Template Literals

Definition

Usually, when displaying or returning a message containing variables, we always end up with a lot of **plus signs “+”** (as shown below).

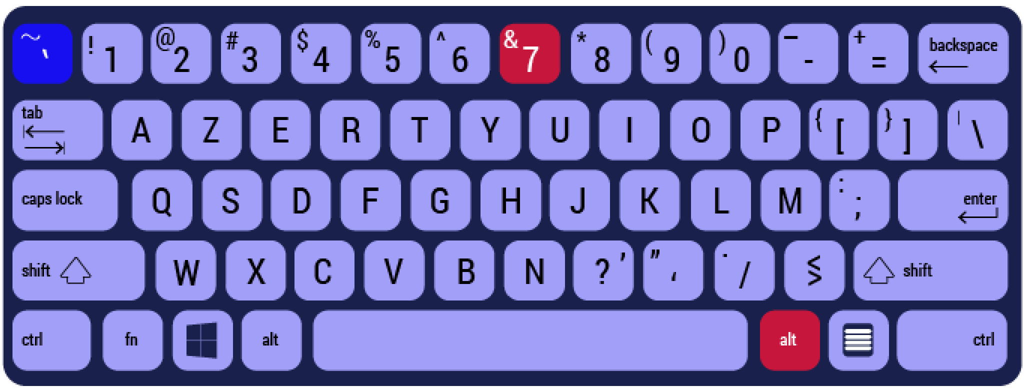
Fortunately, **ES6** introduced a new way to combine strings and variables using special quotes `` called **template literals**.

```
let name = "John"
// using normal quotes
console.log("Hello "+name+", how are you ?");
// using ES6 template literals
console.log(`Hello ${name}, how are you ?`);
```

How to use it?

To use this special apostrophe ('), known as backtick or backquote, you can press on your keyboard "AltGr+7" or press the "~" button, all depending on your keyboard language layout.

For more details about different keyboards combinations have a look at this: Link



Example

To sum up, template literals can be used to:

- Make a multi-line string.
- Create expressions.
- Put variables inside a string.

Ternary operator

A ternary operator is used to evaluate a condition and executes a block of code based on the condition.

```
condition ? expression1 : expression2
```

The ternary operator evaluates the condition.

- If the condition is `true`, expression1 is executed.
- If the condition is `false`, expression2 is executed.

The ternary operator takes three operands, hence, the name ternary operator. It is also known as a conditional operator.

Example

In this example we are going to determine if a student passed or failed in the exam based on marks obtained.

```
//program to check pass or fail
let marks = prompt ('Enter your marks:');
```

```
 // check the condition
let result = ( marks >=40 ) ? 'pass' : 'fail';
console.log (`You ${result} the exam.`);
```

- Output 1

```
Enter your marks: 78
You pass the exam.
```

- Output 2

```
Enter your marks: 35
You fail the exam.
```

Ternary Operator Used Instead of if...else

In JavaScript, a ternary operator can be used to replace certain types of `if..else` statements. For example,

You can replace this code

```
// check the age to determine the eligibility to vote
let age = 15;
let result;

if (age >= 18) {
      result = "You are eligible to vote.";
} else {
      result = "You are not eligible to vote yet.";
}

console.log(result);
```

with this:

```
// ternary operator to check the eligibility to vote
let age = 15;
let result =
    (age >= 18) ? "You are eligible to vote." : "You are not eligible to vote yet";
console.log(result);
```

The output of both programs will be the same.

- Output

```
You are not eligible to vote yet
```

Chained ternary operators

You can also nest one ternary operator as an expression inside another ternary operator. For example,

```
// program to check if number is positive, negative or zero
let a = 0;
let result = (a === 0) ?  "zero" : a> 0 ? "positive"  : "negative";
console.log(`The number is ${result}.`);
```

- Output

```
The number is zero.
```