1.Invoke function:

It is common to use the term "call a function" instead of "invoke a function".

The code inside a function is executed when the function is invoked.

```
function myFunction(a, b) {
    return a * b;
}
myFunction(10, 2);
```

2.difference between break and continue:

You have already seen the `break` statement used in an earlier chapter of this tutorial. It was used to "jump out" of a `switch()` statement.

The `break` statement can also be used to jump out of a loop:

The `continue` statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

The `break` statement, without a label reference, can only be used to jump out of a loop or a switch.

3.different type of function:

There are 3 ways of writing a function in JavaScript:

- Function Declaration
- Function Expression
- Arrow Function

Function declaration:

```
// Function declaration
```

```
function add(a, b) {
        console.log(a + b);
}

// Calling a function
add(2, 3);
```

Function expression:

```
      // Function Expression
const add = function(a, b) {
      console.log(a+b);
}

// Calling function
add(2, 3);
```

Arrow Functions:

```
// Single line of code
let add = (a, b) => a + b;

console.log(add(3, 2));
```

4.string inbuilt method:

String.prototype.toLowerCase()

Returns the calling string value converted to lowercase

```
const sentence = 'The quick brown fox jumps over the lazy dog.';

console.log(sentence.toLowerCase());
```

String.prototype.trim()

Trims whitespace from the beginning and end of the string. Part of the ECMAScript 5 standard.

```
const greeting = '   Hello world!   ';

console.log(greeting);
```

console.log(greeting.trim());

charAt()

const sentence = 'The quick brown fox jumps over the lazy dog.';

const index = 4;

console.log(`The character at index ${index} is ${sentence.charAt(index)}`);

String.prototype.indexOf(searchValue [, fromIndex])

Returns the index within the calling String object of the first occurrence of searchValue, or -1 if not found.

toString()

`toString()` is one of the most commonly-used functions pertaining to strings. It belongs to *all* `Object`s and returns a string-representation of the object, effectively converting an object of any type into it's string representation:

```
let x = 100;

let y = 200

let z1 = x+y

let z2 = x.toString() + y
```

```
console.log(z1); // Output: 300

console.log(z2); // Output: 100200
```

concat()

`concat()` adds two strings together and returns a new string:

```
let x = "some ";

let y = "string";


console.log(x.concat(y)); // Output: some string
```

5.replace and replace all:

Replace :

The first parameter can be a string or a regular expression. If it is a string value, only the first instance of the value will be replaced.

```
const str = "JavaScript Courses";

const newStr = str.replace('JavaScript', 'Java');


console.log(newStr); // Java Courses
```

replaceAll();

This method replaces all appearances of the search string with the replacement text and returns a new string.

6.ternery operator:

The conditional (ternary) operator is the only JavaScript operator that takes three operands: a condition followed by a question mark (?), then an expression to execute if the condition is truthy followed by a colon (:), and finally the expression to execute if the condition is falsy.