Title: What does calling a function mean?

Answer:

Defining a function does not execute it. Defining the function simply names the function and specifies what to do when the function is called. **Calling** the function actually performs the specified actions with the indicated parameters. For example, if you define the function `square`, you could call it as follows:

```
square(5);
```

The preceding statement calls the function with an argument of 5. The function executes its statements and returns the value 25.

Functions must be in scope when they are called, but the function declaration can be hoisted (appear below the call in the code), as in this example:

```
console.log(square(5));
/* ... */
function square(n) { return n * n; }
```

The scope of a function is the function in which it is declared, or the entire program if it is declared at the top level.

Note: This works only when defining the function using the above syntax (i.e. `function funcName(){}`). The code below will not work. That means, function hoisting only works with function declaration and not with function expression.

```
console.log(square); // square is hoisted with an initial value undefined.
console.log(square(5)); // TypeError: square is not a function
var square = function(n) {
  return n * n;
}
```

Tags: functions / methods, javascript