# The VS Code Roadmap 2017

With a very successful 2016 behind us, now is the good time to look towards the future. We typically look out 6 to 12 months, establish a set of themes we want to work towards, and then schedule work each milestone supporting those themes. When planning, we typically look at potential work from three perspectives:

- **Happy Coding:** At its core VS Code is a lightweight, keyboard focused, multi-language code editor. Features such as tabs, the task system, and the integrated terminal are all core experiences every developer uses on a daily basis. VS Code is already pleasant to code with, and we want to make the experience even more pleasant.
- **Node, JavaScript, and TypeScript:** On top of the core editor, we want to enable great end-to-end experiences for individual languages and frameworks. We build the Node, JavaScript, and TypeScript end-to-end experiences on top of the core platform and want to provide a great experience out of the box.
- **A Rich Extension Ecosystem:** Because we cannot deliver an experience for every language, VS Code has a rich extensibility system which enables an ecosystem of great end-to-end experiences for the breadth of modern languages and frameworks such as C#, Go, Python, C++, and more. In fact, we deliver the Node, TypeScript, and JavaScript experience using the exact same public APIs ensuring that developers have consistent experiences across the breadth of languages they use.

VS Code will continue to ship monthly, and we'll make progress against each of the following themes during each iteration. We describe some initiatives as "investigations" which simply means our goal in the next few months is to better understand the problem and potential solutions before committing to real feature work. Once an investigation is done, we will update our plan, either deferring the initiative or committing to it.

Legend of annotations:

| Mark | Description |
| --- | --- |
| • [ ] | work not started |
| • [x] | work completed |
| :runner: | on-going work |
| :muscle: | stretch goal |

## Happy Coding with VS Code

Over the past year, we have worked hard to eliminate core adoption blockers - the addition of Tabs and an Integrated Terminal are two good examples, but there is still work to do here. Customers often tell us that the lack of multiple root folder workspaces means that they can't use VS Code as their day-to-day editor. We will continue to address missing features, we will continue to make it easy for first time users to instantly be productive with VS Code, and we'll continue to add new experiences to make VS Code even more pleasant to use.

### Workbench

- [x] Support multiple root folder workspaces
- :runner: Support more flexible layouts, e.g., vertical panels
- Improve UI notification system
- [x] Improve key binding customization experience (hide keycodes, inconsistent command names from user)
- [x] Support theming of the workbench

- :runner: Import, export of settings (VS Code as well as other tools)

### Core Editor

- [x] Drag and drop
- [x] Copy and paste as formatted HTML
- [x] Minimap
- [x] Improve support for large files
- [x] Mouse map settings - support to customize commands bound to the mouse

### Terminal

- [x] Link detection
- Support splitting, viewing multiple terminals
- [x] Improve selection in the terminal

### Tasks

- [x] Task output in Terminal
- [x] Bind tasks to keyboard shortcuts
- [x] Simplify running and managing multiple tasks
- [x] Provide an API for TaskProviders so that extensions can contribute task detection for particular task runners

### Performance

- :runner: Monitor and improve startup time

### Source Code Control

- [x] SCM provider API
- [x] Git support as an extension (ships with VS Code)
- [x] Support partners building new SCM providers (e.g. Visual Studio Team Services)
- [x] Support for merging changes
- [x] Enable Explorer decorations for SCM status (e.g. colorize modified files)
- Support to view incoming changes

### Accessibility

- :runner: Improve usability of VS Code using NVDA on Windows
- Ensure all commands are accessible from the Command Palette
- Ensure keyboard accessibility in all dialogs
- [x] Narrator support (monaco-editor)

### Setup

- [x] Linux auto update
- [x] Windows 64bit support

## TypeScript, JavaScript, and Node Development

We want VS Code to be a great tool for developing modern MEAN/MERN applications with JavaScript and TypeScript. We will continue to collaborate deeply with the TypeScript team to deliver the richest code editing, navigation, and understanding experiences for both TypeScript and JavaScript. We will continue to make it easy to

configure debugging of your Node based applications while adding support for both client and server side debugging, asynchronous call stacks, and even live editing of your applications during a debug session.

### JavaScript and TypeScript

- Code actions and refactorings (partner with TypeScript)
- `tsconfig.json` and `jsconfig.json` IntelliSense (completions) and validation
- IntelliSense (completions) support for HTML embedded in JavaScript and TypeScript (Angular, React), partner with TypeScript
- Assist user entering JSDoc comments
- Improve JavaScript Linting/type checking
- TypeScript compile on save
- Enable semantic colorization (see Extensibility)
- :runner: Investigate improving support for popular JS technologies (webpack, Babel)

### HTML, SASS, CSS, LESS

- Color picker for CSS/LESS/SASS/HTML
- close end tag in HTML
- Investigate SASS/LESS/CSS multi-file support (understand imports)
- Investigate live preview of HTML

### Node

- Provide Node/MEAN Extension packs
- Update recommended Node extension triggers (e.g. on `package.json`, recommend Node extension pack)
- Include the npm extension functionality
- Node Debugging
  - Async stacks
  - Column break points
  - :muscle: Investigate profiling support
- Investigate a Mocha test runner
- Investigate MongoDB extension

## Extension creation, discovery, and management

Of course, VS Code is not just a Node, JavaScript, and TypeScript tool. Our rich extensibility model and extension ecosystem means that you can install support for just about every modern language and framework, from C++ to C# to Go, Python, and more. Looking ahead, we want to make acquiring extensions for these languages (and more!) as easy as possible. We want to enable extension authors to be able to be more productive and deliver richer experiences to developers. And at the same time, we want to give users more control over how those extensions contribute to their environment.

### For extension authors

- Investigate support for extensions that need to install additional tools on first run (e.g. .NET Core debugger, Go tools)
- :runner: Continue to improve and expand extension samples
- Add recently installed or recently updated extensions to the Welcome page, with links to open the extension's README

- Support extensions contributing an interactive playground
- Formalize HTML Preview API
- [x] Support explorer like contributions (refining the Custom explorer API exploration)

### For language extension authors

- Enable language services to contribute semantic coloring
- :runner: Evolve the Language Server Protocol

### Provide users with more control over extension contributions

- [x] Identify and resolve keybindings conflicts

### Contribute to Extensions

- :runner: GO
- :runner: Docker

## Summary

These are examples of just some of the work we will be focusing on in the next 6 to 12 months. We continuously tune the plan based on feedback and we will provide more detail in each of our monthly iteration plans. Please follow along and let us know what you think!