KLIANT

# Using Visual Studio Code with TypeScript

Anthony Sneed: @tonysneed tony.sneed@cloud.com

Eric Swanson: eswanson@kliant.com

# About Me

- **Personal**
  - Married, three children
  - Los Angeles, Dallas, Slovakia
- **Work**
  - Consultant, Author, Instructor
- **Open Source**
  - Trackable Entities
  - Simple MVVM Toolkit

# Contact Me

- **Email**
  - tony.sneed@icloud.com
- **Blog**
  - blog.tonysneed.com
- **Social Media**
  - Twitter: @tonysneed
  - Google+: AnthonySneedGuru
  - Facebook: anthony.sneed
  - LinkedIn: tonysneed


# Get the Bits

- Available on **GitHub**:
  **http://github.com/tonysneed/**
- **Slides**: **Kliant.VSCodeTypeScript**
- **Code**: **Demo.VSCode.TypeScript**
- **Yeoman Generator**:
  **generator-tonysneed-vscode-typescript**

# Objectives

- **Background**
  - What is **TypeScript** ?
  - What is **Visual Studio Code** ?
- **Additional tools**
  - Package management: **NPM**
  - Task runners: **Gulp**
  - Project scaffolding: **Yeoman**
- **Development**
  - **Intellisense**, **compiling**, **debugging**, **linting**
- **Testing**
  - **Jasmine**, **Karma**, **PhantomJS**, **Travis CI**

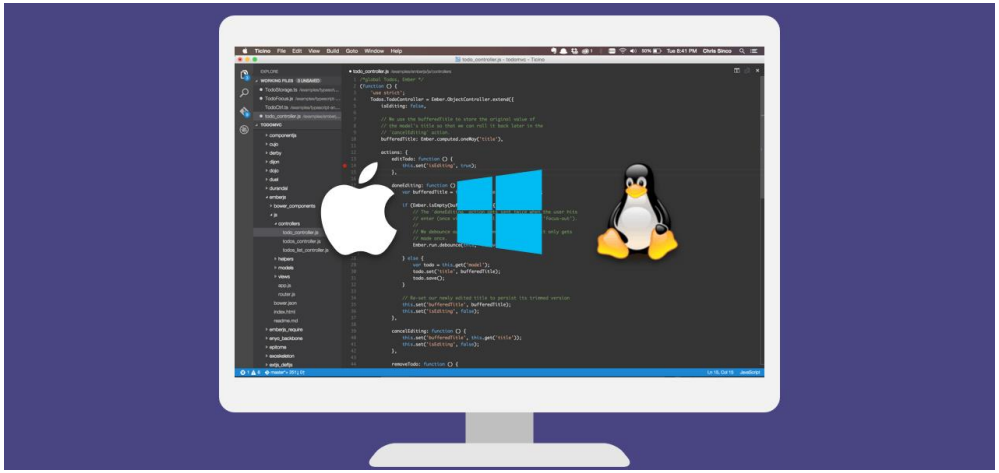
# Some Background ...

# What is TypeScript?

- **TypeScript = JavaScript**
  - All JavaScript is valid TypeScript
  - Future JavaScript features (ES 2015, etc)
    - Classes, interfaces, namespaces, modules
- **+ Static Typing**
  - Compile-time error checking
  - Type safety
  - Intellisense
  - Code refactoring

# What is Visual Studio Code?

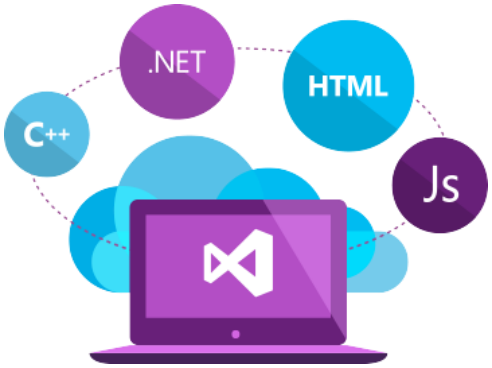
- **Souped up code editor**
  - Folder-based vs project-based
  - Syntax highlighting, code completion
  - Task Runner
  - Debugging
  - Git integration
  - Extensions



# Why Visual Studio Code?

- **Lightweight**
  - Runs on **basic hardware**
  - Not a **memory** hog
- **Cross-platform**
  - Windows, **Mac** or Linux
- **Open-source**
  - Contribute on **GitHub**
- **Extensible**
  - Lots of plugins on **extensions gallery**

# Why Not Visual Studio 2015?




- **Windows-only**
  - Native or **virtual machine**
- **Resource hog**
  - **Memory**, disk space
  - **Time**: downloads, installation and updates
- **Plain-vanilla TypeScript template**
  - Doesn't **buy** you much (still work to do)

# What's Missing from VS Code?

- **No File - New Project**
  - Starting from **scratch** can be difficult
- **Scaffold new projects with Yeoman**
  - Find a **generator** to suit your needs
  - *Or roll your own!*

YEOMAN

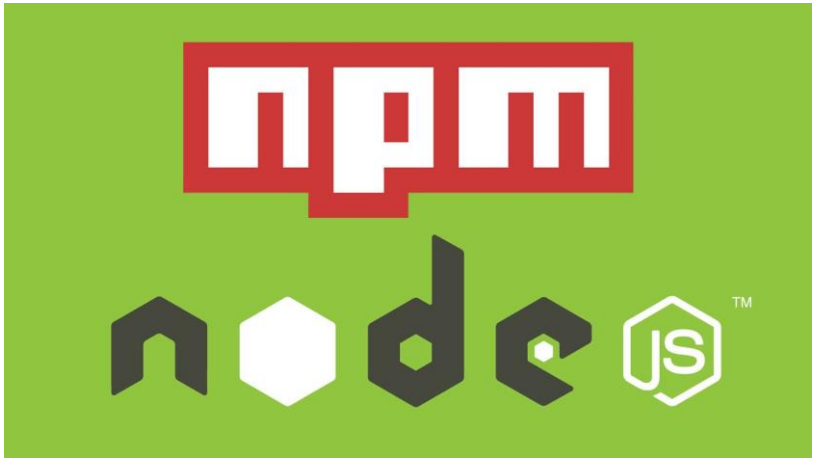
# Additional Tools ...

# Node JS

- **JavaScript runtime**
  - Built on Chrome's **V8** JS engine
  - Event-driven, **non-blocking** I/O

# NPM: Node Package Manager

- **Install packages globally**
  - For ex: **typescript** compiler, **gulp** task runner
- **Install packages locally**
  - Dependencies managed via **package.json** file



# Gulp Task Runner

- **Lets you automate common tasks**
  - Based on **streams** vs config files
  - Add tasks in **JavaScript** to a **gulpfile**
  - There are numerous **plugins**
  - Can **chain** tasks together
  - **VS Code** recognizes gulp tasks
- **For example:**
  - **Transpiling** TypeScript
  - Validating code quality: **linting**
  - Running **tests**
  - Build steps: **bundling**, **minification**

# Demo:
# TypeScript from Scratch

# Steps: TypeScript from Scratch

1. Prerequisites
   - Install Node.js: https://nodejs.org
   - **npm install -g typescript**
2. Initialize a project with npm
   - **npm init –y**
3. New folder: src/; new file: **app.ts**
   - **Greeter class**, **ctor**, **greet** method
   - Create new Greeter, console.log greeting
4. Compile: **tsc app.ts** -> app.js
   - Run: **node app.js**

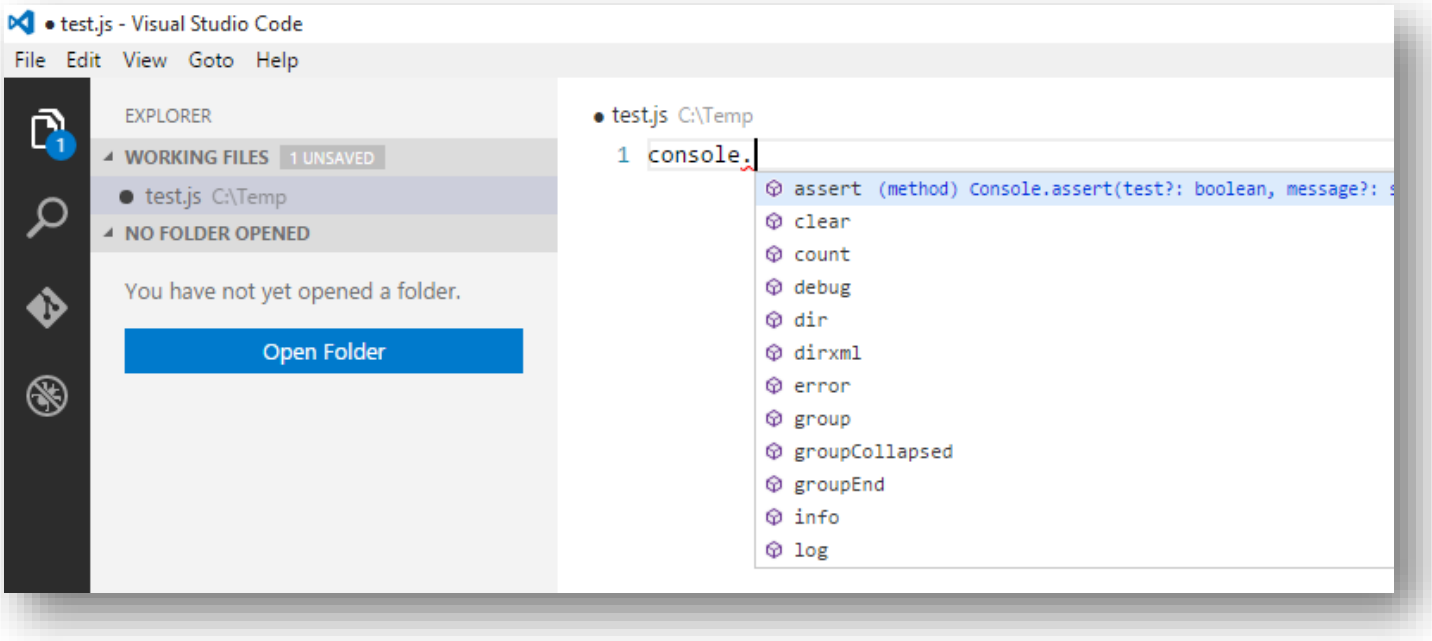
# Code: TypeScript from Scratch

```
/**
 * Greeter
 */
class Greeter {
    constructor(public message: string) {

    }
    greet(): string {
        return "Hello " + this.message;
    }
}

var greeter = new Greeter("World");
var msg = greeter.greet();
console.log(msg);
```

# Development ...

# Intellisense: Default

- VS Code uses TypeScript **definition files** to provide **intellisense**
  - **Primitive types** and the **DOM** already **included**



# Intellisense: JavaScript Libraries

- Use **typings** to install more type defs
  - Provides intellisense to **JavaScript** libraries
  - Replaces **tsd** tool, which is now deprecated

```
$ npm install –g typings
```

```
$ typings install node --save --ambient
```

# Compiling: tsconfig.json

- Place **tsconfig.json** file at **source root**
  - JSON schema provides **intellisense**

```
{
    "compilerOptions": {
        "module": "commonjs",
        "target": "es5",
        "sourceMap": true,
        "declaration": true,
        "noImplicitAny": true,
        "rootDir": ".",
        "outDir": "../dist"
    }
}
```

# Compiling: tsc task

- Configure VS Code tasks with **task.json** file
  - Specify **tsc** for "**command**"; **-p** and **src** for "**args**"
  - Press **Cmd+Shift+B** to compile

```
{
    "version": "0.1.0",
    "command": "tsc",
    "isShellCommand": true,
    "showOutput": "silent",
    "args": ["-p", "src"],
    "problemMatcher": "$tsc"
}
```

# Compiling: run npm script

- Add **script** to **package.json** file
  - Remove **dist** directory
  - Invoke **tsc** with **project** argument
    - Uses **tsconfig.json** in specified directory

```
"scripts": {
    "typings": "tsd install -r -o --save-dev",
    "compile": "rm -rf dist && tsc -p src",
    "test": "gulp tests-run"
}
```

```
$ npm run compile
```

# Compiling: gulpfile.js

- Add a **gulpfile.js** file to the project
  - Install **gulp** locally: **npm --save-dev install gulp**
  - Run gulp tasks from the **command line**

```
var gulp = require('gulp');
var exec = require('child_process').exec;

gulp.task('compile', function () {

    exec('rm -rf dist && tsc -p src');
});
```

```
$ gulp compile
```

# Compiling: gulp task

- Update **tasks.json** for **gulp**
  - Specify **gulp** for "**command**" with gulp **task name**
  - Press **Cmd+Shift+B** to compile with gulp task

```
{
    "command": "gulp",
    "isShellCommand": true,
    "tasks": [ {
            "taskName": "compile",
            "isBuildCommand": true,
            "showOutput": "always",
            "problemMatcher": "$tsc"
        }
    ]
}
```

# Debugging

- Press **F5** to add **launch.json** file
  - Specify **sourceMaps** , **cwd**, **outDir**

```
{
    "configurations": [ {
            "name": "Debug Current File",
            "type": "node",
            "request": "launch",
            "program": "${file}"
            "stopOnEntry": false,
            "sourceMaps": true,
            "cwd": "${workspaceRoot}",
            "outDir": "${workspaceRoot}/dist"
        } ]
}
```

# Demo:
# Compiling and Debugging

# Refactoring: Export and Import

1. Mark Greeter class with **export**
   - Move **greeter.ts** into a **greeter folder**
2. Move code *outside* of Greeter to **app.ts**
   - Add an **import** statement
   - Change **var** to **let**

```
export class Greeter { // Remaining code elided for clarity
```

```
import { Greeter } from "./greeter/greeter";

let greeter = new Greeter("World");
let msg = greeter.greet();
console.log(msg);
```

# Compiling: tsc, npm, gulp

1. Install **typings**
   - Add typings for **node**
2. Add **tsconfig.json**
   - Place in the **src** folder
3. Add **tsc** task to **tasks.json**
   - Validate paths for **source maps**
4. Add "compile" **npm script** to **package.json**
5. Add "compile" **gulp** task
   - Update **tasks.json** to use **gulp**
6. Add **launch.json** to enable **debugging**

# Testing …

# Testing: Jasmine

- **Jasmine** is a **behavior-driven** development **testing** framework
  - Has built-in **assertions** (**expect**) and **mocks** (**spies**)
  - **Suites**: **describe** functions; **specs**: **it** functions

# Installing Jasmine

- Install **jasmine-core** and jasmine **typings**
- Unzip GitHub **release**: **jasmine-standalone**
  - Copy **lib** folder and **specrunner.html** to project

```
$ npm install –-save-dev jasmine-core
```

```
$ typings install jasmine --save --ambient
```

# Writing Jasmine Tests: JavaScript

- **Write** a jasmine test: **sample.spec.js**
- **Include spec file** in specrunner.html
- **Open** specrunner.html in a **browser**

```
describe("A suite", function() {
  it("contains spec with an expectation", function() {
    expect(true).toBe(true);
  });
});
```

```
<script src="src/sample.spec.js"></script>
```

# Serving Jasmine Tests

- **Refreshing** the browser can become **tedious**
  - Install **browser-sync**, create **gulp** task

```
gulp.task('serve-tests', function () {
    var options = {
        port: 3000, server: './',
        files: ['./dist/**/*.js',
                './dist/**/*.spec.js',
                '!./dist/**/*.js.map'],
        logPrefix: 'spec-runner',
        reloadDelay: 1000,
        startPath: 'SpecRunner.html' };

    browserSync(options);
});
```

# Writing Jasmine Tests: TypeScript

- **Write** test in TypeScript: **greeter.spec.ts**

```
/// <reference path="../../typings/main.d.ts" />
import {Greeter} from "./greeter";
describe("Greeter", () => {
    describe("greet", () => {
        it("returns Hello World", () => {
            // Arrange
            let greeter = new Greeter("World");
            // Act
            let result = greeter.greet();
            // Assert
            expect(result).toEqual("Hello World");
        });
    });
});
```

# Running Jasmine Tests: TypeScript

- Install **system.js**
  - Update **specrunner.html** to use **system.js**

```
$ npm install --save-dev systemjs
```

```
<script src="node_modules/systemjs/dist/system.src.js"></script>

<script>
    System.config({ packages: {
        'dist': {defaultExtension: 'js'}}});
    Promise.all([System.import('dist/greeter/greeter.spec')]);
</script>
```

# Demo:
# Writing Tests with Jasmine

# Running Tests with Karma

- **Karma** is useful for *running tests*
  - from the **terminal**
  - in multiple **browsers**
  - on **continuous integration** servers

# Configure Karma

- Add a **karma.conf.js** file
  - Use **plugins** for **typescript** and **jasmine**
  - Use **PhantomJS** headless **browser**
  - Use **module loaders** and **polyfills** for **phantomjs**

```
$ npm install --save-dev karma karma-jasmine
  karma-phantomjs-launcher karma-systemjs
  karma-typescript-preprocessor
```

# Gulp Task for Running Karma

- Add a **gulp** task to run **karma** tests
  - Start **watch** by setting **singleRun** to false

```
gulp.task('run-tests', ['compile'], function () {

 var server = require('karma').Server;

    var server = new Server({
        configFile: __dirname + '/karma.conf.js',
        singleRun: true // false for watch
    });

    server.start();
});
```

# Continuous Integration with Travis CI

- Run **tests** when commits **pushed** to **GitHub**
  - Create **account**, log in, add **repositories**
  - Add **.travis.yml** file to run scripts

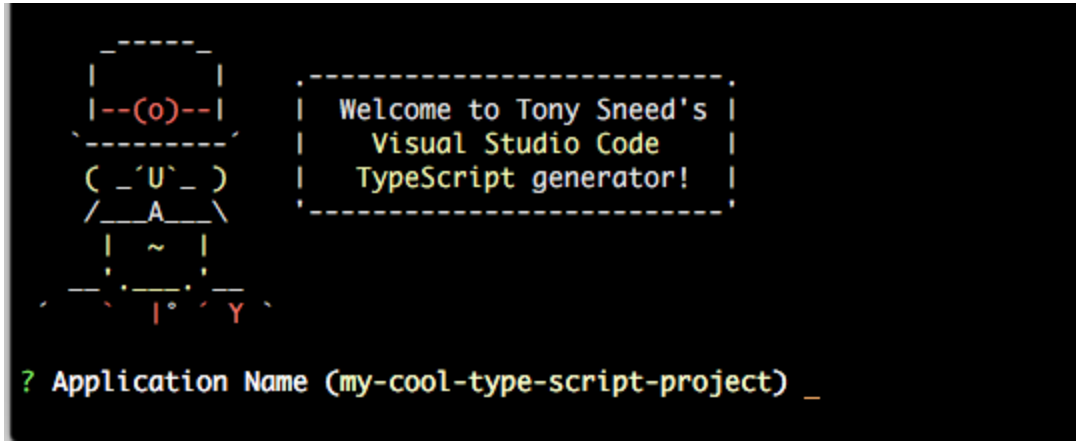
```
language: node_js
node_js:
  - 'stable'
script: node_modules/karma/bin/karma
        start karma.conf.js --single-run
before_script:
    - npm install
```

# Wrap Up: Using the Yeoman Generator

- Install **yo** and **generator** **globally**
  - Create project directory, run the generator

```
$ npm install -g yo generator-tonysneed-vscode-typescript
```

```
$ yo tonysneed-vscode-typescript
```



# Demo:
# Using the Yeoman Generator Running Tests with Karma

# Contact Me

- **Email**
  - tony.sneed@icloud.com
- **Blog**
  - blog.tonysneed.com
- **Social Media**
  - Twitter: @tonysneed
  - Google+: AnthonySneedGuru
  - Facebook: anthony.sneed
  - LinkedIn: tonysneed


# Get the Bits

- Available on **GitHub**:
  **http://github.com/tonysneed/**
- **Slides**: **Kliant.VSCodeTypeScript**
- **Code**: **Demo.VSCode.TypeScript**
- **Yeoman Generator**:
  **generator-tonysneed-vscode-typescript**

KLIANT

# Using Visual Studio Code with TypeScript

Anthony Sneed: @tonysneed tony.sneed@cloud.com

Eric Swanson: eswanson@kliant.com