**Logic: A Simplified Overview**

Computer logic is an entire topic unto itself, which is really beyond the scope of this class. However, I wanted, at least, to INTRODUCE some of the concepts regarding computer logic to you before we went any further.

Computer logic is not like human logic, it's really more of a routing process, or a mechanization process. In fact, it is comprised of only TWO forms in JavaScript: BRANCH logic, and LOOP logic.

**Branch Logic**

Branch logic allows for simple decisions to be made, based on a CONDITION. A condition can be anything, and is TESTED, returning a value of either true or false (a boolean!).

For example, let's say the CONDITION that I am testing is the color of the sky. Depending on what color the sky is, I am going to take off my hat, or put on my overcoat, or take out my umbrella.

IF the sky is blue, then I am going to take off my hat. IF the sky is gray, then I am putting on my overcoat. IF the sky is dark gray OR black, then I am taking out my umbrella.

I am about to go out of the house, and have come to a decision point. Depending upon the color of the sky, I am going to take different actions. My path, at this point, will BRANCH, depending upon the condition that I am testing (in this case, the color of the sky). This is an example of branching logic: I come to a decision point, and my path may BRANCH in any number of different ways, depending upon the result of the condition test that I am making.

In JavaScript, this sort of logic is expressed using an IF STATEMENT.

Example:

if (theSky == "blue") takeOffMyHat();

First, I state the keyword, if, which makes this an IF STATEMENT, followed by a space. Then I put opening and closing parentheses characters; these parentheses characters are NOT, in this case, a function call operator, but a container for the CONDITION that I am testing. If whatever is being tested inside the parentheses is true, then I will perform whatever action follows the parentheses; if whatever is being tested inside the parentheses is false, then I will go on to the next line of JavaScript code, ignoring the statement following the parentheses.

I may perform MULTIPLE lines of code if my condition is true by using curly-brace characters to define space, much as I do with the function declaration.

Example:

if (theSky == "blue") {

takeOffMyHat();

openTheWindowCurtains();

waveToTheNeighbors();

};

Unlike a function declaration, it is customary to place a semi-colon end-of-line marker at the end of the curly-braces of an if statement.

If I want to BRANCH my decision point, I can use the ELSE statement in combination with the IF statement.

Example:

if (theSky == "blue") {

takeOffMyHat();

} else {

putOnMyGaloshes();

};

The else statement provides a "catch-all" action for anything which does not meet the first condition. Notice how I have NOT placed a semi-colon after the first set of curly-braces and BEFORE the else; you may ONLY place semi-colons AFTER the final curly-brace of the branch, and at the ends of lines of command BETWEEN the curly-braces.

If I want to branch in more than two directions, I can use ELSE IF statements after my initial IF statement.

Example:

if (theSky == "blue") {

takeOffMyHat();

} else if (theSky == "gray") {

putOnMyOvercoat();

} else if (theSky == "black") {

putOnMyOvercoat();

getOutMyUmbrella();

};

Note: I may include a final ELSE statement at the end of my branch if I want a catch-all statement for everything that doesn't meet the previous conditions.

With sequential if/else-if statements in a set, as soon as one of the conditions is met in the branch, that part of the code is executed and the branch is ended. You can't get TWO separate portions of the branch to execute if TWO of the conditions are true; to do that, you need to use TWO separate IF statements rather than an if/else-if branch like the one above.

That's enough about branch logic for this introductory lecture. There are other forms of branch logic, very similar to the if statement, which we do not have the time to cover in this class.