The Visual Studio Code Roadmap 2020

As 2019 is coming to an end, now is the time to look towards the future. We typically look out 12-18 months and we establish topics we want to work on. We don't start with our roadmap on a blank sheet. We develop it based on our last roadmap, the findings we made over the course of the last year, and of course what we heard from you in issues, in face-to-face discussions, stack overflow, and twitter.

When we execute on our roadmap, we keep learning and our assessment of some of the topics listed changes. As a result, we may add or drop topics as we go. In 12 months from now, we then come together to develop the next roadmap.

We describe some initiatives as "investigations" or "explorations" 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 listen to your feedback and adapt our plans if needed.

Legend of annotations:

Mark	Description
bullet	work not started
check mark	work completed
:runner:	on-going work
:muscle:	stretch goal
:red_circle:	missing link

Themes

Our roadmap covers the broadly the following themes:

- Become the best editor for anyone who relies on accessibility features
- Improve performance, scalability, and security of VS Code and its extensions
- Tackle some of the most wanted and most emotional user features
- Polishing and a constant trickle of design refreshments
- Incrementally improve already existing features
- Responsibly enable extensions that have broader extensibility requirements
- Tackle a couple of big rocks that push the boundaries of what VS Code can do

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.
- [ ] Improve support for RTL languages.
- [ ] :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] Continuously monitor the size of the VS Code download to ensure it downloads in seconds no matter where you are.
- [ ] Improve VS Code performance in large workspaces
    - [ ] :runner: Investigate 'working sets' of files and folders in the file explorer
    - [ ] Provide guidance to users how to configure VS Code
    - [ ] Improve Open File / Quick Pick performance
- [ ] :runner: Make VS Code more secure by running renderer processes **without** access to `node` APIs.

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.
    - [x] Allows to move views between the panel and sidebar.
    - [ ] Support a more flexible workbench layout such as allow sidebars on the left and the right. For example, you could have your outline on the right and the file explorer on the left.
- [ ] Investigate how to safely provide richer customizability in the workbench
    - [x] Support custom editors.
    - [x] Investigate custom views based on `WebViews` .
- [ ] Support to configure workbench font and font size
- [ ] Broaden support to customize the UI, e.g. menu bar, context menus.

Settings

- [x] Support synchronizing settings and extensions across VS Code installations on different machines.
- [ ] :runner: In the settings editor, provide high-fidelity support for additional settings types, such as colors.
- [ ] Provide a dedicated theme customization editor.
- [x] Support to synchronize settings between the insider and stable versions of VS Code

Search

- [x] Show search results not only in the side bar or a panel, but also in an editor. This allows users to work with multiple search results and to see additional context for each match.

UX

- [ ] Continue to incrementally improve presentation and behavior across the board. Examples include:
    - [ ] :runner: Harmonize hovers, completion items, completion item details
    - [ ] :runner: Welcome page
    - [ ] Use tabs instead of the terminal dropdown
- [ ] :runner: Revisit/review the first run experience for VS Code as well as newly installed extensions. For example, we want to reduce the number of notifications that are shown.
- [ ] :runner: Touch and mobile support for VS Code in the browser

- [ ] Investigate how users can express what they want to use VS Code for after they installed it. For example, a user wants to develop a website with a Go backend, VS Code should provide light-weight guidance to get all the right extensions and configure them appropriately.

Editor

- [ ] :runner: Notebooks (for example Jupyter) have become a popular means in data science and education and are making their way into every corner of development. We'll explore how VS Code and the Monaco Editor can provide a rich notebook experience (deep language services, light-weight/simple debugging).
- [ ] Investigate isolating the editor from misbehaving grammars
- [x] Investigate support for semantic coloring
- [ ] Investigate how to simplify the maintenance of textmate grammars
- [ ] Expand 'inline' experiences
    - [ ] Inset editing
    - [ ] Inline errors
    - [ ] :runner: Richer minimap
    - [ ] :runner: Inline values (on by default, see debugging)
- [ ] Smarter indent support (e.g. for Python)
- [ ] Prevent tooltips from getting into your way when editing
- [x] Support word-wrap in the diff editor

Languages

- [x] Enable programming language extensions to provide support for call hierarchies.
- [x] Enable programming language extensions to provide support for type hierarchies.
- [ ] Improve support for workspace-level edits (an semantic unit comprised of name, location, and content manipulations across multiple files)
    - [ ] :runner: Enable language extensions to participate in workspace edits such as rename or move.
    - [x] Enable an one-step undo of complex workspace edits.
    - [x] Investigate how a user can preview workspace edits (e.g. preview the changes of a rename refactoring).
- [ ] Incrementally improve support for nested languages. This will be code changes on our side as well as samples and documentations.

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. One focus item will be to show the errors and warnings for the entire project(s) in your workspace.

HTML/CSS

- [ ] Provide a live preview.

SCM

- [ ] :runner: Investigate synergies between the core git support and GitLens

- [x] Provide an overall improved GitHub integration, e.g. we'll make it easier to hover over issue URL in your code or associate a GitHub issue with your commit.
- [ ] Provide full merge support (3-way)
- [ ] Investigate support for showing a local history of a file and folder and combine that with the git history
- [x] Provide a contributable timeline view
  - [ ] Investigate combining local history with other events such as successful test runs or refactorings

Debug

- [ ] :runner: Leverage language services to improve debug experience. For example, improve hovering and inline values by leveraging the knowledge about the programming language so that the `Inline Values` feature can be enabled by default
- [x] Provide improved 'full stack' debugging, i.e. combined `node` (server) and `chrome` (client) debugging.

Testing

- [ ] :runner: Investigate how VS Code can improve the testing support. Several extensions are already providing testing support, explore what APIs/UIs could be added to improve these testing extensions and the test running experience.

Install

- [x] Investigate migrating to the MSIX installer on Windows.

Terminal

- [x] Investigate how to persist integrated terminal sessions across window reloads and, for remote scenarios, application restarts.
- [ ] Investigate how to support problem matchers to run in any internal terminals

Extensions

- Extension Acquisition

  - [ ] Revisit how users find and install extensions including a simple text-based approach
  - [x] Make it easier for users to pick and choose which functionality they want from extensions packs.
  - [ ] Improve the recommendation system, e.g., allow users to ask for recommendations based on their prior use of VS Code.
  - [ ] Add support to only activate signed extensions (see *Extension Authoring*).

- 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 Authoring

    - [ ] :runner: Provide clear guidance to extension authors for how to structure their functionality so that their users can get as little or as much functionality as they want. For example, ensure text editor lovers can get Python language support without too much overhead.
    - [ ] :runner: Provide an integrated 'Welcome' experience for extensions.
    - [ ] :runner: Investigate how to effectively inform users about new features provided by an extension update.
    - [ ] :runner: Investigate into play grounds like the VS Code playground (`Help>Interactive Playground`) for extensions.
    - [ ] Provide extensions more control over the sequence and grouping of their settings in the settings editor, e.g., support to group *commonly used* settings.
    - [ ] Give extensions more realtime insights into how user configured them, e.g. the user hides the custom tree view your extension contributes from the explorer.
- Extension Publishing
    - [ ] :hand: Work with the marketplace to allow platform specific flavors of extensions.
    - [ ] :hand: Support publishing of signed extensions.
    - [ ] :hand: Add support for verified publishers.
    - [ ] :hand: Investigate providing a GitHub action for publishing to the marketplace.

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
- [ ] :runner: GitHub Pull Request extension and related extensions
- [ ] :runner: Azure Account extension
- [ ] :runner: Vue extension
- [ ] :runner: GitLens extension

We will continue to maintain the following extensions:

- [ ] :runner: ES Lint
- [ ] :runner: Ruby
- [ ] :runner: Markdown customization extensions

We will investigate into improving the performance of popular extensions that not performing as well as users expect

- [ ] Bracket Pair Colorizer
- [ ] VIM

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.
- [ ] :runner: Continue to refine and improve the Language Server Index Format with support from the community.

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

- [ ] :runner: Provide ongoing improvements to `xterm.js` for performance, stability, and maintainability

Engineering

- [x] Migrate from `tslint` to `eslint`. Tslint will be deprecated.
- [ ] Improve the built-in issue reporter, e.g., support to paste/attach images.
- [x] Implement tooling that allows us to detect issues in which emotions fly high.
- [ ] :runner: Improve our "smoke" tests and revisit the current approach on how we implement them.
- [ ] Investigate automating how we sanity check all our binaries right before we release.
- [ ] :runner: Investigate moving our GitHub bots to GitHub actions.
- [ ] :runner: Improve our GitHub issue bots, examples:
  - [ ] reject invalid incoming issues automatically
  - [ ] :runner: make it easier to move issue to extension repositories
  - [ ] :runner: automate training of our classification bot

Explorations

- [ ] Explore support to synchronize more state when opening the same folder in different VS Code installs such as untitled files, open files, focussed editor.
- [ ] Explore how to better provide support for low-end devices.
- [ ] :runner: Explore how to support new user in learning VS Code (interactive exploration mode, videos)
- [ ] Explore how to make it easier to show, tell, and share your VS Code customizations.
- [ ] :runner: Explore how to support trying out VS Code for a particular language or technology stack without any local installation. For example, 'Debug Java code in less than 1 minute'.

Documentation

- [ ] :runner: Refresh all of our dated overview videos.
- [ ] Provide translated documentation for major user groups such as Mandarin, Korean, Spanish, Portuguese, French. Noe: the list of languages might change.

Community Support

- [ ] :runner: 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 12-18 months. We continuously tune the plan based on feedback and we will provide more detail in each of our monthly iteration plans. We will develop our next roadmap in around 12 months from now. Please follow along and let us know what you think!