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

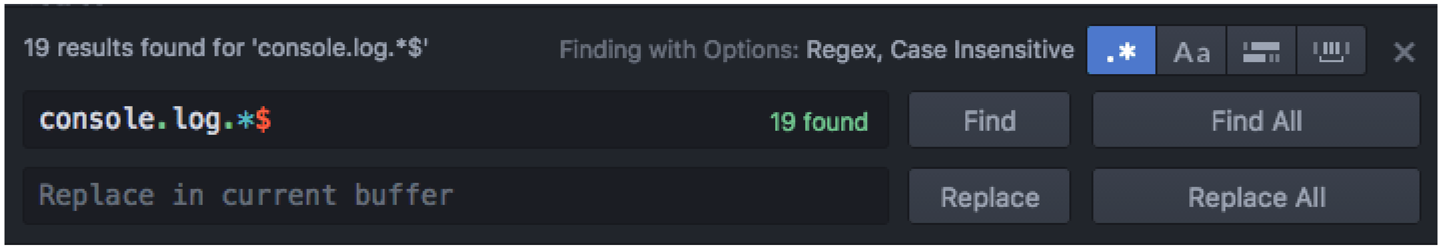
Michael Lee Sep 1 '17 · 1 min read

#regex #atom #texteditor #javascript

Using `console.log()` in JavaScript files are great to debug your code. But when it comes to shipping your code to production or a git repo it's good to clean up your code by removing `console.log()` s.

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

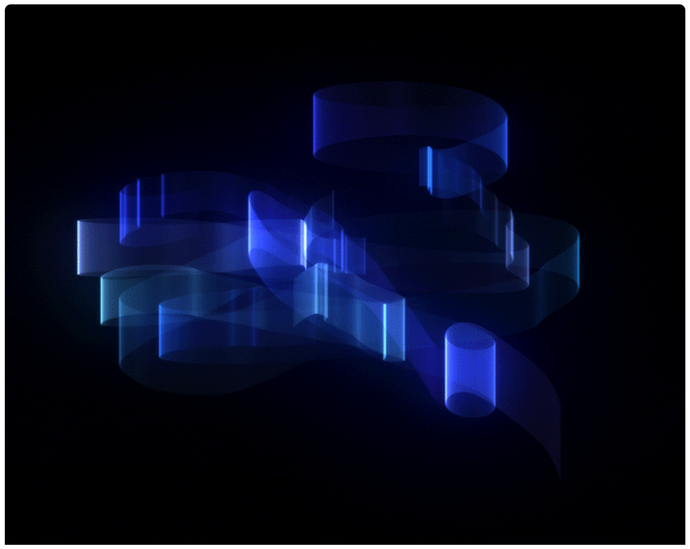
```
console.log.*$
```

19 found

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.

If you're wondering why we left the second field of the *Find in Buffer* panel 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



dev.to now has dark mode.

Select **night theme** in the "misc" section of **your settings** ♡

Michael Lee + FOLLOW

Maker of things, giver of high-fives

@michael michaelsoolee michaellee michaelsoolee.com

Add to the discussion

PREVIEW SUBMIT

Ghost Sep 1 '17

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

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

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

2 REPLY

Michael Lee 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

Michael Lee Sep 2 '17

Gotcha, thanks David for elaborating :)

♡ 1 REPLY

▼

Joe Gaudet 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 Sep 1 '17

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

1 REPLY

Massimo Artizzu 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.

As a rule of the thumb, I always say to my collaborators: "Never commit `console.log` statements, so that's why we're enforcing `no-console` in ESLint."

2 REPLY

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 effects, as regular expressions can only match words of regular languages correctly.

If somebody wrote a comment like `// lorem ipsum console.log dolor sit amet`, the regex would also match `dolor sit amet` and it would be deleted. And even if the regex was fixed to eliminate this problem by 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

Joe Gaudet — Sep 1 '17

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 :)

As for regex, you could do something like this:

```
/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:

```
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)

♡ 2 THREAD

Michael Lee Sep 2 '17

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

Michael Lee

Sep 5 '17

Thanks Paul for the suggestion :)

♡ 1 REPLY

code of conduct - report abuse

Classic DEV Post from Jan 2

If you've recently switched code editors— How's it going so far?

Ben Halpern

...

53 95

Another Post You Might Like

Using MobX with React Hooks

Ryan Dsouza

A simple example of how we can use Mobx (a truly awesome state management library) with React Hooks

63 8

Another Post You Might Like

Build Angular Like An Architect (Part 2)

Steve Belovarich

141 1

Kentico CMS Quick Tip: Minimal JSON Web APIs with IHttpHandler and .ashx Files

Sean G. Wright - Oct 28

Explaining Front-End Humor

bob.js - Oct 28

What happened to Graphical GUI Builders? - Help me understand - (They are still here, I guess)

Rohan Sawant - Oct 28

API-less Prototyping with Angular

Giancarlo Buomprisco - Oct 28

Home About Privacy Policy Terms of Use Contact Code of Conduct