webpack

Microsoft Azure

Is Your VS Code Extension Slow? Here's How to Speed it Up!

John Papa Mar 12 · 5 min read

45

10

48

DISCUSS

- Extensions Rock -

VS Code users (and there are a lot of us) just love our extensions. There are thousands of VS Code extension to choose from and many of us have several installed. They do everything from lighting up your favorite language, formatting your code, or even colorizing your theme.

Have you ever noticed that some extensions take a few moments to initialize as you start VS Code? What might cause this delay?

> What can you do about it? A lot actually. Stay with me to see how you can help your favorite extensions load fast!

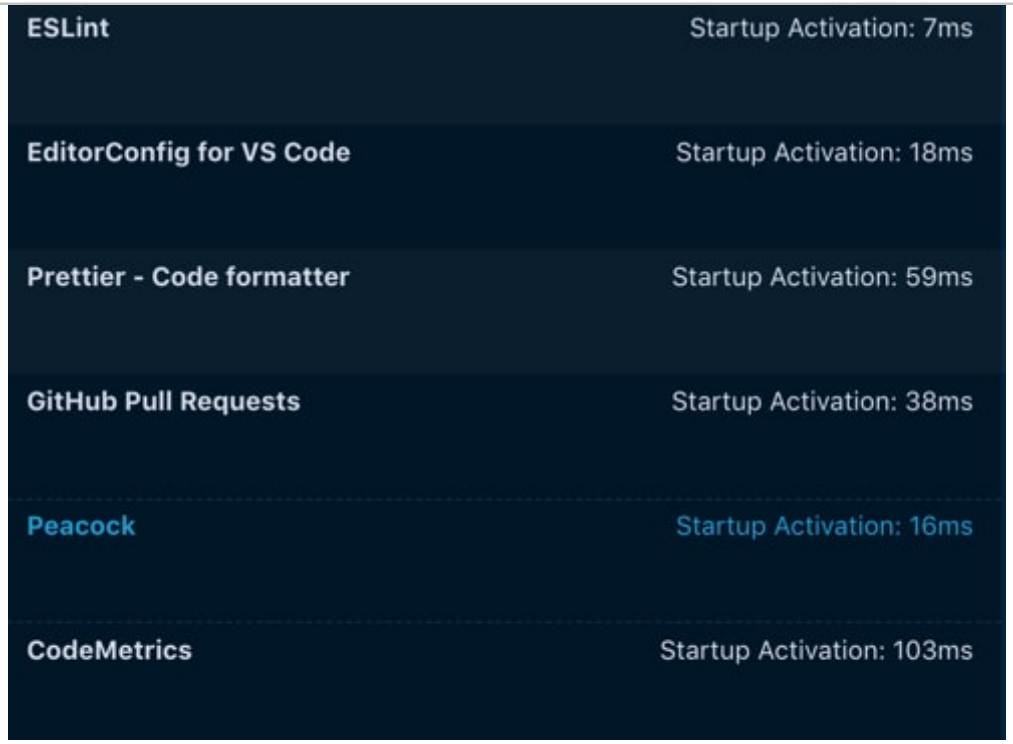
One possible cause is the number of files or the size of the extension. Some extensions have so much functionality in them that they can slow down over time.

Wait, Why is that?

When we build apps for the web, we write dozens or hundreds of files in JavaScript, CSS, and HTML. We don't want to send 1,000 files across the web to a browser as it may be a poor experience of waiting and waiting. When we write our code it isn't optimized for the browser quite as much as it can be, either. Modern tools help us solve this by compressing the files into a single (or a small set) of files. One popular tool is WebPack.

If you use the command to "Developer: Show Running Extensions" you will see a list of the activated extensions in your VS Code instance. You will also see, to the right, how long each extension took to activate in ms.

This is a great way to find out which ones may be slower activating. Notice the list below from my instance of VS Code shows a few of my installed extensions and their activation times. Obviously, some take longer than others to load, because they do more.



What can you do if one is taking too long for your tastes? (maybe 1000ms?)

Making Extensions Faster

Recently the VS Code team released the ability to use WebPack to bundle the files in extensions.

I found that my Peacock extension was putting 48 files in the package. I made a few tweaks and I cut this down by a lot.

First, I added some file to the `.vscodeignore` file

```
# Files I excluded
azure-pipelines.yml
ISSUE_TEMPLATE.md
PULL_REQUEST_TEMPLATE.md
vsc-extension-quickstart.md
node_modules/**/test/**

# After webpack, we have more to ignore
node_modules
out/
src/
tsconfig.json
webpack.config.json
```

Then I created a new branch for my extension. I went through the steps in the VS Code docs to update my project to use WebPack

My goals were to make all of these still work:

- packaging with `npm run package`
- publishing with `npm run publish`
- local and CI testing with `npm run test`
- F5 debugging with the `launch.json`
- F5 debugging the tests with the `launch.json`

The approach has me compiling both with webpack and `tsc` for the tests and debugging.

Here is my project https://github.com/johnpapa/vscode-peacock

Changed my main file in `package.json`

```
"main": "./dist/extension",
```

 45  10  48 

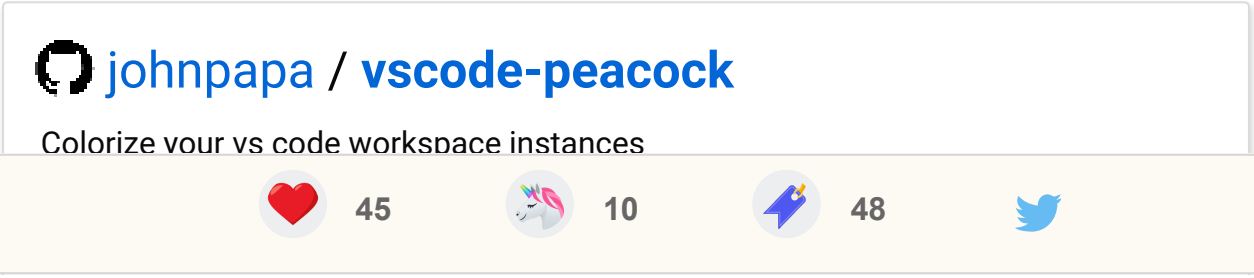
```
"scripts": {
  "package": "npx vsce package",
  "publish": "npx vsce publish",

  "vscode:prepublish": "webpack --mode production",
  "compile": "webpack --mode none",
  "watch": "webpack --mode none --watch",

  "postinstall": "node node_modules/vscode/bin/install",
  "just-test": "node node_modules/vscode/bin/test",
  "test-compile": "tsc -p ./ && npm run compile",
  "test": "npm run test-compile && node node_modules/vscode/bin/test"
},
```

My `launch.json` configurations for debugging the runtime and tests:

```
"configurations": [
  {
    "name": "Run Extension",
    "type": "extensionHost",
    "request": "launch"
```

```
      "outFiles": ["${workspaceFolder}/dist/**/*.js"],
      "preLaunchTask": "npm: test-compile"
    },
    {
      "name": "Extension Tests",
      "type": "extensionHost",
      "request": "launch",
      "runtimeExecutable": "${execPath}",
      "args": [
        "${workspaceFolder}/testworkspace",
        "--disable-extensions",
        "--extensionDevelopmentPath=${workspaceFolder}",
        "--extensionTestsPath=${workspaceFolder}/out/test"
      ],
      "outFiles": ["${workspaceFolder}/out/test/**/*.js"],
      "preLaunchTask": "npm: test-compile"
    }
  ]
```

And here is the entire repo where you can see everything in context 👇



Peacock for Visual Studio Code

Visual Studio Marketplace v2.5.1 | installs 214.09K | rating average: 4.59/5 (29 ratings)
license MIT | Greenkeeper enabled

Azure Pipelines succeeded


A Visual Studio Code extension that subtly changes the workspace color of your workspace. Ideal when you have multiple VS Code instances and you want to quickly identify which is which.

Install

1. Open **Extensions** sidebar panel in Visual Studio Code. `View → Extensions`
2. Search for `Peacock`

What Kind of Impact Can it Have?

This is a great question, and one we should definitely ask. I mean, after all, to make any code change there has to be some value. I was able to get permission (thanks to the VS Code team and Erich Gamma) to share some performance stats (unofficial tests) of two extensions you may have used.

Both of these extensions have a considerable amount of logic in them and do some pretty impressive and useful things.

Azure Account

The Azure Account extension's size and number of files decreased considerably ... like from "holy moly" to "not bad"!

The warm activation is a term for how long it takes the extension to activate, when that extension has already been installed previously (not the first time). This was cut in half for this extension. Not bad at all!

- Download size (the .vsix): 6.2M to 840K.
- Packaged files: 4300 to 11
- Warm activation time: 676ms to 338ms

Docker

The Docker extension had a noticeable warm activation improvements to

activation is how long it might take the extension to activate when it was just installed.

- Warm activation time: 3.5s to <2s
- Cold activation time (after 1st install): 20s to 2s

Tips

Several things are affected by using webpack to bundle an extension. This is why it's super important to test all of these out.

- Run the extension locally in your debugger (and test you can hit a breakpoint)
- Package the extension and load it (load from VSIX) from the menu
- Run your tests with your debugger (and test you can hit a breakpoint)
- Run your test script from `npm test`

When you are done, you can check the activation time again.

That's OK, that but if you like the extension, consider creating a pull request (PR) on its repository to enabled webpack bundling!

The great thing about OSS is that everyone gets a voice. This is a great way to help your favorite projects and your peers!

> Thanks to Erich Gamma for pointing this out to me!

Cross posted to johnpapa.net

John Papa + FOLLOW

Husband, father, & Catholic enjoying life with my family. Working @ Microsoft. Disney fanatic, web and mobile developer

@john_papa John_Papa johnpapa johnpapa.net

 45  10  48 

Add to the discussion

PREVIEW SUBMIT

Corey Alexander

Mar 12

What was your before and after loading times for the Peacock extension? Curious to see how big the savings here are!

I was under the impression that one reason we (collectively) try to bundle JS assets is to not have to deal with downloading multiple large files over HTTP,but the extension files should be already downloaded locally,so I wonder what the benefit of using Webpack here is

2

REPLY

John Papa Author

Mar 12

45 10 48

Webpack reduces the size to just the code we use and makes some other optimizations. So in this case, that is helping. I've asked the VS Code team to clarify whether the number of files has an impact or not.

Ph - and Peacock was anywhere from 8ms to 16ms to activate before, and 6ms to 14ms after. (I tried it 5 times each, with totally unscientific measuring)

2 REPLY

Corey Alexander

Mar 13

Ahh yes I was forgetting about tree shaking and the like!

I'm interested in if the number of files makes a significant difference as well! Thanks for the response!

2 THREAD

John Papa Author

Mar 13

I was told yes, the number of files will also affect load time.

BTW - I updated the post with some more stats that are pretty impressive. Thanks to the VS Code
team for sharing these and giving me permission to share with you all!

45 10 48

Classic DEV Post from Nov 10 '18

If it's Saturday and you won't be coding again until Monday, how do you get your mind off your current work?

Ben Halpern

I find myself lost in problem solving even though I can't do anything about it ...

65 62

Ben Halpern

+ FOLLOW

Keep Calm, and Keep Coding with Cosmos and Node.js

John Papa

Quickly learn to code with the Cosmos DB SDK for Node.js

92 2

John Papa

+ FOLLOW

Another Post You Might Like

If you could install only one VSCode extension?

Camilo Martinez

What would it be and why?

Camilo Martinez

+ FOLLOW

45 10 48

Getting DOM Elements using JavaScript

Atta - Jul 25

An ode to the CSS owl selector

Kevin Pennekamp - Jul 25

React Accessibility

Eric Bishard - Jul 25

Vs Code one tip a week: code folding

Aurelio - Jul 25

 45  10  48