The Visual Studio Code Roadmap 2021-2022

Our roadmap typically looks 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. After around 12 months we 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 plans, 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

Values

Before we go into the details, let's start with our values that guide the development of VS Code. They have pretty much remained the same since the early days and we don't have any intention to change them:

- Out of the box, VS Code is a text editor: lightweight and fast. Users may install extensions that impact the appearance and nature of VS Code.
- Performance trumps functionality.
- Simple, uniform UI and UI interactions.
- Allow for CLI / UI co-existence and interplay whenever possible.
- Although we limit the UI choices extension authors can make, authors can write powerful extensions such as one that turn VS Code into an IDE with bells and whistles.

Themes for 2021

For 2021, we'll particularly focus in the following themes.

- Continue on our journey to be the best editor for anyone who relies on accessibility features
- Introduce support for Profiles
- Introduce Trusted Workspaces, and explore signed and sandboxed extensions
- Enable extension authors to provide great notebook experiences or contribute custom output renderers to existing notebook experiences
- Update the infrastructure of our website; start the process of visual updates
- Support the official launch of GitHub Codespaces.

Accessibility and Internationalization

- [ ] :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.
- [ ] Move off our custom IME story and use `ContentEditable` to improve text segmentation for RTL and CJK.

Performance

- [ ] :runner: Keep start-up times within a predictable and suitable range for users across all platforms.
- [ ] Improve VS Code performance in large workspaces
  - [ ] :runner: Investigate 'working sets' of files and folders in the file explorer
  - [ ] Improve Open File / Quick Pick performance
- [x] Ensure VS Code loads instantly in the web.

Security

- [x] Trusted workspaces: Ensure it's safe to clone a repo and read the source code without any unauthorized code execution.
- [ ] :runner: Make VS Code more secure by running renderer processes **without** access to `node` APIs. This is also crucial of the longevity of our platform as it brings our application model closer to the pure web model.
  - [x] Move from `webviews` to `iFrames` (we'll continue to refer to the custom views and editors as being webview-based)
- [ ] Add support to allow users to enforce that only select signed extensions are activated / or can be installed.
- [ ] :runner: Ensure all extension code can be signed (needs platform specific extensions)
- [ ] Explore sandboxing of extensions in both local and remote scenarios

Codespace and Remote

- [ ] Simplify to move from a local folder to a container to a codespace or the other way around
- [ ] Investigate client-side caching of file resources (web and desktop) to make working with unreliable connection less dreadful
- [ ] Remote-SSH
  - [ ] Prevent server installation on micro-instances
  - [x] Support 'Reopen in Container' leveraging Remote-Container support
- [ ] Remote-Container
  - [ ] :runner: Simplify creating a custom dev container configuration, e.g., 'a la cart consumption of the features in the universal image'
  - [x] Assist users in installing `Docker` when not yet installed
  - [x] Support for untrusted remote-containers
- See also *Support browsing and modifying GH repositories* in the SCM section.

VS Code Core

General

- [ ] Review the default values for all settings we ship out-of-the-box

UX

- [ ] :runner: Investigate how users can express what they want to use VS Code for after they installed it and provide help with setting up their environment: runtime, extensions, settings.
- [x] Rethink: "New Untitled File" is a relict from a time when we only supported text files - how much can we do by automatic detection, how much user input do we need
- [ ] Refresh the themes we ship out of the box
- [ ] Continue to incrementally improve presentation and behavior across the board. Examples include:
  - [x] Welcome page
  - [ ] :runner: Harmonize hovers, completion items, completion item details
  - [ ] Command palette touch-ups
- [ ] :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 and allow extensions to integrate their welcome screens into the workbench's welcome page that are shown.
- [ ] :runner: Touch and mobile support.

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: 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.
- [ ] Support to configure the workbench (also see https://github.com/microsoft/vscode/issues/115641)
  - [ ] Workbench font and font size
  - [ ] Workbench display density (small versus large buttons, compact activity bar)
  - [ ] Broaden support to customize the UI, e.g. configure available actions in the menu bar, context menus, toolbars. Investigate Bartender like support.
  - [ ] Introduce profiles describing appearance (and behavior) or the workbench (and extensions); related to *Sets of Settings*, maybe it's the same, we don't know yet; Examples: Text editor profile, IDE profile, Student profile
- [ ] Explore to enable users (or the workbench on the user's behalf) to manage notifications
- [x] Improve handling opening a file when there are multiple editors available, for example, let the user choose on first open

API

- [ ] Clean-out or finalize APIs that have been in proposed mode for quite some time such as Search
- [ ] Improve custom editors and custom views
  - [x] Improve data transfer between webviews and the extension host
  - [ ] :runner: Support web-workers in webviews
  - [x] Support a predefined set of webcomponents in webviews

- [ ] :runner: Our API provides access to open text editor, but not notebooks, or custom editors. Extension authors rely on less than optimal workarounds.
- [ ] Enrich tree views to lower the need to implement a custom webview-based view (model-based search/filtering, inline editing, help text, large buttons, drag&drop, checkboxes)

Settings

- [ ] :runner: Provide high-fidelity support for additional settings types, such as colors.
- [ ] Provide a simplified theme customization support ("define 6 base colors")
- [x] Provide extensions more control over the sequence and grouping of their settings in the settings editor, e.g., support to group *commonly used* settings.
- [x] Support private workspace settings
- [ ] Support Sets of Settings/extensions users can switch between (e.g. note taking, java dev, react app); related to *Workspace Profiles* (maybe it's the same, we don't know yet)

Editor

- [ ] Expand 'inline' experiences
  - [x] Inline values (on by default, see debugging)
  - [x] Inline type hints (see language section)
- [ ] Spell-checking as a driver for improving how a user interacts with diagnostics. Spelling errors are frequent and frequently on the same line. Going through the quick-fix is not sufficiently fluent. Outlook, for example, uses a single-click.

Notebooks

- [ ] :runner: Achieve feature parity with regular text editors: minimap, scrollbar markers, find across inputs and output
- [ ] :runner: Public notebook API and allow dedicated notebook extensions in the VS Code marketplace; extension recommendations
- [ ] :runner: Extend language servers to natively support notebooks
- [x] Explore improved debugging of notebook cell execution
- [ ] Investigate compliance test suites for notebook providers, renderers
- [ ] :runner: See "Markdown" section for what may apply to markdown cells

SCM / Pull Requests

- [ ] :runner: Investigate synergies between the Core git, GHPRI, and GitLens (implementation sharing, plugin extensibility)
- [ ] :runner: Bring GitHub closer into the inner loop across VS Code Core, and the GHPRI and GitLens extensions
  - [x] Support browsing and modifying GH repositories without a need to clone the repository or open a codespace
  - [x] Assign from overview editor
  - [x] Support resolving conversations
  - [ ] Filter comments in comment view by resolved status
  - [ ] `@` and `#` completions everywhere including comments
- [ ] Improve the diff experience
  - [ ] Actions for copying changes back and forth in the diff editor gutter

    - [ ] Collapse unchanged lines

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
- [ ] :runner: Expose more UI for DAP features that are currently not surfaced in the VS Code debugging UI. This includes to support Memory and Disassembly views.

Debug Adapter Protocol (DAP)

- [ ] :runner: Continue to refine and improve the Debug Adapter Protocol with support from the community.
- [ ] :runner: Green threads DAP support

Testing

- [x] Provide testing support and help the ecosystem to move over.
- [ ] :runner: Ensure Testing support scales for mono repos
- [ ] Explore code coverage support

Terminal

- [x] Investigate how to persist local terminal sessions across window reloads and application restarts.
- [x] Investigate improved shell/profile selector (see Windows Terminal for inspiration)
- [x] Use tabs instead of the terminal dropdown
- [x] Investigate to "resume" a terminal by deserializing its state rather than replaying raw data events
- [x] Investigate to allow terminals in the editor area
- [ ] :runner: Provide ongoing improvements to `xterm.js` for performance, stability, and maintainability
- [ ] Explore intellisense-inspired CLI help support in the terminal

Extensions

- 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.
    - [ ] Extend our development tooling:
        - [x] Support for web worker extensions
        - [ ] :runner: Improve testing (unit testing and Playwright)
    - [ ] Extend our extension guidelines to not only cover UX, but also performance
    - [ ] Allow extension authors to integrate GH Sponsor information
    - [ ] Revisit the format of our extension author community call
- Extension Publishing (See *Security*)

    - [x] Support for platform specific flavors of extensions.
    - [ ] Support publishing of signed extensions.
    - [x] Add support for verified publishers.
    - [x] Add support for insider extensions

Serviceability

- [ ] Improve the built-in issue reporter
    - [ ] Support to paste/attach images
    - [ ] Allow extensions to contribute data to the reported issues (templates, extension logs)

Languages

- [ ] Language aware spellchecking in comments

TypeScript

We will continue to collaborate deeply with the TypeScript. See also the TypeScript roadmap.

- [ ] Adopt LSP for TS
- [x] Inline type hints
- [ ] :runner: Explore supporting multiple files/libraries in the Web Worker based language service
- [ ] Explore support for TS notebooks: This allows us to drive the LSP/notebook discussions.

HTML/CSS

- [x] :runner: Provide an integrated live preview.

Markdown

- [ ] Improve support for images (paste images, preview images on hover, etc.)
- [ ] Investigate integration of a spell-checking service (such as MS Editor)
- [ ] In-place toolbar for formatting markdown ("code actions on steroids")

Language Server Protocol and LSIF

- [ ] Make it easier for language server owners to support LSIF
- [ ] :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.
- [ ] :runner: Support type hierarchies in LSP

Contributions to VS Code Extensions

- [x] GitLens extension
- [ ] :runner: ES Lint
- [ ] :runner: Ruby
- [ ] :runner: Markdown customization extensions

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

- [x] Bracket Pair Colorizer
- [ ] :runner: VIM

Install / Update

- [x] Make VS Code available in the Windows Store.
- [ ] Provide a MSIX installer on Windows.
- [ ] Rework our update mechanism based on Chromium's OSS updater; this allows for more incremental updates which helps with slow connections and also makes universal downloads like the one for macOS more acceptable

Engineering

- [ ] :runner: Improve our smoke tests and revisit the current approach on how we implement them.
- [x] Invest into a unified, improved and fast file watching
- [ ] :runner: Improve our GitHub issue bots, examples:
  - [x] reject invalid incoming issues automatically
  - [ ] :runner: automate training of our classification bot
  - [x] detect issues of limited use that 'need more information' automatically
- [ ] :runner: Adopt a safe supply chain
- [x] Explore improving the build time by using esbuild (requires AMD support) - ensure we don't lose our ability to use with the latest TS RC

Website

- [ ] :runner: Refresh all of our dated overview videos.
- [ ] Modernize the website's architecture/implementation/technology stack
- [ ] Modernize the appearance of our website

Community Support and OSS Project

- [ ] :runner: We'll continue our work on our GitHub bots that enable members of our community give us a hand triaging and resolving issues.
- [ ] Ensure we have always good collection of issues labeled `help wanted` and `good first issue`

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!