

# PROMISES WITH GENERICS USING TYPESCRIPT

# WHO AM I?

Brennan Stehling

Long time JavaScript Developer

JavaScript > AJAX > Node.js > TypeScript



# SUMMARY

- ▶ Why use Promises?
- ▶ What are Generics?
- ▶ What is TypeScript?
- ▶ Which IDE can I use?
- ▶ How can I automated this transpiling?
- ▶ How can I ensure my coding standards?

# WHY USE PROMISES?

- ▶ Introduced with ES6
- ▶ Callback Hell
- ▶ Function Chaining
- ▶ Cleaner Syntax

# WHAT ARE GENERICS?

```
export function eachItem<T>(  
  items : T[],  
  processItemFunction: (item: T) => Pr  
  let index = 0;  
  let promise = new Promise<any>((re  
  let continueWhile = function(n  
    let item = nextItem();  
    if (item) {  
      processItem(item).then  
        index++;
```

# WHAT IS TYPESCRIPT?

- ▶ Superset of JavaScript
- ▶ Includes Promises & Generics
- ▶ ES5 and ES6 compatible
- ▶ TS is transpiled to JS
- ▶ Visit [typescriptlang.org](https://typescriptlang.org)

```
let doAsyncWork = function (): Promise<Message> {
  let p = new Promise<Message>(function (resolve, reject) {
    log({ text: 'I\'m working here!' });

    setTimeout(function () {
      let isOdd = Date.now() % 2 === 1;
      if (isOdd) {
        resolve({ text: 'We are all going to die.' });
      } else {
        let error = new Error('I cannot work anymore. I am dead.');
        reject(error);
      }
    }, 250);
  });

  return p;
};
```

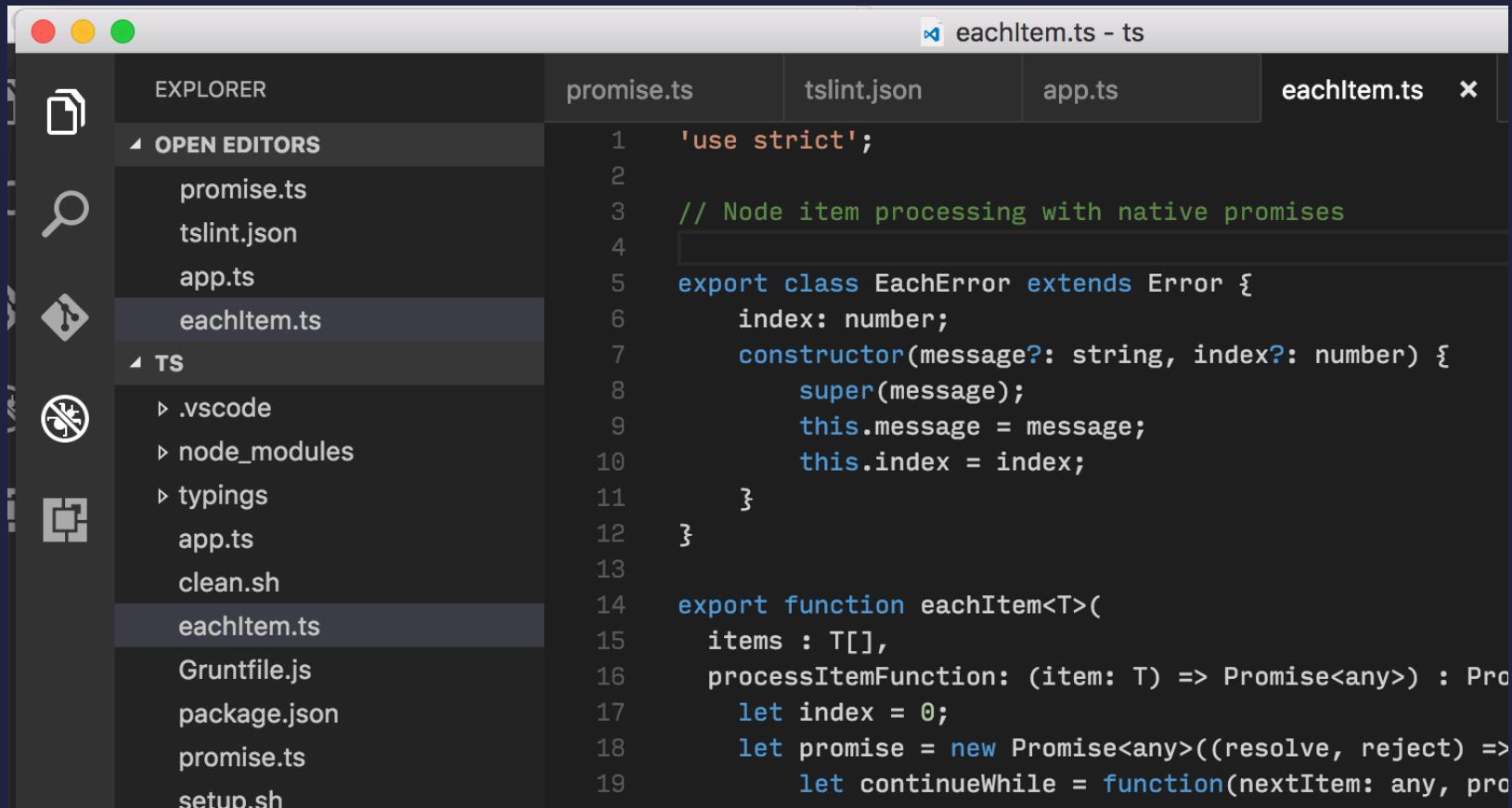
# CREATED BY ANDERS HEJLSBERG

- ▶ Delphi
- ▶ C#
- ▶ TypeScript



# WHICH IDE CAN I USE?

- ▶ **VSCode (now v1.4.0)**
- ▶ **Built on Atom/Electron**
- ▶ **Helpful IDE features**
- ▶ **Extensions Gallery**
- ▶ **Task manager integration**
- ▶ **Visit [code.visualstudio.com](http://code.visualstudio.com)**



A screenshot of the Visual Studio Code (VSCode) interface. The Explorer sidebar on the left shows files like promise.ts, tslint.json, app.ts, and eachItem.ts. The main editor area on the right displays a TypeScript file named 'eachItem.ts' with the following code:

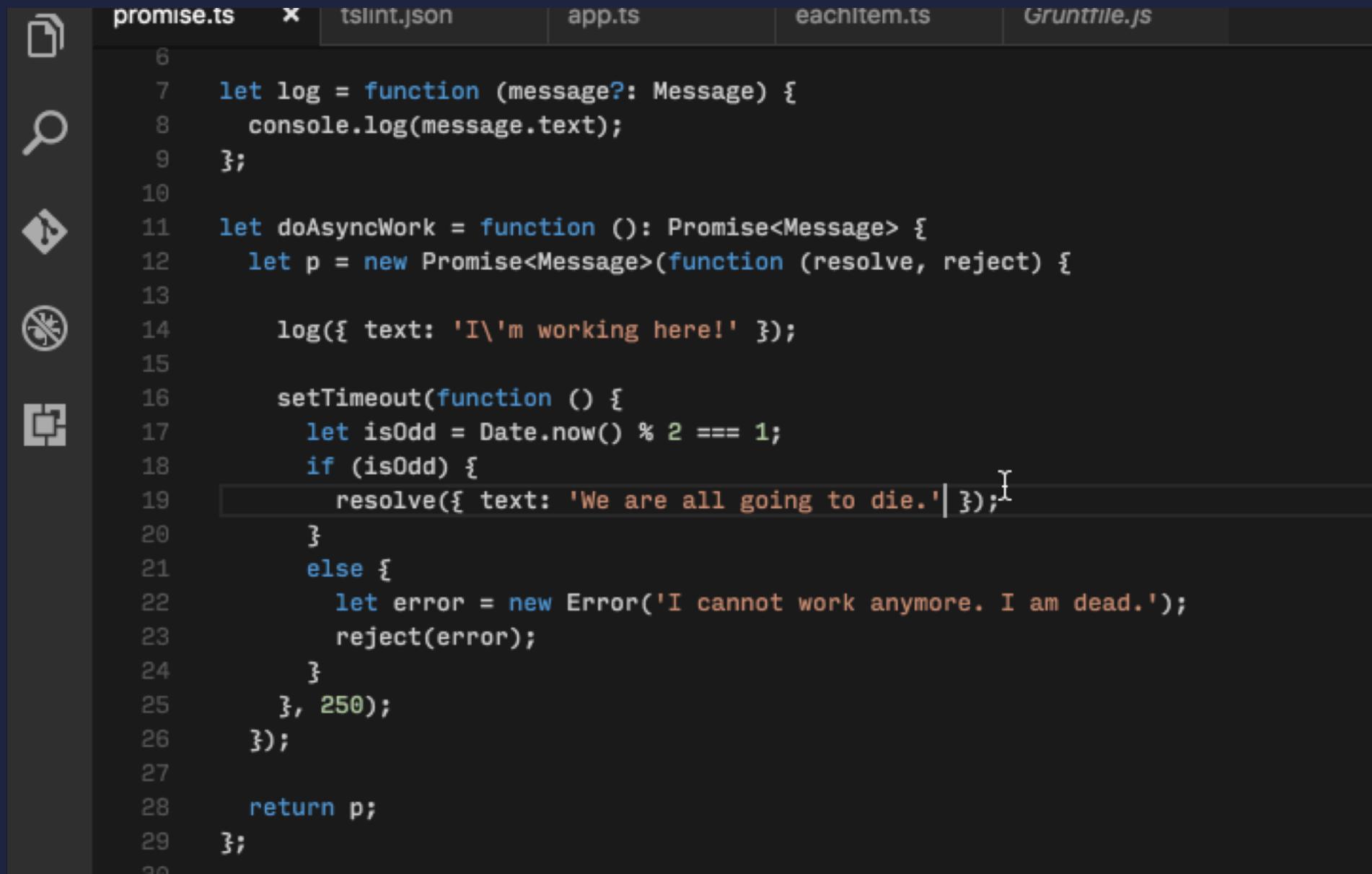
```
'use strict';

// Node item processing with native promises

export class EachError extends Error {
    index: number;
    constructor(message?: string, index?: number) {
        super(message);
        this.message = message;
        this.index = index;
    }
}

export function eachItem<T>(
    items : T[],
    processItemFunction: (item: T) => Promise<any>) : Promise<any> {
    let index = 0;
    let promise = new Promise<any>((resolve, reject) =>
        let continueWhile = function(nextItem: any, pro
```

# TYPE AND SYNTAX SUPPORT



A screenshot of a code editor interface, likely Visual Studio Code, demonstrating TypeScript syntax support. The editor shows several files in the title bar: promise.ts, tslint.json, app.ts, eachitem.ts, and Gruntfile.js. The promise.ts file is the active tab, displaying the following code:

```
6
7  let log = function (message?: Message) {
8      console.log(message.text);
9  };
10
11 let doAsyncWork = function (): Promise<Message> {
12     let p = new Promise<Message>(function (resolve, reject) {
13
14         log({ text: 'I\'m working here!' });
15
16         setTimeout(function () {
17             let isOdd = Date.now() % 2 === 1;
18             if (isOdd) {
19                 resolve({ text: 'We are all going to die.' });
20             }
21             else {
22                 let error = new Error('I cannot work anymore. I am dead.');
23                 reject(error);
24             }
25         }, 250);
26     });
27
28     return p;
29 }
```

The code editor features a dark theme with light-colored syntax highlighting. A tooltip or status bar at the bottom right indicates "TypeScript 2.8.3". The left sidebar contains icons for file operations like open, save, and search.

# GOOD BUT NOT PERFECT 😎

- ▶ Submit bugs as issues on GitHub
- ▶ Comment on current issues
- ▶ Provide details when you can
- ▶ The team is very responsive
- ▶ Updates are regular

# EXTENSIONS FOR VS CODE

- ▶ Install extensions easily
- ▶ Stay up to date
- ▶ Create your own extensions
- ▶ See Extension Gallery

**JavaScript Snippet Pack** 0.1.5 ⚡ 6K ★ 5  
A snippet pack to make you more produc...  
Mahmoud Ali [Install](#)

**language-vscode-javascript-angular2**  
Visual Studio Code language extension f...  
nwallace [Install](#)

**JavaScript Standard Format** 0.0.10  
Converts your code into Standard JavaSc..  
Sam Chen [Install](#)

**Wymsee's Angular 1 JavaScript and Typ...**  
Angular 1 JavaScript and TypeScript snip...  
Wymsee [Install](#)

**JavaScript Atom Grammar** 0.1.7 ⚡ 2K ★ 5  
The Atom editor's JavaScript text mate g...  
Microsoft [Install](#)

**JavaScript Unit Test snippets** 1.0.0  
VS Code JavaScript Unit Test snippets  
iZDT [Install](#)

# HIGHLY CONFIGURABLE

- ▶ `.vscode/launch.json`
- ▶ `.vscode/settings.json`
- ▶ `.vscode/tasks.json`

```
2  {
3      "editor.fontFamily": "Input Mono",
4      "editor.tabSize": 2,
5      "files.autoSave": "afterDelay",
6      "files.autoSaveDelay": 2000,
7      "editor.autoClosingBrackets": false,
8      "editor.quickSuggestions": true,
9      "editor.quickSuggestionsDelay": 250,
10     "files.exclude": {
11         "**/.git": true,
12         "**/.DS_Store": true,
13         "**/*.js": {
14             "when": "$(basename).ts"
15         },
16         "**/*.js.map": true
17     }
18 }
```

# HIDING JAVASCRIPT

- ▶ Show TypeScript
- ▶ Hide JavaScript

```
1 // Place your settings in this file to overwrite default settings
2 {
3     "editor.fontFamily": "Input Mono",
4     "editor.tabSize": 2,
5     "files.autoSave": "afterDelay",
6     "files.autoSaveDelay": 2000,
7     "editor.autoClosingBrackets": false,
8     "editor.quickSuggestions": true,
9     "editor.quickSuggestionsDelay": 250,
10    "files.exclude": {
11        "**/.git": true,
12        "**/.DS_Store": true,
13        "**/*.js": {
14            "when": "$(basename).ts"
15        },
16        "**/*.js.map": true
17    }
18 }
```

# DEBUGGING TYPESCRIPT

- ▶ Set Breakpoints in TS
- ▶ Use SourceMaps

```
"configurations": [  
  {  
    "name": "Run",  
    "type": "node",  
    "request": "launch",  
    "program": "${workspaceRoot}/app.js",  
    "stopOnEntry": false,  
    "args": [],  
    "cwd": "${workspaceRoot}",  
    "preLaunchTask": null,  
    "runtimeExecutable": null,  
    "runtimeArgs": [  
      "--nolazy"  
    ],  
    "env": {  
      "NODE_ENV": "development",  
      "PORT": "3010"  
    },  
    "externalConsole": false,  
    "sourceMaps": true,  
    "outDir": "${workspaceRoot}"/"  
  },  
  {
```

# DEMO!



# HOW CAN I AUTOMATE THIS TRANSPILING?

- ▶ Watch `tsc -w`
- ▶ Grunt or Gulp
- ▶ Configure Tasks in VSCode

```
/**  
 * Compile TS  
 */  
ts: {  
    default : {  
        src: [  
            '**/*.ts',  
            '!node_modules/**/*.ts',  
        ],  
        options: compilerOptions  
    }  
},
```

# HOW CAN I ENSURE MY CODING STANDARDS?

- ▶ **tslint.json**
- ▶ **Like jslint/jshint**

```
"max-line-length": [true, 120],  
"member-ordering": [true, "public-before-private"],  
"no-arg": true,  
"no-console": [false, "log", "trace", "error"],  
"no-duplicate-key": true,  
"no-duplicate-variable": true,  
"no-empty": true,  
"no-eval": true,  
"no-unreachable": true,  
"no-unused-expression": true,  
"no-unused-variable": true,  
"no-use-before-declare": true,  
"no-var-keyword": true,  
"label-undefined": true,  
"no-construct": true,  
"quotemark": [true, "single", "avoid-escape"],  
"rule": {}
```



**THANK YOU!**



# QUESTIONS?

## BRENNAN STEHLING

@BRENNANSV

[GITHUB.COM/BRENNANMKE](https://github.com/BRENNANMKE)

[GIST.GITHUB.COM/BRENNANMKE](https://gist.github.com/BRENNANMKE)