The test plan for the December iteration plan December Iteration Plan.

Debug - long text wraps in repl

- ☒ win @alexandrudima
- ☒ mac @jrieken

Verify that long text now nicely wraps across multiple lines in the debug repl both for evaluation requests and for output text.

Debug - breakpoint state

- ☒ mac @jrieken
- ☒ linux @aeschli

We have changed breakpoints states and now show it differently in the UI. Verify: * new UI properly and intuitivly reflects if breakpoints are enabled, disabled, verified or deactivated. * breakpoints get hit / not hit depending on their enablement state

Debug - extension debugging

- ☒ win @dbaeumer
- ☒ linux @aeschli

Extension debugging is now using a new strategy to attach the debugger to the extension * verify you can debug your extension as before * verify you can restart debugging from the debugger side and the debugger reconnects properly * verify you can change folders or refresh on the extension development side and the debugger reconnects * verify closing the extension development side stops debugging * verify stopping the session from the debugger closes the extension development window

Update - channel from settings

- ☒ win @egamma
- ☒ mac @joaomoreno

Verify you can change the update channel from settings.

View - persisted zoom settings

- ☒ win @dbaeumer
- ☒ mac | linux @weinand

Verify you can change and persist the zoom factor for windows. * negative and positive values work * the zoom actions are still working as before but do not persist * the change is live

Quick open - Path and Fuzzy matching

- ☒ win @aeschli
- ☒ mac | linux @isidorn

Unless you include a path character in your query or enabled fuzzy matching, quick open for files should work as before: * results are sorted properly with most relevant ones to the top * editor history always comes first on top of matching file results * if you narrow down a search, subsequent matching is fast and does not go to the disk (verify with a larger workspace)

Verify the new support to match on paths if included in search * using the native path separator in the search enables matching on (workspace relative) paths for both editor history results and file matching * more relevant results are still sorted to the top * results are highlighted properly also in their path segment

Verify the new support for fuzzy file matching * find and enable the new setting to enable fuzzy matching for quick open. this will only enable fuzzy matching for file search, it should not have any impact to other quick open providers, nor the editor history search * verify results from file searches match on the entire (workspace relative) path of the file without having to use the native path separator * verify matching happens in substrings of the path, not requiring the characters to be in sequence (e.g. "fr" would match "**f**oo_ba**r**") * more relevant results are sorted to the top

Extensions info in status bar #1123

- ☐ win @bpasero
- ☐ mac @egamma (blocked by #1256)
- ☐ linux @SofianHn

Verify that the extensions status shows up in the status bar if you have extension errors. Verify clicking on the status bar shows all the error / warning messages to the user and that you can uninstall the extension via action in message.

Emmet support in JSX and TSX files

- ☒ win | mac | linux @joaomoreno

In both .tsx and jsx files the following snippet

```
.foo>input[name="bob"]+label[for="bob"]
```

Should expand to

```
<div className="foo">\n\t<input type="text" name="bob"/>\n\t<label htmlFor="bob"></label>\n<
```

ES6 by default

- ☒ win @dbaeumer
- ☒ linux | mac @joaomoreno

Make sure that ES6 is supported without the need to configure something. Test: * JS code inside html supports ES6, like `let`, `const`, and `class C {}` * Have no jsconfig file and make sure ES6 is supported * create a new jsconfig.json and make sure the default is ES6 for the target property

Language Server

☒ win | mac | linux @jrieken

The language client / server libraries now support the complete set of language features available in the extension host. Implement a sample language server with more features by taking https://github.com/Microsoft/vscode-languageserver-node-example as a starting point. Focus on the new support for CodeActions, CodeLens, Formatting and rename.

vscode-tslint

☒ win | mac | linux @bpasero

- added support for a configFile config setting, see the README.md (known limitations captured in Microsoft/vscode-tslint#16)
- schema support for the tslint.json configuration file

line/page scrolling

☒ win @aeschli

☒ mac @isidorn

☒ linux @aeschli

- not platform specific
- win/linux: Ctrl+Up, Ctrl+Down, Ctrl+PageUp, Ctrl+PageDown
- mac: Ctrl+PageUp, Ctrl+PageDown, Cmd+PageUp, Cmd+PageDown
- should now scroll the viewport by a line or a page.

select current line

☒ win | mac | linux @isidorn

- not platform specific
- win/linux: Ctrl+I
- mac: Cmd+I
- should expand selection to the current line and, if pressed multiple times, to the following lines.

- please check that it works on long lines where word wrapping kicks in or when using `wrappingColumn: 0`.

numpad keys

☒ win @bpasero

☒ mac @bpasero

☒ linux @bpasero

- **is platform specific**
- please check that you can add and trigger keybinding rules with the following keys:
- `numpad0`
- `numpad1`
- `numpad2`
- `numpad3`
- `numpad4`
- `numpad5`
- `numpad6`
- `numpad7`
- `numpad8`
- `numpad9`
- `numpad_multiply`
- `numpad_add`
- `numpad_subtract`
- `numpad_decimal`
- `numpad_divide`

find widget improvements

☒ win | mac | linux @weinand

☒ linux @aeschli

- not platform specific
- Ctrl+H (mac: Cmd+Alt+F) now always focuses the replace field in the find widget.

- Ctrl+F (mac: Cmd+F) now always focuses the find field in the find widget.
- when focus is in the find field or the replace field in the find widget, Ctrl+Down (mac: Cmd+Down) will focus the editor.
- Alt+C (mac: Cmd+Alt+C) now toggles case sensitive. This should work also when the find widget is hidden and should reflect in the selection highlight.
- Alt+W (mac: Cmd+Alt+W) now toggles match whole words. This should work also when the find widget is hidden and should reflect in the selection highlight.
- Alt+R (mac: Cmd+Alt+R) now toggles match regex. This should work also when the find widget is hidden and should reflect in the selection highlight.

extension deactivate()

☒ win @dbaeumer
☒ mac @joaomoreno
☒ linux @aeschli

- **is** platform specific
- the extension host now calls `deactivate()` and `dispose()` on subscriptions pushed in the `activate()` ctx
- please try to author an extension that:
- when activated, forks/spawns a process
- the forked/spawned process logs to a file if it is alive (with a timestamp)
- on `deactivate()` or through a subscription pushed in the `activate()` ctx,
- the extension sends the forked/spawned process a graceful shutdown request
- the forked/spawned process listens to this graceful shutdown request and gracefully shuts down
- verify that when closing VSCode the forked/spawned process shuts down gracefully.

keybindings with non-US standard kb layout

☐ win | mac | linux @bpasero (maybe someone else needs to try, I am running into: Keyboard: Widget not showing correct values #1275)

- **is** platform specific
- please see #713 for a detailed explanation
- **TL;DR**. We now have a widget to help with authoring keyboard rules and we now render UI labels taking the keyboard layout under consideration.
- **Setup**: please change to a keyboard layout different than US standard. Preferably, this is the keyboard layout that your physical keyboard has.

- The widget:
- it only shows when authoring the `keybindings.json` file
- it is invokable via Ctrl+K Ctrl+K (mac: Cmd+K Cmd+K)
- it is invokable via clicking on the "Define Keybinding" launcher
- it shows an input box that reflects **what VSCode sees when you press keys**
- it does not help with defining chords or seeing conflicts, etc, its intent is to help with different kb layouts.
- no keybindings are triggered while you try keybindings
- Escape or focus loss dismisses it
- Enter accepts it
- The UI labels:
- easiest to see them in the F1 list
- the keyboard layout is taken into account only once and then cached, so if you change keyboard layouts, you need to reload the window
- the idea is that the keys `VK_OEM_1`, ... etc are looked up and checked what they would produce if typed under the current system keyboard layout.
- the labels then render with the produced character.
- e.g. `Split Editor` shows under `German (Switzerland)` as Ctrl+ä because you trigger it by pressing the keys Ctrl and the key that would produce `ä`.
- dead characters: Sometimes, under some kb layouts, some keys don't produce anything and act more like composing (accents). To be able to render something, we display in the UI label the value or the shiftedValue of those keys.

Suggest widget #1006 #1079

☒ win @bpasero
☒ mac @egamma

The Suggest widget underwent quite some heavy rework behind the scenes. Make sure everything still works as before.

Scoped git services #718

☒ mac @isidorn
☒ linux @weinand

You should now be able to open a folder inside a git repo and still have the git features enabled.

Git sync & publish action #908

- ☒ win @dbaeumer
- ☒ mac @egamma
- ☒ linux @weinand

The sync action is back in the status bar:

Figure 1: image

If you check out to a branch that doesn't have an upstream link, we assume it is **unpublished**. The following UI should show up. Clicking it will publish the branch.

Figure 2: image

If you are in a repository without a **remote** configured, none of the previous actions should show up:

Figure 3: image

ES6 JavaScript grammar

- ☒ any platform @joaomoreno

We're using a modified TypeScript grammar for JavaScript tokenization * Have a look at some JavaScript files, including ES6, make sure they look fine * Test the default light and dark theme as well as some contributed themes * Verify all duplicates of #133

JSON schema configuration

☒ any platform @dbaeumer

The built-in schemas have been removed. They are now offered by the extensions. The schemas are now loaded directly from the schema server. That way they are more up-to-date, but might lack descriptions * Check hover and intellisense support for package.json, project.json, bower.json * Test tsconfig.json, global.json and jsconfig.json * Test that associating patterns to schema still can be done in the settings * Test that you can configure a schema from a extension. Test an extension from a server or from a file in the extension (see #489, last comment for configuration instructions)

Dark and light theme polish

☒ any platform @isidorn

Due to the move to textmate tokenizer, there were changes in the appearance of the default light and dark theme: Some themes got far more colorful, in particular JavaScript, some languages lost colors, e.g. Jade and XML. The goal was to stay as close as possible to what we had in 0.9.0: We stick to a few major colors: blue for keywords, green for comments and red for strings. * Test JavaScript and TypeScript in the light and dark theme, compare it our old state at https://opentools.azurewebsites.net/try. Note, new colors for method, function and parameters declarations * Test XML, HTML, Jade, Razor, Handlebars, CSS, LESS and SASS, JSON in the light and dark theme: They should all look consistent. Check stings, comments... * Try some of the other languages as well. Check stings, comments...