String-Tagged Templates

This lesson will teach you how to use a tagged template as a reusable way to format code.

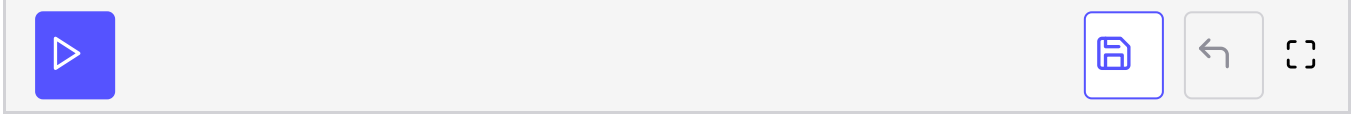
The last detail you are going to learn about TypeScript and string is what you may call a *tagged template*. This concept is rarely used, but offers a different way of specifying a function before opening a backquote. Each group of strings and placeholders is defined by a dollar sign `$` and curly braces `{}`.

In this case, you still provide an expression, but you can have the whole string manipulated by the method (placeholders and strings before and after placeholders).

In the following snippet, the **third line** uses string interpolation but, at the start, it has `analyzeString` which is the **string template**.

> The code below does not compile because we have not yet defined the template. ✕

```
const number = 84;
const number2 = 100;
const endResult = analyzeString`The number is ${number} which is not like the second number $
```



The function provided in the tagged template takes two parameters.

The first one will be all string literals in an array, and the second is a list of all placeholders.

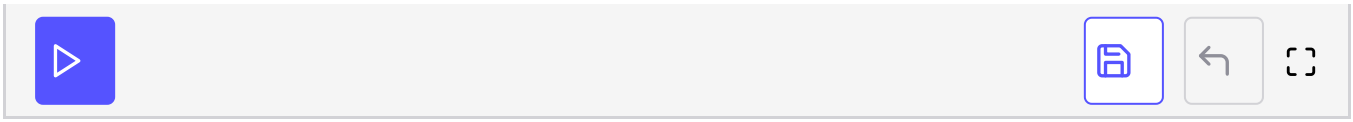
The string literal is all the text that is not a placeholder. For example, you can have a sentence that contains a few characters and a placeholder followed by some other characters and another placeholder. At the start of the string, you transform it into a tagged template using a tag. The tag is a function, in the

below case, it is `analyzeString`, that you place in front of the backquote. This function as mentioned above, has two parameters: the first parameter an array comprising two items, and the second parameter will be an array of placeholders. The return of the **function** must be a string.

```
function analyzeString(literals: TemplateStringsArray, ...placeholders: any[]) {
    let result = "";
    for (let i = 0; i < placeholders.length; i++) {
        result += literals[i];
        result += "*" + placeholders[i] + "*";
    }
    result += literals[literals.length - 1];
    return result;
}

const number = 84;
const number2 = 100;
const endResult = analyzeString`The number is ${number} which is not like the second number $

console.log(endResult);
```



The code above demonstrates the two parameters discussed previously. The `literals` is a list that contains the strings:

- `"The number is"`
- `"which is not like the second number`"

`placeholders` is also a list. In the example, it contains the two following values:

- 84
- 100

The string template could be replaced by a function. The advantage is that the string is already divided in portion with the *`literals`* (`TemplateStringsArray`) and the list of *`placeholders`* (`any`). The **tagged template** can also return something other than a string.