Learning Objectives:
• Understand the concept of a "function" and why it is important
• Deeply comprehend how variables outside a function differ from those inside
• Learn the JavaScript syntax for creating and calling functions
Functions

Anything complex is easier to understand if it is broken into parts. This is true for computer programs as much as (or more than) anything else in the world. Writing programs in terms of little, isolated, cohesive chunks of useful code is called modularity, and it is an eminently worthwhile goal.

One way that JavaScript (and nearly every programming language) allows you to improve modularity is the idea of a function. A function is really nothing more than a "subprogram" that has its own input(s) and its own output. Stringing together many functions to build a large program is similar to building a car engine out of car parts like pistons and gears, rather than trying to build it out of atoms and molecules.
Variables in functions

Perhaps the most important mechanical aspect to functions is how \textbf{variables} are treated. Recall that every running program has a "memory worksheet" in which its named variables have values that may change over time as the program runs. If a program chooses to use (or "call") a function, the program gives (or "passes") information to the function as its input(s), but the main program's variables are \textbf{not} visible to the function. Similarly, any variables that the function uses are \textbf{not} visible to the main program.

The way the function knows which of the inputs it's given correspond to which things is by means of \textbf{ordering}. The inputs a function is given are simply numbered: input #1, input #2, input #3. The function does \textbf{not} know or care what variable names the main program might have used in preparing that input. All it knows is that it is given (say) three inputs, in order, and it depends on that order being correct in knowing how to interpret what's what.

The way a function gives back an answer to the program that called it is via a \textbf{return value}, which is simply the "answer" (or "output") that the function provides.
JavaScript functions

In JavaScript, there are two elements to making a function work: (1) defining the function, and (2) calling the function.

(1) Defining the function: at the top of your JavaScript file, you list all of your functions, in any order. For each one, you use the keyword "function" followed by the function's name. Then, in a comma-separated list, you list the variable names as seen by the monster. Then the code for the function follows inside curly braces, which will contain (usually one) return statement. For example:

```
function bmi(height, weight) {
    var theBmi = 703 * weight / (height * height);
    return theBmi;
}
```

```
function howCool(numberOfFacebookFriends) {
    var coolness;
    if (numberOfFacebookFriends < 10) {
        coolness = "not very cool";
    }
    if (numberOfFacebookFriends >= 10 &&
        numberOfFacebookFriends < 500) {
        coolness = "real cool";
    }
    if (numberOfFacebookFriends >= 500) {
        coolness = "facebook nerd";
    }
    return coolness;
}
```
(2) Calling the function: to "call" a function is actually to invoke it; to make use of its functionality. Calling a function is simply a matter of typing the function's name plus a list of the values you want to give it. You then use a variable to capture the return value by assigning the function call to that variable.

For example:

```javascript
var stephensBmi = bmi(inches, pounds);

var howCoolIsKaren = howCool(karensFBfriendCount);
```

Note that the names of the variables literally passed to the function are not usually the same as what the function calls them internally. In fact, you don't even have to have a variable in order to call a function! Consider:

```javascript
var stephensBmi = bmi(74, 211);

alert("Karen is this cool: " + howCool(296));
```