Localization

The document describes the proper way to introduce new text to dashboard.

Overview

Dashboard is currently localized through the Google Closure Compiler's `goog.getMsg()` primitive. It allows the developer to define text that needs to be localized as simple variables anywhere in the code.

The localization process itself is integrated into the build pipeline and back-end component of Dashboard and happens automatically. Apart from placing new text into `MSG_` variables and using those in the angular templates, the developer is not required to do anything else.

The MSG variables

Here is an example of a single `MSG_` variable definition:

```
/** @export {string} @desc The text appears when the image pull secret name exceeds the maximal l
let MSG_IMAGE_PULL_SECRET_NAME_LENGTH_WARNING =
   goog.getMsg('Name must be up to {$maxLength} characters long.', {maxLength: '253'});
```

Guidelines: * All variable names *must* start with `MSG_`. * Placeholders (`maxLength` above) are parts of the text that should remain the same in all translations. * The description of each variable (`@desc` above) should describe the context of its usage. It is supposed to help the translator. * The JavaDoc, containing all the annotations, should be kept at 1 line to reduce scrolling time.

For a quick reference please see this cheat sheet.

Organizing the text variables

Currently, the `MSG` variables are stored in a constant dictionary at the bottom of the controller, which scopes the place (template) where they are used. The variables can then be referenced directly in the respective template's HTML code.

Here is an example of how to organize and use such variables:

```
class exampleController {
  constructor() {
    ...

    /**
     * @export
```

```
     */
    this.i18n = i18n;

    ...
  }
}

const i18n = {
  /** @export {string} @desc a simple example */
  MSG_EXAMPLE_TEXT: goog.getMsg('This is an example'),
  ... // other variables
}
```

```
<html-block>
{::ctrl.i18n.MSG_EXAMPLE_TEXT}}
</html-block>
```

In the HTML code, use one-time bindings like `{{::ctrl.i18n.MSG_EXAMPLE_TEXT}}` for efficiency.

Naming conventions and guidelines

- Consistently name the object containing the variables for a given controller `i18n`.
- In the variable's name, after `MSG_`, try to write down the name of the controller (or part of Dashboard) to indicate where the variable is being used.
- The suffix of the name should indicate the type (or role) of the text-resource in dashboard.

Table of text-resource types

Suffix	Usage
_TITLE	Title at the top of a window / view
_SUBTITLE	Subtitle placed directly beneath the title
_LABEL	Text used as a temporary placeholder or to name an input field
_ACTION	A short phrase expressing an action (verb), usually meant to be clicked
_WARNING	Warning when a validation error occurs
_TOOLTIP	Tooltip / toast text that appears on hover
_USER_HELP	Long text giving more details about a part of the UI

Suffix	Usage

Capitalization and punctuation

- Only `_USER_HELP` and `_TOOLTIP` messages have standard punctuation and end with .
- All of the messages have only their first word capitalized. Exceptions are names which appear in the middle of a message.
- The human translators are supposed to keep the original capitalization and punctuation if applicable to the specific language.