Functions

Functions (also referred to as procedures) are a part of our everyday lives. Individuals and organizations utilize them as a way to assure that a task is performed in a thorough and predictable manner each time it is needed. Computers also use functions in this manner. Functions encode the operations needed to accomplish a task. In other words, functions encode algorithms.

Importance of Functions

- Functions encapsulate functionality (useful instruction) so that it can be used anywhere, anytime.
- Functions help manage complexity
  - If you find yourself writing the same code statements multiple times in your program, this is a good indication that you need a function to minimize the amount of code.
  - Functions also clean up your code by placing them all in one area to leave the rest of the HTML/script clean.

Remember Friday's Program?

- Friday we coded up a simple program to test if a number entered was positive, negative or zero
- All the code was put in the event handler, onClick
  - That is OK, but very messy-especially if onClick is handling a LOT of instructions
  - We could have encapsulated that code elsewhere and called it when needed
  - All the program needs to send when calling that function is the number entered by the user and all the instructions could be run.

A Simple Scenario

- We use email every day to send mail to friends in the state, across the country or around the world.
- You receive mail from your friend in Australia telling you it's 30°
- The temperature is Celsius, but you want Fahrenheit
- You could do a quick calculation, but since you write to this person a lot, it would be better to just write a little function to do the calculation every time.
Structure Of A Function

- Functions encapsulate (package up) a computation to be used anywhere, anytime
- Functions have the following features:
  - Name: term used to refer to the task the function performs, example: convertC2F
  - Definition: The steps that will accomplish the task. Also known as the body of the function
    \[ f = \frac{9c}{5} + 32 \]
  - Parameters: the data to be used by the function-the inputs
    - Parameters can be values, variables or object properties
  - Declaration: the entire package of the name, definition and parameters

You've already seen Functions!

- Event Handlers are like pre-built function declarations
  - The event handling routine that we filled with code on Friday was a type of function
  - They just wait for programmers to add instructions to be executed
  - Instead of being called in the code, they are called every time a user activates that particular event

The Whole Function Package

- Function Declaration: Includes name, definition and parameters

```
function convertC2F (c) {
    var f;
    f = (9 * c) / 5 + 32;
    return f;
}
```

Calling A Function

- The function declaration specifies how the function works and only needs to be given once
- The function call says when, where and with what values the function will be performed (executed)
  - A function call can be used anywhere that the task to be performed is needed.

```
... onClick="ConvertC2F(document.stinky.number.value)" ......
```

```
function ConvertC2F (c) {
    var f;
    f = (9 * c) / 5 + 32;
    return f;
}
```
Methods are built-in Functions

- All methods are pre-defined program statements to be run when called.
  - The only difference is that you don’t have to define the method, just call it when needed
    - `document.write("Hello World");`
    - `document.write(d.getDate());`
    - `prompt("What is your name?", "Enter name here")`
    - `alert("Wrong answer!!!!")`
- You just provide the name of the function and arguments:
  - The name is the combination of the object and method with arguments in parenthesis

Variables: Global vs. Local

- Variables are used all over in programs
- Most variables declared can be referenced anywhere in the program
  - There values can be changed from anywhere
  - They are global
- However, variables given as formal parameters or declared inside a function, are local
  - They only exist inside the function and can’t be changed or referenced from elsewhere in the program
  - This means you can reuse variable names created in functions without name conflicts.
  - Just make sure you have no global variables with the same name

Parameter Correspondence

- The arguments name the input values and the function then can output results
- The number of formal parameters in the declaration must match the number of arguments in the call, and they correspond one-to-one

```javascript
function ConvertC2F (c) {
  var f;
  f = (9 * c) / 5 + 32;
  return f;
}
```

```javascript
onClick = "answertextbox = ConvertC2F (document.stinky.number.value)"
```

What Happens…

- A function call “makes it happen”…
- Substitution Rule: The function call operates as if the function definition replaces the call and the arguments replace the parameters

```javascript
function ConvertC2F (c) {
  var f;
  f = (9 * c) / 5 + 32;
  return f;
}
```

```javascript
onClick = "answertextbox = ConvertC2F (document.stinky.number.value)"
```

Is the same as:

```javascript
onClick = "answertextbox = 9*(document.stinky.number.value) / 5 +32"
```
Calling the convertC2F Function

```html
<FORM NAME="conversion" ACTION="convertC2F.html">
<H2>Enter the temperature in Celsius:</H2>
<input type="text" name="celsius" value=" ">
<input type="button" name="change" value="Change to Celsius"
    onClick='document.conversion.fahrenheit.value = convertC2F(document.conversion.celsius.value)'>

<H2>That's ?? degrees Fahrenheit</H2>
</FORM>
```

```javascript
function convertC2F(c) {
    var f = (9 * c) / 5 + 32
    return f;
}
```

Function Abstraction

- Whenever the same operations are performed in different places in a program, there is an opportunity for function abstraction.
- Function abstraction gives a name to the operations.
- It also encapsulates the operations so they can be executed out-of-view, receiving input via parameters and returning results or creating effects at the point of the call.

Summary

- Function declarations encapsulate name, parameters and definition.
- Function calls cause the function to be executed.
- Arguments must match formal parameters in number and order.
  - Order matters!
    - The 1st argument corresponds to the 1st formal parameter
    - The 2nd argument corresponds to the 2nd formal parameter
- The Substitution Rule defines how functions work.