The VS Code Roadmap 2019 - UPDATED

As 2018 has come to an end, now is the time to look towards the future. We typically look out 6 to 12 months and establish topics we want to work on.

As we go we learn and our assessment of some of the topics listed changes. Thus, we may add or drop topics as we go.

We describe some initiatives as "investigations" which means our goal in the next few months is to better understand the problem and potential solutions before scheduling actual feature work. Once an investigation is done, we will update our plan, either deferring the initiative or committing to it.

As always, we will listen to your feedback and adapt our plans if needed.

Legend of annotations:

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

Themes

Our roadmap covers the following themes:

- Become the best editor out there for anyone who relies on accessibility features.
- performance, scalability, serviceability, security
- tackle some of the most wanted user features
- polishing and a constant slow trickle of design refreshments
- incrementally improve already existing features
- responsibly enable extensions that have broader extensibility requirements

Fundamentals

- :runner: Make VS Code an outstandingly accessible developer tool. We'll engage and work with our community to get input and guidance, and we need you to keep us honest.

- :runner: Keep start-up times within a predictable and suitable range for users across all platforms and improve the overall performance for large workspace:

    - Load less code on start-up and investigate improving the workbench restoration time by expanding on the rapid render approach.
    - [x] Implement a new tree widget that scales better
    - Adopt the new tree widget across the workbench:
        - [ ] quick pick
        - [x] comments
        - [x] explorer
        - [x] search

        - [x] settings
        - [x] outline
        - [x] debugger

- Improve serviceability
    - [x] Make it easy to identify extensions that negatively impact the overall performance of VS Code.

Workbench

- Workbench layout
    - Support for detachable workbench parts is our most upvoted feature request which due to architectural issues is challenging to implement. We will explore how we can work around this limitation. This investigation will focus on detaching terminals (2nd most upvoted feature request) and editors.
    - :runner: Enable a more flexible panel/sidebar layout.
- Improve working with the file explorer
    - [x] Support to compress/flatten single child directories
    - Investigate 'working sets' of files and folders
- Investigate how to safely provide richer customizability in the workbench
    - :runner: Support custom editors.
    - Investigate custom views (based on `WebView`).
- :runer: Support synchronizing settings and extensions across VS Code installations on different machines.
- [x] Provide filtering and fast keyboard navigation in trees across the workbench.

Search

- :runner: Show search results not only in the side bar or a panel, but also in an editor. This allows us to show additional context information for each match.

UX

- Continue to incrementally improve presentation and behavior across the board. Examples include:
    - :runner: Harmonize hovers, completion items, completion item details
    - [x] Iconography
    - Welcome page
    - :runner: Use tabs instead of the terminal dropdown
- Explore how to integrate fluent design on Windows

Editor

- Investigate isolating the editor from misbehaving grammars
- :runner: Investigate support for semantic coloring
- Investigate how to simplify the maintenance of textmate grammars
- :runner: Provide a richer minimap
- [x] Bring back localization support in the standalone Monaco editor. This support had been suspended when we added support for language packs for VS Code.

Languages

- Improve 'Expand Selection' to better adhere to the semantics of programming languages.
- Improve support for navigating and presenting complex error descriptions such as those generated by TypeScript for React or Vue.
- :runner: Enable programming language extensions to provide support for call hierarchies.
- Enable programming language extensions to provide support for type hierarchies.

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. see also the TypeScript roadmap.

- Improve the integration of tslint by running it as a TypeScript Server plugin.

Debug

- Support data breakpoints
- Improve hovering and inline values by leveraging the knowledge about the programming language so that the `Inline Values` feature can be enabled by default
- Continue to invest in documenting debugging recipes for common configurations

Extensions

- Extension Acquisition
  - Revisit how users find and install extensions
  - Improve the recommendation system.
  - Add support to only activate signed extensions (see next section).
  - Support installing an extension without having to reload the workbench. This is our 3rd most upvoted feature request.
- Extension Management
  - Make the consumption of extensions more secure and improve the process for how we handle malicious extensions.
  - Show runtime information for an extension (activation state, performance, error logs)
- Extension Publishing
  - :runner: Collaborate with extension authors to improve their extensions. Examples are: Use Webpack to improve install and activation, minimize dependencies of an extension, ensure `vscode` is only a development dependency.
  - :runner: Enable extensions to install additional platform specific components.
  - Support publishing of signed extensions.
  - Add support for verified publishers.

Contributions to VS Code Extensions

Our teams contributes to a number of extensions that are available in the market place.

Our main focus will be on the following extensions:

- :runner: VS Code Remote SSH
- :runner: VS Code Remote WSL
- :runner: VS Code Remote Containers

- :runner: GitHub Pull Request extension
- :runner: Azure Account extension
- :runner: Vue extension

We will continue to maintain the following extensions:

- :runner: Chrome Debug extension
- :runner: ES Lint
- :runner: TS Lint
- :runner: Ruby
- :runner: Markdown customization extensions

We will investigate into improving the performance of existing popular extensions that make the extension host unresponsive, e.g.

- Bracket Pair Colorizer
- [x] Auto Rename Tag.

Contributions to Underlying Components and Technologies

VS Code is made possible through a wide range of technologies. Below are examples of technologies in which we are particularly active.

Language Server Protocol

- :runner: Continue to refine and improve the Language Server Protocol with support from the community.
- [x] Define a Language Server Index Format (LSIF, pronounce like "else if") that enables a language server to persist their language intelligence, so that it can be subsequently used to answer LSP requests at-scale (for example, hover and go to definition).

Debug Adaptor Protocol

- :runner: Continue to refine and improve the Debug Adapter Protocol with support from the community.
- :runner: 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.

xterm.js

- [x] Work with the `xterm.js` community to improve parsing and internal line representations
- [x] Adopt `conpty` on Windows
- [x] Reflow lines when resizing the terminal
- [x] Investigate replacing canvas based rendering through WebGL based rendering

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.

Community Support

- We'll continue our work on our GitHub bots that enable members of our community give us a hand triaging and resolving issues.

Summary

These are examples of 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!