide web server
 - There are many different packages to choose from
- Example packages which provide a web server for testing
 - http-server
 - express
 - Browser-sync (*this is the one we will be using*)

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install browser-sync --save-dev
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch"
+ browser-sync@2.24.6
added 222 packages in 23.346s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```



Using browser-sync to Serve Content

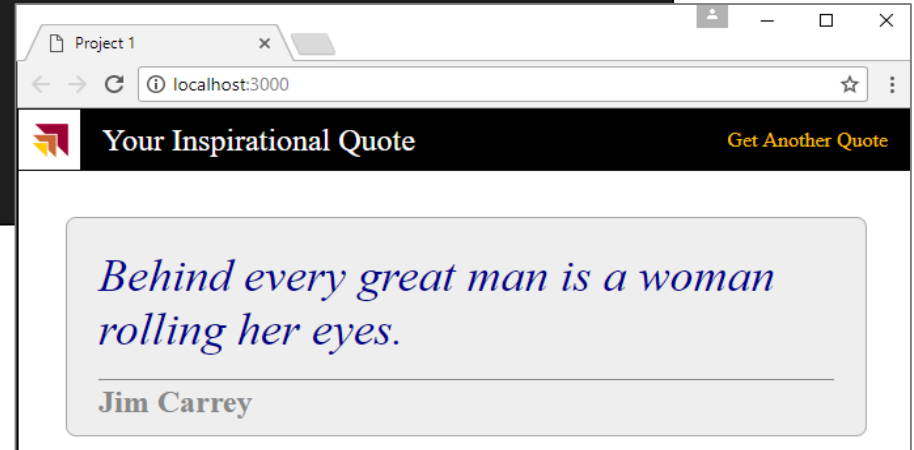
- **browser-sync start** command used to start web server
 - **--server** parameters references root folder with **index.html**

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> browser-sync start --server dist  
[Browsersync] Access URLs:
```

```
-----  
Local: http://localhost:3000  
External: http://10.0.0.3:3000  
-----
```

```
UI: http://localhost:3001  
UI External: http://10.0.0.3:3001  
-----
```

```
[Browsersync] Serving files from: dist
```



Stopping the Web Server Session


- Type CTRL + C into console to interrupt session

```
Local: http://localhost:3000
External: http://10.0.0.3:3000
-----
UI: http://localhost:3001
UI External: http://10.0.0.3:3001
-----
[Browsersync] Serving files from: dist
^CTerminate batch job (Y/N)? █
```



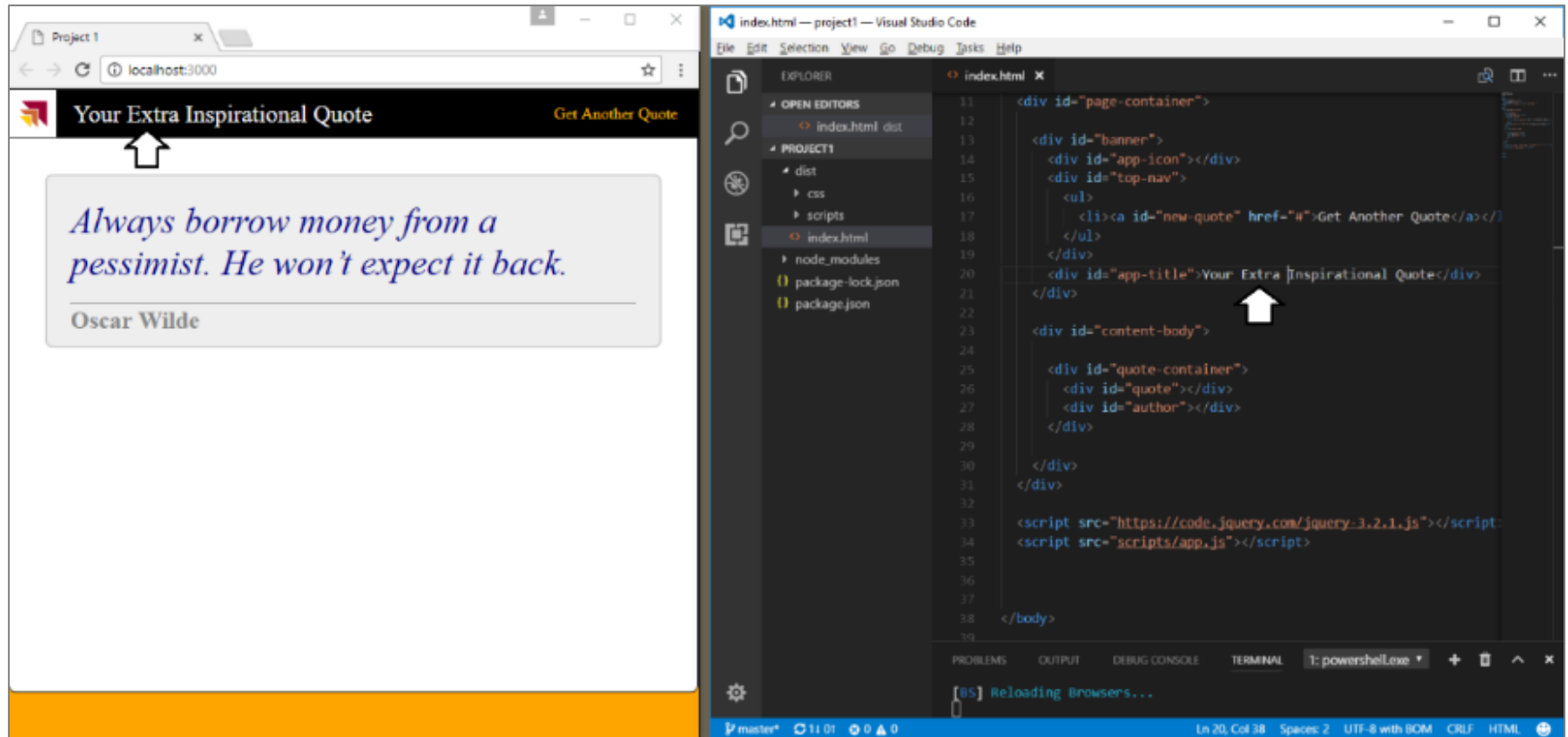
Starting Browser-sync with File Watching

- Browser-sync support `--files` parameter
 - `browser-sync start --server dist --files dist`

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> browser-sync start --server dist --files dist
[Browsersync] Access URLs:
-----
    Local: http://localhost:3000
    External: http://10.0.0.3:3000
-----
    UI: http://localhost:3001
    UI External: http://10.0.0.3:3001
-----
[Browsersync] Serving files from: dist
[Browsersync] Watching files... 
```



Automatic Updates



Agenda

- ✓ Introduction to Node.JS and Visual Studio Code
- ✓ Installing and Updating NPM packages
- ✓ Configuring Server-side Debugging Support
- Node.JS Development with TypeScript
 - Using Gulp to Automate Running Tasks
 - Bundling the Source Files using WebPack



Installing the TypeScript Package

- typescript package must be installed into project
 - Installed just like any other npm package

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install typescript --save-dev
npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.4 (node_modules\fsevents)
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4
+ typescript@3.0.1
added 1 package in 8.156s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```

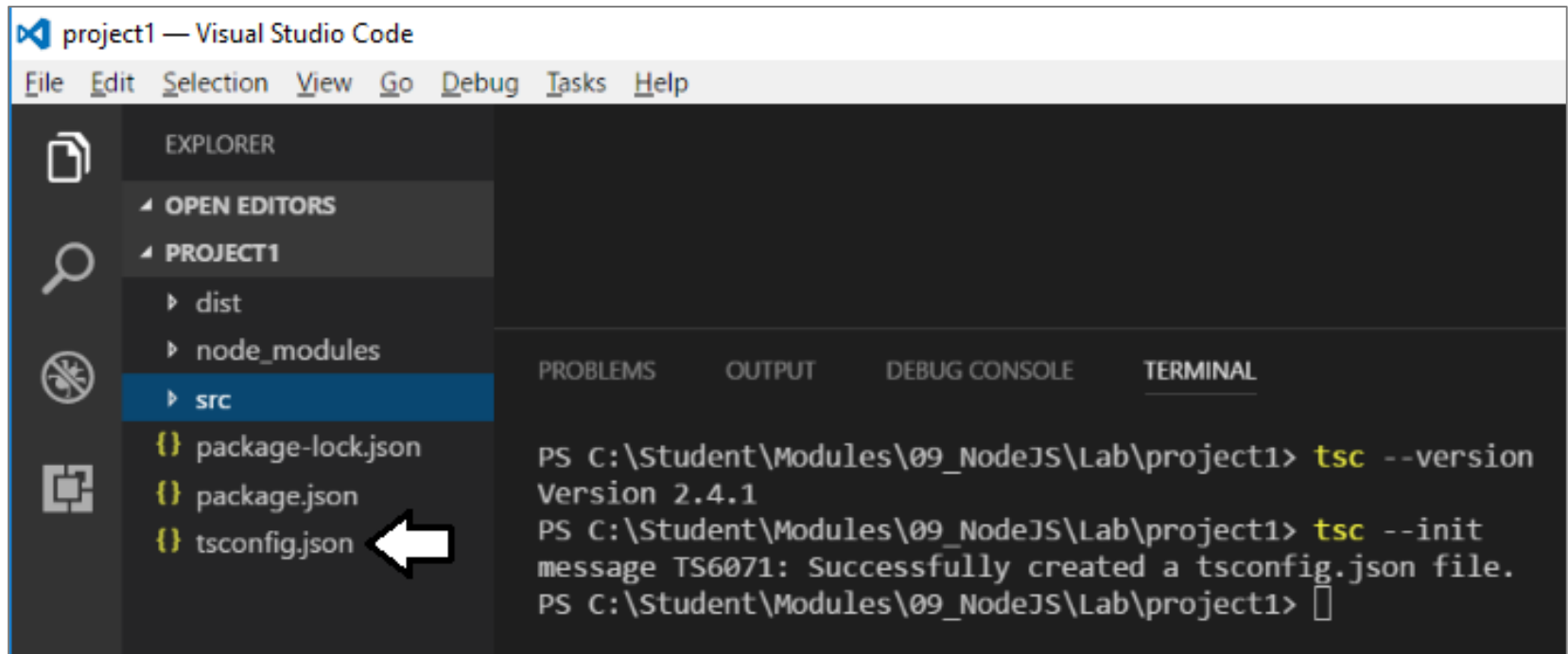
- Take note of version number of typescript package
 - typescript version may vary from one project to another
 - Determine project-specific version using **npx tsc --version**

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npx tsc --version
npx: installed 1 in 3.79s
Path must be a string. Received undefined
C:\Student\Modules\02_NodeJS\Lab\project1\node_modules\typescript\bin\tsc
Version 3.0.1
PS C:\Student\Modules\02_NodeJS\Lab\project1>
```



Generating tsconfig.json

- Typescript compilation controlled using **tsconfig.json** file
 - Generated using **tsc --init** command



The screenshot shows the Visual Studio Code interface for a project named 'project1'. The Explorer sidebar on the left displays the file structure: 'PROJECT1' contains 'dist', 'node_modules', and 'src'. Below these, the files 'package-lock.json', 'package.json', and 'tsconfig.json' are listed. A white arrow points to 'tsconfig.json'. The Terminal panel at the bottom shows the following commands and output:

```
PS C:\Student\Modules\09_NodeJS\Lab\project1> tsc --version
Version 2.4.1
PS C:\Student\Modules\09_NodeJS\Lab\project1> tsc --init
message TS6071: Successfully created a tsconfig.json file.
PS C:\Student\Modules\09_NodeJS\Lab\project1> 
```



tsconfig.json

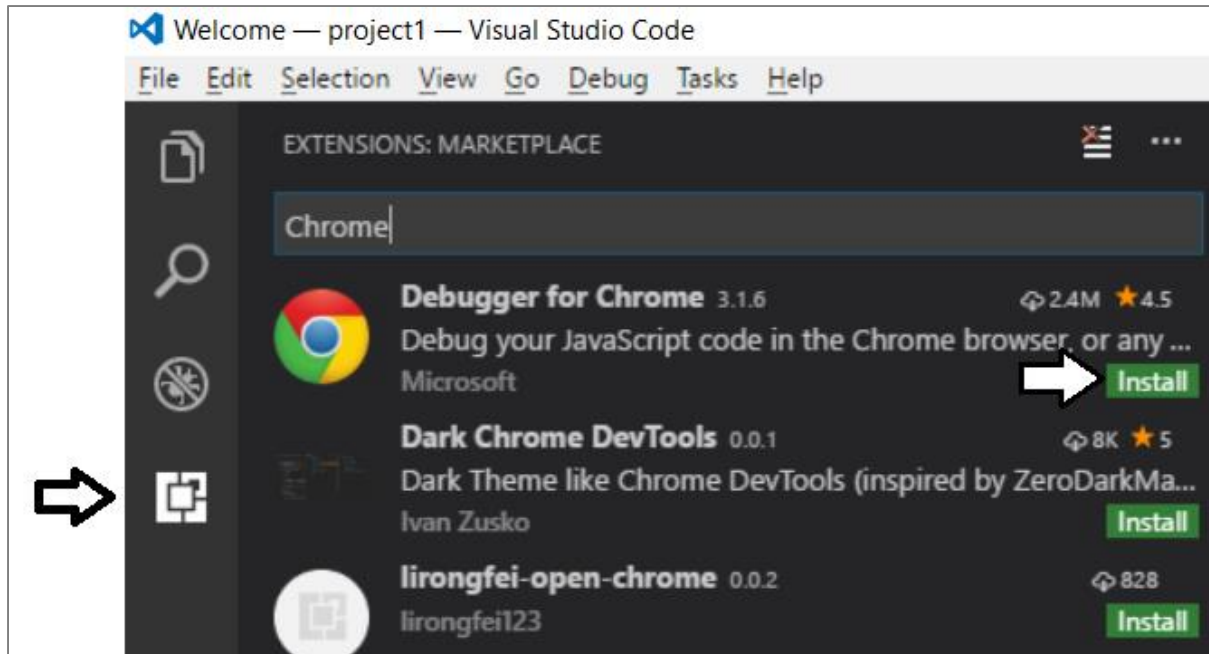
- Example of a **tsconfig.json** file

```
{  
  "compilerOptions": {  
    "noImplicitAny": true,  
    "removeComments": true,  
    "preserveConstEnums": true,  
    "outFile": "./dist/scripts/app.js",  
    "sourceMap": true,  
    "lib": [  
      "dom",  
      "es6"  
    ]  
  },  
  "files": [  
    "./src/scripts/app.ts"  
  ],  
  "exclude": [  
    "node_modules"  
  ]  
}
```



Chrome Debugging Support

- Visual Studio Code provides Chrome debugger extension
 - Provides ability to debug client-side Typescript code



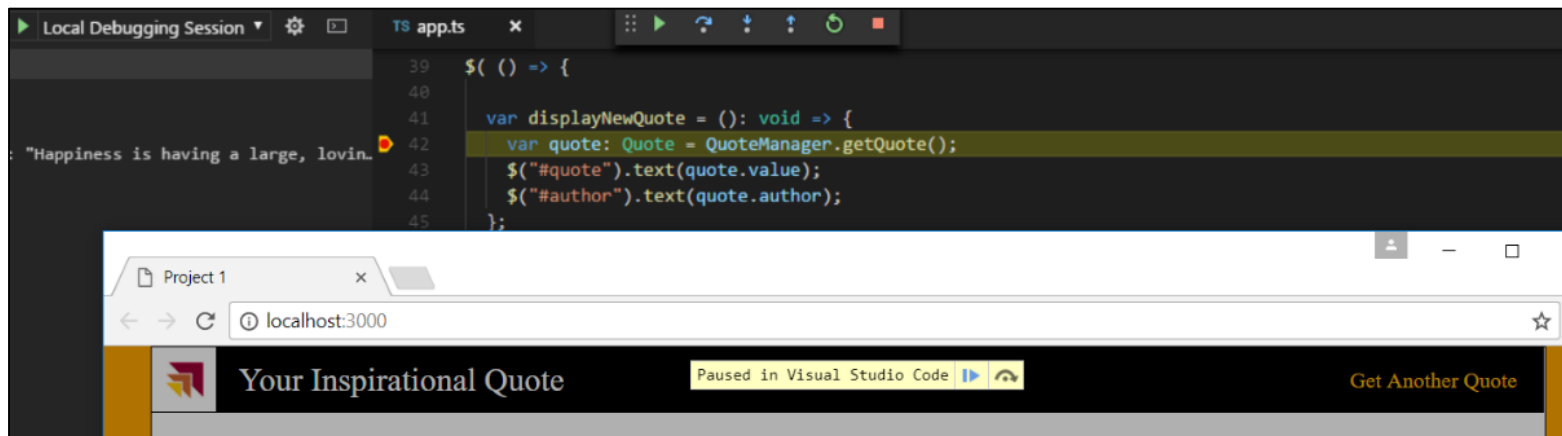
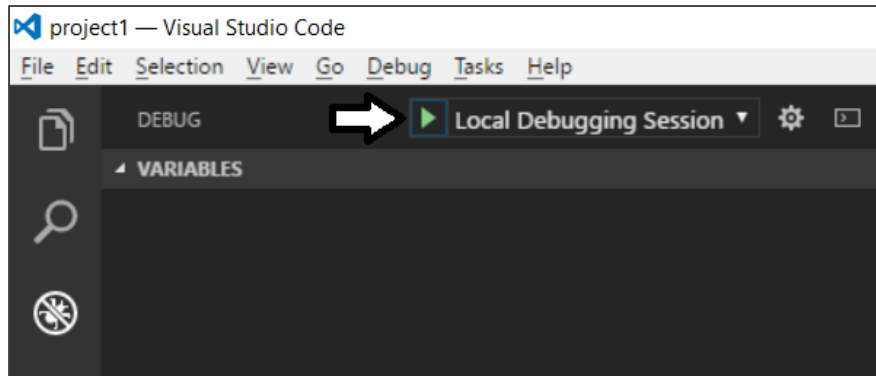
Visual Studio Debugging Support

- Debugging configurations tracked in launch.json

```
{ } launch.json ●
1  {
2      "version": "0.2.0",
3      "configurations": [
4          {
5              "name": "Local Debugging Session",
6              "type": "chrome",
7              "request": "launch",
8              "url": "http://localhost:3000/",
9              "webRoot": "${workspaceRoot}/dist",
10             "sourceMaps": true,
11             "runtimeArgs": [
12                 "--remote-debugging-port=9222"
13             ]
14         }
15     ]
16 }
```



Running the Debugger



Agenda

- ✓ Introduction to Node.JS and Visual Studio Code
- ✓ Installing and Updating NPM packages
- ✓ Configuring Server-side Debugging Support
- ✓ Node.JS Development with TypeScript
- Using Gulp to Automate Running Tasks
- Bundling the Source Files using WebPack



Gulp as a Task Runner

- Gulp serves as a Task Runner
 - Compiles TypeScript files to JavaScript
 - Compiles SASS files to CSS
 - Bundles and minifies JavaScript and CSS files

```
PS C:\Student\Modules\02_NodeJS\Lab\project1> npm install gulp --save-dev
npm WARN using --force I sure hope you know what you are doing.
npm WARN deprecated gulp-util@3.0.8: gulp-util is deprecated - replace it,
npm WARN deprecated graceful-fs@3.0.11: please upgrade to graceful-fs 4 for
npm WARN deprecated minimatch@2.0.10: Please update to minimatch 3.0.2 or
npm WARN deprecated minimatch@0.2.14: Please update to minimatch 3.0.2 or
npm WARN deprecated graceful-fs@1.2.3: please upgrade to graceful-fs 4 for
> fsevents@1.2.4 install C:\Student\Modules\02_NodeJS\Lab\project1\node_mo
> node install

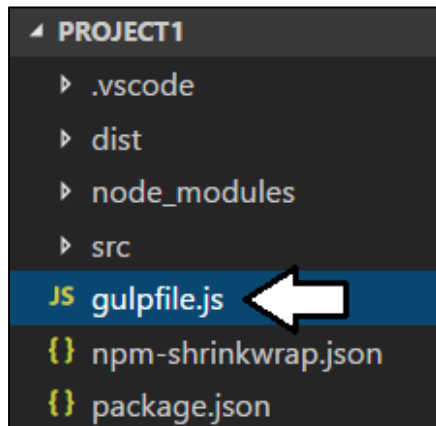
npm WARN project1@1.0.0 No description
npm WARN project1@1.0.0 No repository field.

+ gulp@3.9.1
added 75 packages in 6.404s
PS C:\Student\Modules\02_NodeJS\Lab\project1> █
```



gulpfile.js

- Gulp tasks are programmed inside **gulpfile.js**
 - **Gulpfile.js** must be added to root of project



```
JS gulpfile.js x
var gulp = require('gulp');
var clean = require('gulp-clean');
var ts = require("gulp-typescript");
var tsProject = ts.createProject("tsconfig.json");
var sourcemaps = require('gulp-sourcemaps');
var browserSync = require('browser-sync');

gulp.task('clean', function () {
  console.log("Running clean task");
  return gulp.src('dist/', { read: false })
    .pipe(clean());
});

gulp.task('build', ['clean'], function () {
  console.log("Running build task");

  gulp.src('src/**/*.html').pipe(gulp.dest('dist'));
  gulp.src('src/css/**/*.css').pipe(gulp.dest('dist/css'));
  gulp.src('src/css/img/**/*.png').pipe(gulp.dest('dist/css/img'));

  return tsProject.src()
    .pipe(sourcemaps.init())
    .pipe(tsProject())
    .pipe(sourcemaps.write('.', { sourceRoot: './', includeContent: false }))
    .pipe(gulp.dest("."));
});
```

Agenda

- ✓ Introduction to Node.JS and Visual Studio Code
- ✓ Installing and Updating NPM packages
- ✓ Configuring Server-side Debugging Support
- ✓ Node.JS Development with TypeScript
- ✓ Using Gulp to Automate Running Tasks
- Bundling the Source Files using WebPack



WebPack

- WebPack serves as a bundling utility
 - Bundles many js/ts files into a single file
 - Can handle dynamic module loading
 - Provides a dev server for testing and debugging
- When using Webpack 4
 - Install packages for webpack and webpack-cli
`npm install webpack webpack-cli --save-dev`



Dynamic Module Loading

- Webpack controls dynamic module loading
 - Your project just references app.ts
 - Compiler dynamically determines other files to include

```
TS app.ts x
import { Quote } from './quote';
import { QuoteManager } from './quote-manager';

$( () => {

  var displayNewQuote = (): void => {
    var quote: Quote = QuoteManager.getQuote();
    $("#quote").text(quote.value);
    $("#author").text(quote.author);
  }
});
```

```
TS quote.ts •
1 export class Quote {
2   value: string;
3   author: string;
4   constructor(value: string, author: string){
5     this.value = value;
6     this.author = author;
7   }
8 }
```

```
TS quote-manager.ts x
1 import { Quote } from './quote';
2
3 export class QuoteManager {
4
5   private static quotes: Quote[] = [
6     new Quote("Always borrow money from a pal", "John D. Rockefeller"),
7     new Quote("Behind every great man is a woman", "William Pitt"),
8     new Quote("In Hollywood a marriage is a business", "Marilyn Monroe")
9   ];
10 }
```



Webpack Loaders

- Loaders do two things
 - Identify which file or files should be transformed
 - Transform files and add them to dependency graph
- Example loaders
 - awesome-typescript-loader
 - style-loader
 - css-loader
 - url-loader



Webpack Plugins

- Webpack supports plugins in addition to loaders
 - commonly used to perform actions and custom functionality
 - Plugins act upon compilations or chunks of your bundled modules
- Examples Plugins
 - clean-webpack-plugin
 - copy-webpack-plugin
 - html-webpack-plugin



webpack.config.js

- Build process controlled through webpack.config.js

```
webpack.config.js
const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const CleanWebpackPlugin = require('clean-webpack-plugin')

module.exports = {
  entry: './src/scripts/app.ts',
  output: {
    filename: 'scripts/bundle.js',
    path: path.resolve(__dirname, 'dist'),
  },
  resolve: {
    extensions: ['.js', '.ts']
  },
  plugins: [
    new CleanWebpackPlugin(['dist']),
    new HtmlWebpackPlugin({ template: path.join(__dirname, 'src', 'index.html') }),
    new CopyWebpackPlugin([{ from: './src/favicon.ico', to: 'favicon.ico' }])
  ],
  module: {
    rules: [
      { test: /\.ts$/, loader: 'awesome-typescript-loader' },
      { test: /\.css$/, use: ['style-loader', 'css-loader'] },
      { test: /\.(png|jpg|gif)$/, use: [{ loader: 'url-loader', options: { limit: 8192 } }] }
    ],
  },
  mode: "development",
  devtool: 'source-map'
};
```



Webpack Builds

- Running build process generates files for distribution

```
PS C:\Student\Modules\02_NodeJS\Lab\project2> npm run build
> project2@1.0.0 build C:\Student\Modules\02_NodeJS\Lab\project2
> webpack

clean-webpack-plugin: C:\Student\Modules\02_NodeJS\Lab\project2\dist has been removed.
i |atl|: Using typescript@3.0.1 from typescript
i |atl|: Using tsconfig.json from C:/Student/Modules/02_NodeJS/Lab/project2/tsconfig.json
i |atl|: Checking started in a separate process...
i |atl|: Time: 595ms
Hash: 9bd924fdc1391178039d
Version: webpack 4.16.4
Time: 5486ms
Built at: 2018-08-02 16:29:28

```

Asset	Size	Chunks		Chunk Names
scripts/bundle.js	839 KiB	main	[emitted]	main
index.html	714 bytes		[emitted]	
favicon.ico	1.12 KiB		[emitted]	

```
Entrypoint main = scripts/bundle.js
[./node_modules/css-loader/index.js!./src/css/app.css] ./node_modules/css-loader!./src/css/app.css 1.89 KiB {main} [built]
[./src/css/app.css] 1.05 KiB {main} [built]
[./src/css/img/AppIcon.png] 981 bytes {main} [built]
[./src/scripts/app.ts] 505 bytes {main} [built]
[./src/scripts/quote-manager.ts] 2.38 KiB {main} [built]
[./src/scripts/quote.ts] 275 bytes {main} [built]
+ 5 hidden modules
Child html-webpack-plugin for "index.html":
  1 asset
  Entrypoint undefined = index.html
[./node_modules/html-webpack-plugin/lib/loader.js!./src/index.html] 880 bytes {0} [built]
[./node_modules/webpack/buildin/global.js] (webpack)/buildin/global.js 509 bytes {0} [built]
[./node_modules/webpack/buildin/module.js] (webpack)/buildin/module.js 519 bytes {0} [built]
+ 1 hidden module
PS C:\Student\Modules\02_NodeJS\Lab\project2> █
```

Webpack Dev Server

- Webpack provides its own development server
 - Install the webpack dev server package
`npm install webpack-dev-server --save-dev`
 - Run your project using the webpack dev server CLI
`webpack-dev-server --open`



Summary

- ✓ Introduction to Node.JS and Visual Studio Code
- ✓ Installing and Updating NPM packages
- ✓ Configuring Server-side Debugging Support
- ✓ Node.JS Development with TypeScript
- ✓ Using Gulp to Automate Running Tasks
- ✓ Bundling the Source Files using WebPack

