ANSWER 6-

Template literals, introduced in ECMAScript 6 (ES6), provide an improved way to work with strings in JavaScript. They allow you to create strings with embedded expressions and multiline capabilities. Here's how you can use template literals:

Syntax: Template literals are delimited by backticks (`) instead of single or double quotes. Embedding Expressions: You can embed expressions within template literals using ${} as a placeholder. Any valid JavaScript expression can be placed within ${} and its value will be inserted into the string.

```
const name = 'John'; const age = 25;

const message = `My name is ${name} and I'm ${age} years old. ` ;
console.log(message);

// Output: My name is John and I'm 25 years old.
```

Multiline Strings: Template literals allow multiline strings without the need for escape characters or concatenation.

```
const multiline = ` This is a multiline string. `;

console.log(multiline);

/* Output: This is a multiline string. */
```

Expression Evaluation: Within ${}, expressions are evaluated and their values are converted to strings. This allows for dynamic content in the string.

```
const a = 10; const b = 5;

const sum = `The sum of ${a} and ${b} is ${a + b}.`;
```

```
console.log(sum);
// Output: The sum of 10 and 5 is 15.
```

Tagged Template Literals: Template literals can be "tagged" with a function that modifies the output of the template. The tag function is placed before the template literal, and it receives the template as separate arguments, along with the evaluated expressions.

```
function tagFunction(strings, ...values) { // strings: array of string literals // values: array of evaluated expressions // ...rest: any additional arguments // Manipulate the strings and values as desired return 'Modified output'; }
const message = tagFunction`Hello, ${name}!`;
console.log(message);
// Output: Modified output
```

Template literals provide a more concise and flexible way to work with strings, making it easier to interpolate expressions and work with multiline content. They are widely used in modern JavaScript development for creating dynamic and readable string representations