Gatsby includes a powerful new way of applying compile-time code transformations, Babel macros! Macros are like Babel plugins, but instead of adding them to your `.babelrc`, you import them in the file you want to use them. This has two big advantages:

- No confusion about where a non-standard syntax is coming from. Macros are explicitly imported wherever they are used.
- No configuration files. Macros are included directly in your code as needed.

Like Babel plugins, macros run only at compile time. They are not included in the public JavaScript bundle. As such, macros have no effect on your code beyond the transformations they apply.

Installing macros

Just like plugins, many macros are published as npm packages. By convention, they are named by their function, followed by `.macro`.

For example, `preval.macro` is a macro that pre-evaluates JavaScript code. You can install it by running:

```
npm install --save-dev preval.macro
```

Some macros are instead included in larger packages. To install and use them, refer to their documentation.

Using macros

To use an installed macro, import it in your code like so:

```
import preval from "preval.macro"
```

You can then use the imported variable however the macro's documentation says. `preval.macro` is used as a template literal tag:

```
import preval from "preval.macro"
const x = preval`module.exports = 1` // highlight-line
```

When building your project this code will be transformed into:

```
const x = 1
```

Discovering available macros

Take a look at the Awesome babel macros list to find available macros you can use. Additionally, this list contains helpful resources for using macros and developing them yourself.