search

Remove all console.log()s using Regex in Atom

👤 Michael Lee 🍕 🐦 🐙 Sep 1 '17 · 1 min read

#regex   **#atom**   **#texteditor**   **#javascript**

Using `console.log()` in JavaScript files are great to debug your code. But
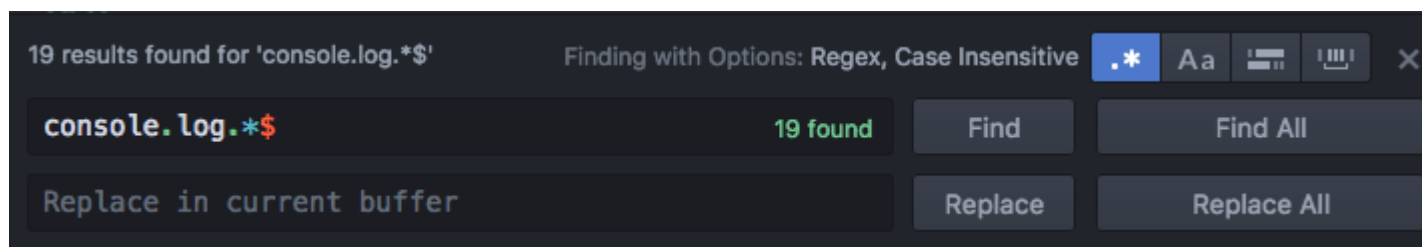
❤️ 9   🦄 4   ⚡ 0   ■■■

search

If you're using Atom as your text editor, it's easy to do this using Regex.

1. First pull up the find in buffer bar, by going to **Find > Find in Buffer**.

2. In the first search field titled *Find in current buffer* type in **console.log.*$**

3. Select the *Use Regex* option found in the upper right corner of the search panel that's designated by the icon **.***

4. Press **Find All** to find all instances of `console.log()` and then press **Replace All**

19 results found for 'console.log.*$'   Finding with Options: Regex, Case Insensitive   .*   Aa ☰ ⌨   ✕

`console.log.*$`   19 found   Find   Find All

`Replace in current buffer`   Replace   Replace All

In step 2, we used a regex to grab all instances of the string `console.log` till the end of the line. By using `.*$` the `.` indicates we want to match any character, `*` is used to indicate we want to match in indefinite amount of any characters and then finally `$` is used to indicate to match until the end of the line.

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

it's because we don't want to replace all the `console.log()` s with anything.
By leaving it blank it is saying replace with nothing.

Originally posted on Michael Lee

Michael Lee 🍕 + FOLLOW

Husband, dad and currently serving Pathos Ethos, a full-service agency as their CTO in NC. I enjoy making things for humans - things that delight, make people smile and bring value to folks.

@michael 🐦 michaelsoolee ⊙ michaellee 🔗 michaelsoolee.com

```
Add to the discussion
```

ⓘ 🖼   PREVIEW   SUBMIT

▼

| Sep 1 '17 ▪▪▪ |

👻 Ghost

Even better is to wrap all `console.log()` s into another method like this:

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

search   ✈️   🔔   ⦂⦂⦂

```
if (debug)
    console.log();
```

This allows you to toggle logging based on whatever is the reason to enable/disable logging

♡ 2   REPLY

Michael Lee 🍕 🐦 ⵔ   Sep 1 '17 ▪▪▪

Hey hey @ghost mind elaborating on your method? Is debug assuming it's an environment variable that puts the app in a certain state in which it console logs? I'm just unsure how this exactly works.

Thanks for your feedback :)

♡ 1   REPLY

👻 Ghost   Sep 1 '17 ▪▪▪

In my example, `debug` is used for demonstration purposes as a placeholder for whatever condition you use to log.

For instance, let's say if the global variable `debug` is set to true, then the script in question should log as much as possible, else it shouldn't.

Full Disclosure: I am primarily a .NET developer and only have limited experience with JS.

♡ 2   THREAD

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

Gotcha, thanks David for elaborating :)

♡ 1   REPLY

▼

Joe Gaudet 🐦 𝕆   Sep 2 '17 ▪▪▪

As per the example I included, usually you'd set the log level through an env variable. This would let you boot set a deployed app to debug / trace level without redploying.

As per: 12factor.net/config

♡ 2   REPLY

▼

Massimo Artizzu 🐦 𝕆   Sep 1 '17 ▪▪▪

The could would be shipped in production anyway, unless you're using some pre-evaluator like Prepack.

♡ 1   REPLY

▼

Massimo Artizzu 🐦 𝕆   Sep 1 '17 ▪▪▪

`console.log` 's happen and you don't always catch them. Not even with regular expressions.

Do you know what do? Linters.

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

search

Joe Gaudet 🐦 ⦿

Sep 2 '17 ▪▪▪

Agreed 100% here, encoding team rules around stuff like this in eslint configurations frees devs to spend their code review time focusing on reviewing logic and design rather than nitpicking style issues.

♡ 2

REPLY

Michael Lee 🍕 🐦 ⦿

Sep 2 '17 ▪▪▪

Definitely agree with the use of Linters. Been using ESLint for a project with other devs. Unfortunately, logs still make it into production :(Definitely could benefit from a build process that would clean that up for us. But that only fixes code and not the source ;)

♡ 1

REPLY

Daniel Worsnup 🐦

Sep 2 '17 ▪▪▪

If this continues to be an issue, it may be worth your time to set up required status checks on your repository using the Status API and webhooks. You can use these tools to automatically reject pull requests that contain code changes that fail ESLint validation.

♡ 3

REPLY

Palle ⦿

Sep 1 '17 ▪▪▪

Parsing a context free language like JavaScript with regular expressions can lead to unpredictable side

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

search   ✈️ 🔔 ⋮⋮⋮

matching the closing parenthesis (`console\.log\(.*\);?`) this could also not work in some cases.

As suggested by other users, using a logging framework, which can be disabled, may be the better solution. Or use a debugger if possible and reduce the number of `console.log` statements. If none of this helps, maybe there is a refactoring tool that can safely delete all `console.log` s.

♡ 2   REPLY

▼

Allen Macdrivel 🐦 🐙   Sep 2 '17 ■■■

Hey great post :D
The regex you wrote might match other things than "console.log" like "console logs", I suggest escaping the dot (`console\.log`).
If you want do a little more `(;|^)\s*console\.log\(.*?\)($|;)` makes sure it's either at the start of a line or behind a semicolon (same with the end)

You could also use this package which does that and the logging part for you:
github.com/vishysank/console-log-atom

♡ 2   REPLY

▼   Michael Lee 🍕 🐦 🐙   Sep 5 '17 ■■■

     Thanks Allen! Didn't know this package existed. Will look into it!

     ♡ 1   REPLY

          ❤️ 9   🦄 4   ⚡ 0   ■■■

search   ✈️ 🔔 ⊞

What happens if the log is inlined?

console.log('foo');bar();

I think a better solution here is to use a logger, with levels.

♡ 2   REPLY

▼
🧑 Michael Lee 🍕 🐦 ⚙️   Sep 1 '17 ▪▪▪

Hey hey @joegaudet . Yeah, definitely the solution wouldn't work for that since the regex matches
starting from `console.log` till the end of line. I suppose the regex could be tweaked to account for the
pattern of the closing parenthesis and semicolon or what not. I'm not savvy with regex, but came to this
solution that seems to work for me.

Do you mind elaborating on the "better solution"? I'm unfamiliar with what logger and levels is and so
doesn't really help me. It seems you're familiar with other tools to get the job done better, mind writing
a post and linking it here?

Thanks again Joe for your feedback :)

♡ 1   REPLY

▼
🧑 Joe Gaudet 🐦 ⚙️   Sep 2 '17 ▪▪▪

Hey Mike,

Happy to provide further context, my apologies for being terse I was on my phone :)

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

```
/console.log([^;]+;/
```

The character class:

```
[^;]
```

Will match against any character except a semi colon. The + indicates that it will match at least one non semi colon character. This of course assumes you are terminating all of your console logs with semi colons.

As for using loggers, there are two issues with leaving log statements around in production.

a) They can cause performance issues - you can pretty easily profile this in chrome or V8 a program that is aggressively logging will execute slower than the same program that is not. This is because logging is not a free operation.
b) Much of what you are logging will be noise in a production environment, and potential leak internal code details that aught to be secure

Usually people solve this problem by wrapping the language logging mechanisms in a logger.

A trivial example:

❤️ 9   🦄 4   ⚡ 0   ■■■

```
const Levels = {
    TRACE: 3,
    DEBUG: 2,
    INFO: 1
}

class Logger {

    level: Levels.Info,

    static trace(msg) {
        if(this.level >= Levels.TRACE) {
            console.log(`[TRACE] ${new Date()} ${msg});
        }
    }
}

Logger.level = Levels.INFO
Logger.trace('foo'); // nothing happens

Logger.level = Levels.TRACE
Logger.trace('foo'); // logs [TRACE] <datestamp> foo
```

If you're just leaving log statements around to print variable values, I'd suggest understanding break points and the debugger, as they will allow you to inspect the whole stack and not just some variables.

If the log statements you are making could be useful at a later date, but should not be present in production, a logger is probably what you need.

In Java land: slf4j.org/

(edited for regex cleanliness)

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

Hey hey Joe! Thanks for coming back and elaborating your response. This is really good stuff. I especially appreciate the two issues regarding leaving logs in production.

I've never come across a logger but can see it's usefulness. Thanks so much for sharing this, definitely something I can implement into projects developed with other devs.

♡ 1   REPLY

▼

Joe Gaudet 🐦 🐙   Sep 2 '17 ▪▪▪

Follow on with a JavaScript logging library:

github.com/winstonjs/winston

♡ 1   REPLY

▼

Paul 🐦 🐙   Sep 5 '17 ▪▪▪

"Some people, when confronted with a problem, think 'I know, I'll use regular expressions.' Now they have two problems."

Use a linter instead. :)

♡ 2   REPLY

▼   Sep 5 '17 ▪▪▪

❤ 9   🦄 4   ⚡ 0   ▪▪▪

search

♡ 1

REPLY

code of conduct - report abuse

Classic DEV Post from May 10

Why I ignore the hype (and you should too)

Steve Belovarich

It's been 20 years since I made my first website. I've been burned by the hype ti...

❤️ 102   💬 11

Another Post You Might Like

(Unknown): Script Error in JavaScript

Deeksha Agarwal

When an exception violates the same origin policy of a browser in response to onerror callback, the browser responds with a 'script error'.

❤️ 70   💬 1

❤️ 9   🦄 4   ⚡ 0   ▪▪▪

search

Frameworkless Web Development

</> Mišo Mirosavljević

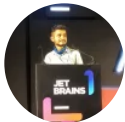It's never been easier to not use a framework for web development.

❤️ 56   〰️ 3

PWA: Uma grande lista de dicas e truques para Progressive Web Apps
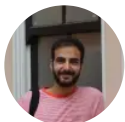
Eduardo Rabelo - Aug 5

Make any Jekyll site into a PWA

Alok Prateek - Aug 5

Update to Angular Version 8 now!

Juri Strumpflohner - Aug 5

Introducing Food Picker, the best way to pick what to eat with your colleagues !

Christopher Kade - Aug 5

❤️ 9   🦄 4   ⚡ 0   •••

search

Home   About   Privacy Policy   Terms of Use   Contact   Code of Conduct

DEV Community copyright 2016 - 2019 🐦

❤️ 9   🦄 4   ⚡ 0   ▪▪▪