Javascript in Formulas

Last Modified on 07/27/2022 8:37 am EDT

You can choose from several step types that allow you to write your own custom Javascript. The function signature for all JS-related step types looks like:

```
/**
 * @param  trigger The trigger that started this execution
 * @param  steps   The list of steps that have been executed up until this point for this execution and all of their ste
p execution values
 * @param  info    Metadata about this formula
 * @param  config  The configuration values set on the formula instance (config variables)
 * @param  done    The callback function that you will need to call at the end of your script step
 */
function (trigger, steps, info, config, done) {
  // your Javascript will be executed here
}
```

Note the following when writing javascript in formulas:

- For all scripts, Javascript `strict` mode is enforced.
- You can use `console.log` to log data to the Javascript console to help debug your formula.
- You can use `notify.email` to send an email notification.
- ES6 is supported.
- The function parameters are immutable, meaning they cannot be assigned to directly. To change an object or value passed into the function, first copy it to your own local variable and then make the necessary changes.

Functions

- `console.log(str)` : Log something from the script. This logged value is returned in an array called `console` , which will be available to see as a step execution value. Takes a `string` as a parameter. Unlike the standard Node `console.log(str)` , this version only accepts a single parameter.
- `notify.email(to, subject, body)` : Send an email notification directly from a Javascript step. `to` can be a single email or a comma separated list of emails, `subject` is the subject of the email and `body` is the body of the email. A value will be returned in an array called `notify` , which will be available to see as a step execution value. Takes three `string` parameters. You can reference anything from the context when passing in `to` , `subject` or `body` in the same way you can access these variables elsewhere in the Javascript. For example:
  `notify.email(steps.previous-step.email, steps.previous-step.subject, steps.previous-step.emailPrefix + '<br>This is the main body.');`