
String dedent

Champions: @jridgewell, @hemanth

Author: @mmkal

Status: Stage 2

Problem

When trying to embed formatted text (for instance, Markdown contents, or the source text of a JS program) in JS code, developers are forced to make awkward concessions for readability of the code or output. For instance, to make the embedded text look consistent with the surrounding code, we'd write:

```
1 class MyClass {
2   print() {
3     console.log(`
4       create table student(
5         id int primary key,
6         name text
7       )
8     `);
9   }
10 }
```

This outputs (using ^ to mark the beginning of a line and · to mark a leading space):

```
1 ^
2 ^.....create table student(
3 ^.....id int primary key,
4 ^.....name text
5 ^.....)
6 ^.....
```

In order to for the output to look sensible, our code becomes illegible:

```
1 class MyClass {
2   print() {
3     console.log(`create table student(
4     id int primary key,
5     name text
6   `);
7   }
8 }
```

This outputs a sensible:

```
1 create table student(  
2   id int primary key,  
3   name text  
4 )
```

With a library

It's possible to write sensible code and have sensible output with the help of libraries.

```
1 import dedent from 'dedent'  
2  
3 class MyClass {  
4   print() {  
5     console.log(dedent`  
6       create table student(  
7         id int primary key,  
8         name text  
9       )  
10    `);  
11  }  
12 }
```

This outputs the sensible:

```
1 create table student(  
2   id int primary key,  
3   name text  
4 )
```

However, these libraries incur a runtime cost, and are subtly inconsistent with the way they perform “dedenting”. The most popular package is stagnant without bug fixes and has problematic interpreting of the Template Object’s `.raw` array, and none are able to pass the dedented text to tag template functions.

```
1 pythonInterpreter`  
2   print('Hello Python World')  
3 `; // IndentationError: unexpected indent  
4  
5 const dedented = dedent`  
6   print('Hello Python World')  
7 `;  
8  
9 pythonInterpreter`${dedented}`; // <- this doesn't work right.
```

Additionally, even if a userland library were to support passing to tagged templates, the array would not be a true Template Object in proposals like `Array.isTemplateObject`. This harms the ability

of tagged templates functions to differentiate dedented templates that exist in the actual program source text (and ascribe a higher trust level to) vs a dynamically generated string (which may contain a user generated exploit string).

Proposed solution

Implement a `String.dedent` tag template function, for a tagged template literal behaving almost the same as a regular single backticked template literal, with a few key differences:

- The opening line (everything immediately right of the opening ```) must contain only a literal newline char.
- The opening line's literal newline is removed.
- The closing line (everything immediately to the left of the closing ```) may contain whitespace, but the whitespace is removed.
- The closing line's preceding literal newline char is removed.
- Lines which only contain whitespace are emptied.
- The “common indentation” of all non-empty content lines (lines that are not the opening or closing) are calculated.
- That common indentation is removed from the start of every line.

Play around with a REPL implementation.

The examples above would be solved like this:

```
1 class MyClass {
2   print() {
3     console.log(String.dedent`
4       create table student(
5         id int primary key,
6         name text
7       )
8     `);
9   }
10 }
```

This outputs the sensible:

```
1 create table student(
2   id int primary key,
3   name text
4 )
```

Expressions can be directly supported, as well as composition with another tagged template function:

```
1 const message = 'Hello Python World';
2 String.dedent(pythonInterpreter)`
3   print('${message}')
4 `;
```

In other languages

- *CoffeeScript* - block strings using `' '` and `"""` triple-quotes.
- *Java* - text blocks using triple-quotes.
- *Kotlin* - raw strings using triple-quotes and `.trimIndent()`.
- *Scala* - multiline strings using triple-quotes and `.stripMargin`.
- *Python* - multiline strings using triple-quotes to avoid escaping and `textwrap.dedent`.
- *Jsonnet* - text blocks with `| | |` as a delimiter.
- *Bash* - `<<-` Heredocs.
- *Ruby* - `<<~` Heredocs.
- *Swift* - multiline string literals using triple-quotes - strips margin based on whitespace before closing delimiter.
- *PHP* - `<<<` heredoc/nowdoc The indentation of the closing marker dictates the amount of whitespace to strip from each line.

Q&A

Why not use a library?

To summarise the problem section above: - avoid a dependency for the desired behaviour of the vast majority of multiline strings (dedent has millions of downloads per week). - avoiding inconsistencies between the multiple current implementations. - improved performance. - better discoverability - the feature can be documented publicly, and used in code samples which wouldn't otherwise rely on a package like dedent. - give code generators a way to output readable code with correct indentation properties (e.g. jest inline snapshots). - support "dedenting" tagged template literal functions with customized expression parameter behaviour (e.g. slonik). - allow formatters/linters to safely enforce code style without needing to be coupled to the runtime behaviour of multiple libraries in combination.

Additional Links

- Original TC39 thread

-
- 2020-09 Committee Meeting
 - 2022-03 Committee Meeting
 - 2022-06 Committee Meeting