openQA Project - action #91638

Ensure standard javascript code

2021-04-23 12:00 - cdywan

Status:	Resolved	**Start date:**	2021-04-23
Priority:	Low	**Due date:**	
Assignee:	kraih	**% Done:**	0%
Category:	Feature requests	**Estimated time:**	0.00 hour
Target version:	Ready		
Difficulty:			

Description

Motivation

Our currenty Javascript linter js-beautify lives up to its name, the code looks a bit nicer but it doesn't cover much. It'd be desirable to flag more logical aspects like uses of const and not just aesthestics of the code.

Acceptance criteria

- **AC1:** semistandard and more common javascript code styles are ensured

Suggestions

- Use Standard, see proof of concept: https://github.com/os-autoinst/openQA/pull/3811
- Use Semi-Standard, which has semicolons!
- Use ESLint with Semi-Standard to get even better linting and arguably go with something that works for most node projects out there
  - Example for GHA https://github.com/mojolicious/server-starter/blob/master/.github/workflows/test.yml#L22 (https://github.com/mojolicious/server-starter/blob/master/package.json#L27)
  - Failures look like this: https://github.com/mojolicious/server-starter/runs/2385146947?check_suite_focus=true
  - According to kraih using eslint in GHA annotates the code in the PR

History

#1 - 2021-04-23 17:08 - okurz

- Subject changed from Consider Standard or to Ensure standard javascript code

- Description updated

- Status changed from New to Workable

- Priority changed from Normal to Low

- Target version changed from future to Ready

Looks reasonable. I suggest to start from https://github.com/mojolicious/server-starter/blob/master/.github/workflows/test.yml and apply what we need in openQA as well.

#2 - 2021-04-27 15:11 - kraih

I wouldn't mind taking care of this. Unless someone else would like to use the opportunity to learn more about JavaScript dev tools. My recommendation would be:

1. Put a package.json and eslint.json into the repo that contain all the deps and settings (semi-standard + extended max line length)
2. Create a tools/eslint script that will check if node is installed (give a useful message otherwise), update deps with npm in the local node_modules, and then run eslint --fix over our JavaScript files (make sure it complains about violations it can't fix)
3. Add a GitHub Action that runs eslint over our JavaScript files and annotates pull requests with rule violations
4. Make sure all the temporary npm files are added to .gitignore
5. Update relevant documentation

Doing it this way also means that clever editors such as VSCode can use the settings to lint your code as you write it.

#3 - 2021-04-27 15:39 - cdywan

Just one thing comes to mind that you didn't mention. Make sure that we have a makefile target that does the linting *and which is used in our CI* - I

don't care that much about installing npm and such here, but it should be trivial to reproduce whatever we do in CI without looking up command lines to execute. This has been a source of problems before. From a glance the mentioned GHA runs 3 different commands.

#4 - 2021-04-28 13:35 - kraih

cdywan wrote:

> Just one thing comes to mind that you didn't mention. Make sure that we have a makefile target that does the linting *and which is used in our CI* - I don't care that much about installing npm and such here, but it should be trivial to reproduce whatever we do in CI without looking up command lines to execute. This has been a source of problems before. From a glance the mentioned GHA runs 3 different commands.

Actually i think a separate GitHub Action would be better, since those can annotate the pull request diff with eslint rule violations.

#5 - 2021-04-29 14:33 - kraih

- Assignee set to kraih

I'll make a proof of concept to show GitHub PR annotations.

#6 - 2021-05-12 15:08 - kraih

I've prepared a very large draft PR that replaces jsbeautifier with eslint and semistandard rules + GitHub Action.
https://github.com/os-autoinst/openQA/pull/3900

#7 - 2021-05-12 15:09 - kraih

- Assignee deleted (kraih)

#8 - 2021-05-12 15:43 - kraih

The PR also demonstrates how eslint is about more than just formatting, it's a proper linter that can find many other issues for us.

#9 - 2021-06-08 12:56 - kraih

I think one thing the eslint results have shown is that our JavaScript code is in a very poor state. Merely running a prettifier over it is not gonna solve that, we also need to make a decision how we want to organise it (ESM etc.).

#10 - 2021-06-23 06:37 - okurz

- Target version changed from Ready to future

#11 - 2021-07-13 14:04 - kraih

Here's another idea, ESLint with Prettier formatting rules. That has the advantage that we can start only with formatting rules and later add more.
https://github.com/os-autoinst/openQA/pull/4042

#12 - 2021-07-16 09:38 - kraih

- Status changed from Workable to In Progress

- Assignee set to kraih

#13 - 2021-07-16 12:14 - kraih

- Status changed from In Progress to Feedback

The PR has actually been merged, so we now have ESLint (with only one rule activated) and the Prettier formatting standard. Rule violations will result in PR annotations on GitHub, and ESLint extensions can be used with IDEs. That should resolve this ticket. Next steps will be to enable more strict rules in ESLint and to fix our JavaScript code.

#14 - 2021-07-16 12:16 - kraih

This commit shows one ESLint rule being enabled as an error. More can follow the same way, or be enabled with a category such as eslint:recommended.

#15 - 2021-07-16 17:57 - okurz

kraih wrote:

> […] That should resolve this ticket.

ok, but why did you not "resolve" it then? Do you think we need more rules enforced?

#16 - 2021-07-22 09:38 - kraih

- Status changed from Feedback to Resolved

okurz wrote:

> kraih wrote:
>
> > […] That should resolve this ticket.
>
> ok, but why did you not "resolve" it then? Do you think we need more rules enforced?

Fair point, if i leave this unresolved it could stay open forever since there will always be new ESLint rules for us to enable in the future.

#17 - 2021-07-22 10:01 - okurz

kraih wrote:

> Fair point, if i leave this unresolved it could stay open forever since there will always be new ESLint rules for us to enable in the future.

makes sense, thanks. We can update/add rules over time.

#18 - 2021-07-23 09:24 - tinita

- Target version changed from future to Ready

#19 - 2021-07-29 14:34 - kraih

Made a followup PR with the most important rules from eslint:recommended. https://github.com/os-autoinst/openQA/pull/4088