Functions: Input, Processing, Output

1. From Pictures to Numbers

Our first experience with Javascript was to write functions to modify the attributes of elements. Suddenly, every element on the page became a variable, something we could change in response to events.

Today we look at another way to make web pages come alive - program them to process information. There are tons of Web sites that analyze data and report results. You can enter a travel destination, and a page will tell you the fare. You can enter a number of dollars and be told the equivalent in euros.

Consider this webpage:

```
<html>
<head>
     <title>Square Root Finder</title>

     <script type='text/javascript'>

     function find_root()
     {
          var  num;
          var  root;

          num = prompt("Find the square root of what number? ");
          root = Math.sqrt(num);
          alert("The square root of " + num + " is " + root );
     }

     </script>

</head>
<body>
<h4>Square Root Finder</h4>

<p>Click <input type='button' onclick='find_root()' value='here'></p>
</body>
</html>
```

This page asks the user for a number and tells the user the square root of that number. How does it work?

There are four main parts to any program that processes data:

input - the user enters some information
storage - the computer stores the information in variables
processing - the computer performs some operations on the data
output - the computer presents the results to the user

How do those four parts work in this program?

input - the prompt() function gets input from the user
storage - the program stores the user input in a variable called num
processing - the program uses a Javascript function called Math.sqrt()
output - the alert() function displays a message in a pop-up box

The computation of the square root is done by a function. A function in Javascript, like a function in math, is a procedure that takes in a value, does some processing, and sends back a result. Javascript provides a function to compute square roots, but what if you want to compute something that Javascript does not have a function to do? Easy, write your own function to perform that action. Today we shall focus on functions.

2. Storage

Javascript, like the other programming languages we have studied, uses variables to store data. A variable is storage box with a name. In Javascript one creates a variable as follows:

```
var nameofvariable;
```

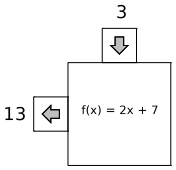
Use variable names that are not likely to be built-in commands; Javascript will not complain, but the program will not work. You store a value in a variable as follows:

```
num = 3;
```

The variable goes on the left, the value goes on the right. The effect is that the value is stored in the variable.

3. Processing: Operations and Functions

In Javascript, there are two ways of processing data: operations and functions. Operations include the math operations like +, -, *, /, and . Comparison operations include <, >, ==, <=, >=, !=. Boolean operations are ||, &&, and ! (or, and, not). In addition to these basic operations, Javascript provides several functions, and Javascript allows you to create your own functions. We now look at user-defined functions. A function in math looks like:




```
//
// function to compute seven
// more than twice its argument
//
function f(x)
{
   return 2*x + 7;
}
```

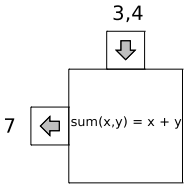
A function in math has a input value, a formula, and produces an output value. This diagram shows the input value going in the top, shows the rule in the box, and shows the result popping out the door on the left. The value 3 goes in, the machine applies the formula, and the value 13 pops out the left. The same function is expressed in Javascript in the right part of the diagram. The parameter represents the input value, the code inside the curly braces specifies what the operation is, and the Javascript code word `return` tells the computer to send that value out as the result.

For practice, do these operations:

1. Given that $f(x) = 2x + 7$, find
   a) $f(3)$
   b) $f(10)$
   c) $f(2r)$
   d) $f(f(2))$
   e) $f(3 + f(4))$

Functions of Several Parameters

In math, a function can take more than one input value. Mathematicians call these *functions of several variables*, but computer science people would call them *functions of several parameters* or *functions that take several arguments*.




```
//
// function to return sum
// of args
//
function sum(x,y)
{
   return x+y;
}
```

Here is a function that takes two numbers as arguments, adds the two numbers, and outputs the sum. In computer terms, we say this function *returns* the sum. Think of a dry cleaner. You input dirty clothes, and when you return, the cleaner returns the processed clothes to you. To the right is the Javascript code. Simply name the two parameters in the first line of the function, and refer to those parameters by name. In sh, we would use $1 and $2 to refer to the arguments. Here, we refer to arguments by name, not by position.

For practice, do these operations:

2. Given that $f(x) = 2x + 7$ and $g(x, y) = x + y$, find
   a) $g(3, 4)$
   b) $g(g(1, 2), 5)$
   c) $f(g(2, 6))$
   d) $g(f(2))$
   e) $g(f(2), g(1, 3))$

4. Writing Functions

We can write Javascript code to define functions. A function takes zero or more arguments, does something with those values, then returns a value. For example, a Javascript function could compute the celsius version of a

temperature given in fahrenheit. Here is such a function:

```
function fahr_to_cels(f)
{
    var result;
    result = (f-32) * 5/9;
    return result;
}
```

This function uses a variable to hold the computed result before returning it. We really do not need that variable, but I included it to show you that functions can define and use their own variables, sort of like scratch paper when you are working out the value of some problem.

3. For practice, write functions with the given name to perform the operations:

a) `square` that returns the square of its argument

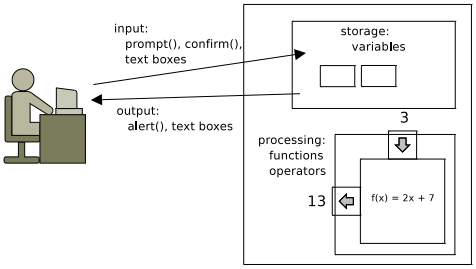
b) `mult` that returns the result of multiplying its arguments

c) `cube` that returns the cube of its argument

d) `larger` that returns the larger of its two arguments

5. Input and Output

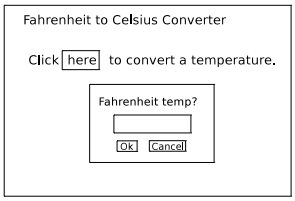
We now know how to write functions and use variables. The other two main parts of any program are input and output.



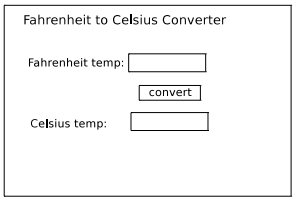
At the left we have a user sitting at a web browser and at the right is the code for the web page. The interaction consists of four parts: user enters data, the program stores the data in variables, the program processes the data, the program displays the results to the user. The processing is done using mathematical, comparison, and boolean operations. But what about the input and output?

You visit web pages. How does a web page ask for input, and how does a web page display results? Two main methods are pop-up windows and text boxes.

Pop-up Version



TextBox Version



6. Code for Pop-Up Version

```
<html>
<!-- fahr to cel converter using prompt
  -- works ok, but needs to handle empty input, cancel, and NaN -->
<head>
    <title>Fahr to Cels Converter</title>
    <script type='text/javascript'>

       function convert()
       {
           var fahr;
           var cels;

           fahr = prompt("What is the temperature in fahrenheit?");
           cels = Math.round(compute_cels(fahr));
           alert(fahr + "F = " + cels + "C");
       }
       //
       // compute the celsius temp from the parameter f
       //
       function compute_cels(f)
       {
           var answer;
           answer = (parseFloat(f)-32) * 5/9;
           return answer;
       }
    </script>
</head>
<body>
    <h3>Fahrenheit to Celsius Converter: Prompt Version</h3>

    <p> This program will compute the celsius temperature for
        any fahrenheit temperature you enter.  </p>

    <p>Click <input type='button' value='here' onclick='convert()' />
       to convert a temperature.
    </p>
</body>
</html>
```

7. Code for Text Box Version

```
<html>
<!-- fahr to cels converter using text input boxes
  -- ok, but needs to handle empty input, non a number
  -->
<head>
    <title>Fahr to Cels Converter</title>
    <script type='text/javascript'>

       // get value from text box, then compute celsius,
       // then put value in text box
       function f2c()
       {
           var fahr;
           var cels;

           fahr = document.getElementById('fahr').value;
           cels = compute_cels(fahr);
           document.getElementById('cels').value = cels;
       }
       //
       // compute the celsius temp from the parameter f
       //
       function compute_cels(f)
       {
           var answer;
           answer = (parseFloat(f)-32) * 5/9;
           return answer;
       }

    </script>
</head>

<body style='background-color: lightgreen; color: blue;'>

    <h3>Fahrenheit to Celsius Converter</h3>

    <table border='1' cellpadding='5' cellspacing='0'>
       <!-- input goes here -->
       <tr>
        <td>Fahrenheit temp:</td>
           <td><input type='text' size='4' id='fahr' value=''></td>
       </tr>

       <!-- button goes in row 2 -->
       <tr>
           <td></td>
           <td><input type='button' value='convert' onclick='f2c()' /></td>
       </tr>

       <!-- output goes here -->
       <tr>
        <td>Celsius temp:</td>
           <td><input type='text' size='4' id='cels' value=''></td>
       </tr>
    </table>
</body>
</html>
```

8. Basics of Text Boxes

We create a text input box with the tag:

```
<input type='text' size='width in chars' id='some id' value='initial value' />
```

The `value` attribute of the text input element can be treated as a variable. Any value the user types there can be read as input, and any value the program stores there can be seen by the user as program output.

One big advantage of text input elements is the user can see several at once. When we use a prompt and alert, the user cannot see the input after it is entered, and cannot see the output after the alert box is dismissed.

9. A Function that Returns a String

We have looked at functions that take in numbers as arguments and return numbers. Javascript allows us to write functions that use strings for arguments and/or return values. Here is a slightly fancier program:

```
<html>
<!-- trend.html
  -- reports the temperature trend for three days
  -- Needs: check for cancel and check for blank and for NaN
  -->
<head>
    <title>Trend Computer</title>
    <script type='text/javascript'>

    function temp_trend()
    {
        var t1, t2, t3;
        var trend;

        t1 = prompt("What is temperature on day1? ");
        t2 = prompt("What is temperature on day2? ");
        t3 = prompt("What is temperature on day3? ");

        trend = find_trend(t1, t2, t3);

        alert("Temperature is " + trend);
    }
    //
    // find the trend by looking at three numbers
    //
    function find_trend(a,b,c)
    {
        if ( a < b && b < c ){
            return "rising";
        }
        if ( a > b && b > c ){
            return "falling";
        }
        if ( a < b && b > c ){
            return "up then down";
        }
        if ( a > b && b < c ){
            return "down then up";
        }
        if ( a == b && b == c ){
            return "steady";
        }
        return "varied";
    }

    </script>
</head>

<body>

    <h3>Temperature Trend describer: Prompt Version</h3>

    <p>
        This program will describe the trend in temperatures
        over three days.
    </p>

    <p>Click <input type='button' value='here' onclick='temp_trend()' />
       to analyze temperature changes.
    </p>
</body>
</html>
```

10. More Projects

With prompts and text inputs, we now know about the two main forms of input/ouput for web pages. The core of a web page is the function. A function does the actual computation, but the input and output elements are essential too. Without these elements, the user cannot send data into a functions and the user cannot receive the results of a function.

Try writing the following pages:

a. Find the larger of two numbers
b. Find the square of a number
c. Find the average of two numbers