The VS Code Roadmap 2018

As 2017 comes to an end, now is the 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 look at potential work from three perspectives:

- **Happy Coding:** At its core VS Code is a lightweight, keyboard focused, multi-language code editor. VS Code is already pleasant to code with, and we want to make the experience even more pleasant, for both new and existing users.
- **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 build support for every language, VS Code has a rich extensibility system which enables an ecosystem of great end-to-end experiences for the breadth of languages and frameworks such as C#, Go, Python, C++, Java, and more. In fact, we deliver the Node, TypeScript, and JavaScript support 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 scheduling 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 6 months, we've worked hard to eliminate blockers that make it hard for people to adopt VS Code. For example, we make multi-root workspaces available in the October 2017 Stable builds. We continue to work with the extension authors to also enable their extension to support multi-root workspaces.

However, we're not done. You have asked for more flexibility in how you position editors and panes, for multi-selection in the explorer, and more. You will see a significant focus on the fundamentals in the next few months as well, focused on performance, localized language support, and accessibility so that every developer can be productive with VS Code.

Fundamentals

A key attribute of an 'editor' is performance and stability, if we add features at the expense of these fundamentals we risk losing a large part of what keeps us competitive in the 'editor' category.

- :runner: Improve startup performance: keep start-up times within a predictable and suitable range for users across all platforms
- :runner: Reduce resource/memory consumption: what can we do to reduce and control resource consumption of helper processes
- :runner: Continue to improve accessibility,
  - terminal
  - menu bar on Windows
- Support language packs for community-contributed translations
- Improve the Windows update experience
- Improve serviceability by providing additional diagnostics

Workbench

- Improve intra-file navigation
  - outline
  - bread crumb
- Support to layout the editor area as a grid
- Investigate improved settings discovery and editing
- Finish Multi-root folder Workspaces support and support extension authors in adopting it
- Show SCM status in the explorer (including `.gitignored` files)
- Support vertical panel layout
- Support multi-selection in the explorer for common actions e.g. delete
- Improve notification UI
- Support diagnostics with multiple error locations

Editor

- Improve the performance and scalability of editor decorations
- Improve Text Storage implementation
- Investigate into improved column selection
- :muscle: Investigate into semantic coloring support
- :muscle: Render more than text in the minimap

Terminal

- Support splitting and viewing of multiple terminals

Source Control Integration

- Support to view changes directly inside the editor using a peek/inline experience
- :muscle: Investigate integrated history view
- :runner: Investigate support to better collaborate on pull requests

WSL Support

- :runner: Improve the WSL support. Investigate how we can enable extensions to leverage tools available in WSL.

Node, JavaScript, and TypeScript Development

We want VS Code to be a great tool for developing modern web applications with JavaScript and TypeScript and 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 and include support for both client and server side debugging in the box.

Language Server Protocol

- :runner: Continue to refine and improve to Language Server Protocol with support from the community.
- ☑ Enable proposed protocol additions from the community.

Debug Adaptor Protocol

- :runner: Continue to refine and improve to Debug Adapter Protocol with support from the community.
- ☑ Expose more UI for DAP features that are currently not surfaced in the VS Code debugging UI. This includes moving the loaded scripts UI into the core.
- ☑ Move the Debug Adapter Protocol into a separate repository and provide a web site.

JavaScript

- ☑ Improve go-to-definition/implementation of a JavaScript symbol when a type cannot be precisely resolved

TypeScript (and JavaScript)

We closely collaborate with TypeScript, see also the TypeScript roadmap

- ☑ Organize imports, remove unused imports
- ☑ Tag completion in `jsx` and `tsx` files
- ☑ Add more refactorings
- :muscle: Investigate into improving TypeScript Source Maps so that they are more precise and includes variable mappings.

Webpack

- :muscle: Linting/validation of configuration
- :muscle: Make it easier to configure `jsconfig.json` / `tsconfig.json` for webpack

Debug

- :muscle: Improve hovering and inline values by leveraging language knowledge
- :runner: Continue to invest in documenting debugging recipes for common configurations
- ☑ Support hot code replace for non-JS scenarios e.g. Java.
- ☑ Support LogPoints

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 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. At the same time, we want to give users more control over how those extensions contribute to their environment.

For extension users

- [x] Improve extension recommendation system
- [x] Improve searching for extensions
- [x] Simplify tracking down issues caused by installed extensions and make it easier to file issues
- [x] Show the user more information about the usage of a extension (startup time, error rate, etc.).

For extension authors

- [x] Improve the language API to support a hierarchical navigation to the symbols in a document
- :runner: Make extension contributions attributable to an extension (e.g. error messages, commands)
- [x] Support migrating the publisher of an extension to another publisher
- [x] Investigate contributing explorer viewlets as a first class element in the activity bar
- [x] Investigate Quick pick improvements (e.g. multi-select, multi-step, commands, grouping)
- [x] Continue to invest into API that enables remote development
- [x] Add API for contributing decorations on resources shown in the explorer (like the Source Control decorations)

Documentation

- [x] Make it easier for extension authors to find their way around. Improve our API documentation, and integrate samples and documentation more closely.
- :runner: Refresh all of our dated overview videos.

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!