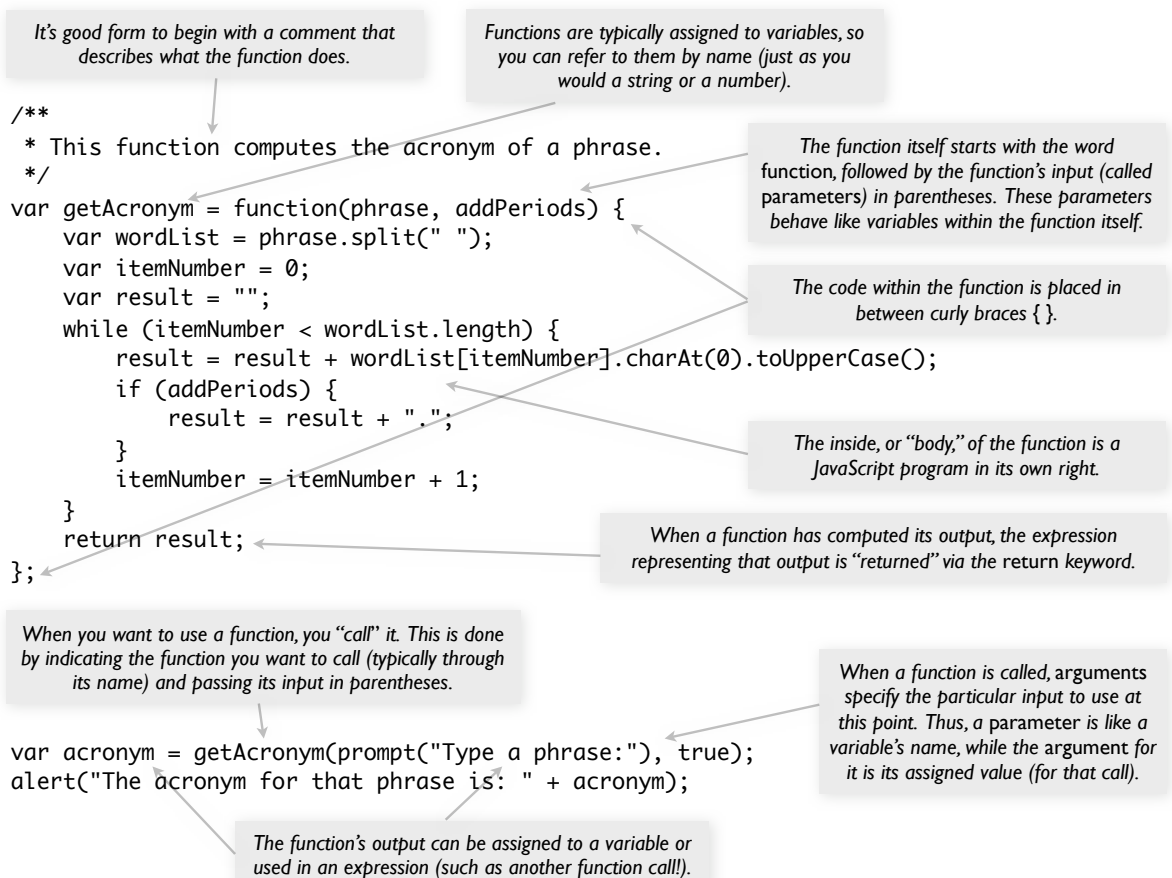# JavaScript Functions

- A *function* is a major programming construct that you have not yet seen — but, I'd argue, once you've seen it, you might wonder how you ever lived without it **:)**

- The reality is that you've already been using functions — *alert, prompt, parseFloat, parseInt*, among others, are actually functions

- What you haven't done yet is to *make your own* functions — until now

What is a Function?

- A function is an object that carries out a specific computation (i.e., it represents an algorithm)

- Functions are written ("defined") once, but can be used or run ("called") over and over again

- Functions allow you to express an algorithm once, then use it as many times as you like without having to copy its instructions all over the place

- Since they represent algorithms, functions have explicit constructs for working with their *input* and *output*

Anatomy of a Function

- As with any other JavaScript construct, a function must be defined using a particular format

- This format is likely more complicated than anything you have seen before; your reward for mastering it is the ability to define sequences of code that you can use over and over, for the life of a program

  ◇ Experienced programmers build "libraries" of useful functions, and frequently share them with (or sell them to) others

- To illustrate this format, we have chosen to "annotate" a sample function and an instance of its use

It's good form to begin with a comment that describes what the function does.

Functions are typically assigned to variables, so you can refer to them by name (just as you would a string or a number).

```javascript
/**
 * This function computes the acronym of a phrase.
 */
var getAcronym = function(phrase, addPeriods) {
    var wordList = phrase.split(" ");
    var itemNumber = 0;
    var result = "";
    while (itemNumber < wordList.length) {
        result = result + wordList[itemNumber].charAt(0).toUpperCase();
        if (addPeriods) {
            result = result + ".";
        }
        itemNumber = itemNumber + 1;
    }
    return result;
};
```

The function itself starts with the word function, followed by the function's input (called parameters) in parentheses. These parameters behave like variables within the function itself.

The code within the function is placed in between curly braces { }.

The inside, or "body," of the function is a JavaScript program in its own right.

When a function has computed its output, the expression representing that output is "returned" via the return keyword.

When you want to use a function, you "call" it. This is done by indicating the function you want to call (typically through its name) and passing its input in parentheses.

When a function is called, arguments specify the particular input to use at this point. Thus, a parameter is like a variable's name, while the argument for it is its assigned value (for that call).

```javascript
var acronym = getAcronym(prompt("Type a phrase:"), true);
alert("The acronym for that phrase is: " + acronym);
```

The function's output can be assigned to a variable or used in an expression (such as another function call!).

A Small "Function Library"

- The code on the following pages demonstrates how functions can be called as much as you need, as well as how functions can call other functions

- In the first part of the program (below), three of the "Intro 8" have been rewritten as functions

- Note how the *splitCheck* function uses the *addTip* function — the savings may not be clear here since *addTip* is fairly short, but imagine the typing (and headaches) you'd save if *addTip* were more complicated!

```
/**
 * This function calculates a tip and adds it to a restaurant bill.
 */
var addTip = function(total, tipRate) {
    return total + (total * (tipRate / 100.0));
};

/**
 * This function splits a check.
 */
var splitCheck = function(total, tipRate, guestCount) {
    return addTip(total, tipRate) / guestCount;
};

/**
 * This function calculates a person's age.
 */
var calculateAge = function(birthdate) {
    var today = new Date();
    var difference = today.getYear() - birthdate.getYear();
    if ((today.getMonth() < birthdate.getMonth()) ||
        ((today.getMonth() === birthdate.getMonth()) &&
         (today.getDate() < birthdate.getDate()))) {
        difference = difference - 1;
    }
    return difference;
};
```

Calling the "Library"

- The second half of the sample program (below) assumes that the code on the previous page has been run, thus making the three functions available through the variables (names) *addTip*, *splitCheck*, and *calculateAge*

- This "main" part of the program repeatedly asks the user for a command, and does what the command says (it ignores any command that is not recognized)

- The program keeps on doing this until the user types "quit" as his or her command

```
/**
 * This "main" part of the program allows the user to choose among the
 * three available functions.
 */
var command = "";

// Keep going until the user types "quit" (exactly as shown).
while (command !== "quit") {
    command = prompt("Which function would you like to run (tip/split/age/quit)?");

    // We separate the way we gather the input (in this case, using prompt) from
    // the functions themselves.  That way, if needed, we can supply input in a
    // different way (for example, via direct values: addTip(200, 15) ).
    if (command === "tip") {
        var bill = parseFloat(prompt("How much was the bill?"));
        var rate = parseFloat(prompt("How much (in %) would you like to tip?"));
        alert("The total is $" + addTip(bill, rate) + ".");
    } else if (command === "split") {
        var bill = parseFloat(prompt("How much was the bill?"));
        var rate = parseFloat(prompt("How much (in %) would you like to tip?"));
        var party = parseInt(prompt("How many were in the party?"));
        alert("Each person should pay $" + splitCheck(bill, rate, party) + ".");
    } else if (command === "age") {
        var date = new Date(prompt("When is your birthdate (mm/dd/yyyy)?"));
        alert("You are " + calculateAge(date) + " years old.");
    }
}
```