With ES6 there are now three types of template literals

2 came with ES6

• Template literals (code): multi-line string literals that support interpolation

• Tagged template literals (code): function calls

• Web templates (data): HTML with blanks to be filled in

Tagged template literals are created by mentioning a function before a template literal

Tagged template literals are function calls whose parameters are provided via template literals

Template literal interpolates inline expressions means that what is included in the resulting string is their result

Literals are syntactic constructs that produce values

Literals are delimited by backticks `, interpolated expressions inside the literal are delimited by ${}

Template literals always produce strings

Putting a template literal after an expression triggers a function call, similar to how a parameter list (comma-separated values in parenthesis) triggers a function call

the name before the content in backticks is the name of the function to call, the tag function

tags have to be a particular type of function

tag functions must receive

1. template strings such as 'hello '

2. substitutions that can be any expression that are interpreted as the stuff in ${…}

Template strings are known statically, at compile time

Substitutions that will be interpolated in the template string are only known at runtime

Tag functions can do whatever it wants with the parameters including ignoring them

Tag functions get two versions of each template string

1. A raw version in which backslashes are not interpreted
2. A cooked version in which backslashes are special

Tagged template literals allow you to implement custom embedded sub-languages (which are sometimes called domain-specific languages) with little effort, because JavaScript does much of the parsing for you. You only have to write a function that receives the results.

In line A, String.raw enables us to write the backslash as we would in a regular expression literal. With normal string literals, we have to escape twice: First, we need to escape the dot for the regular expression. Second, we need to escape the backslash for the string literal.

A template is basically a function: data in, text out.

The idea behind a global template object is that the same tagged template might be executed multiple times (e.g. in a loop or a function). The template object enables the tag function to cache data from previous invocations: It can put data it derived from input kind #1 (template strings) into the object, to avoid recomputing it. Caching happens per realm (think frame in a browser). That is, there is one template object per call site and realm.

The number of template strings is always one plus the number of substitutions. In other words: every substitution is always surrounded by two template strings

If a substitution is first in a literal, it is prefixed by an empty template string:

If a substitution is last in a literal, it is suffixed by an empty template string:

Template literals and tagged template literals were borrowed from the language E, which calls this feature quasi literals¹⁸

The backtick was one of the few ASCII characters that were still unused in JavaScript. The syntax ${} for interpolation is very common (Unix shells, etc.).