Preboot - Chaining and Nesting

So, we see that we can build complex expressions for a single **IF** statement. Why stop there? We can chain **IF..ELSE** statements with other **IF..ELSE** statements, like this:

```
if(myName == "Martin")
{
    console.log("Hey there Martin, how's it going?");
}
else if(myName == "Beth")
{
    console.log("You look fabulous today!");
}
else
{
    console.log("Greetings Earthling. Have a great day!");
}
```

We can also nest an **IF** statement within another (or within an **ELSE**). This chaining and nesting can go on indefinitely, as needed. Can you decipher the below? When would you walk/fly/swim?

```
if(weather != "rainy")
{
    if(distanceToStadium < 3)
    {
        console.log("I think I'll walk to the game.");
    }
    else
    {
        console.log("It's a bit far, so maybe I'll fly.");
    }
}
else
{
    console.log("Hey, I'm a duck! A little water is OK. I'll swim.");
}
```

If not rainy, and if the distance to the stadium is less than 3 (miles?), then we walk. If not rainy, and if the distance is 3 or more, then we fly. If the weather is rainy (regardless of distance), then we swim. What an odd duck!

Preboot - Complex Conditionals

The expression evaluated by an **IF** statement can be more than a simple comparison. It can perform multiple comparisons, combining or modifying these values with AND, OR and NOT connectors – as long as it eventually produces a Boolean that the **IF** can use to determine *which way to go at the 'Y'*.

In spoken language, we can create compound conditional statements such as *"If it is Friday and I'm in a good mood, then let's go out and have some fun!"* Hence we would not go out if either it is not Friday, or if I'm *not* in a good mood. There are symbols in JavaScript that represent these logical AND, OR and NOT concepts.

The **AND** operator combines two logical tests, requiring **both** inputs to be **true** for the result to be **true**. The symbol for logical AND is a double-**&**, located between the two logical conditions, like this:

```
if(today == "Friday" && moodLevel >= 100)
{
    goDancing();
}
```

The OR is just the flip side of the AND. As conveyed above, we need both expressions to be **true**, but this could also be accurately described by the converse statements – such as *"If it is not Friday, or if I'm not in a good mood, then let's not go out."* The **OR** operator combines two logical tests, evaluating to **true** if **either** input is **true**. Another example might be *"If it is raining or if it is too far to walk, then let's call Uber instead!"* The logical OR is double-|, located between two logical conditions, like this:

```
if(raining == true || distanceMiles > 3)
{
    callUber();
}
```

We run the **callUber()** function unless *both* tests are *not* true (i.e. not raining *and* distance is three or less). So, we have the AND operator and the OR operator. The **NOT** operator inverts a single boolean: what would have been **true** becomes **false**, and what would have been **false** becomes **true**. Logical NOT is an exclamation point !. "If it isn't snowing, I'll wear shorts" could become this code:

```
if(!snowing)
{
    bravelyDonSomeShorts();
}
```

The function would be called only when we enter the **IF** statement, which is when **!snowing** is equal to **true,** which (rephrased) is when **snowing** is equal to **false**. Make sense?

Preboot - Conditionals (Concepts)

If you are driving and reach a fork in the road, you must decide which way to go. Most likely, you will decide based on some very good reason. In code, there is a similar mechanism. **IF** statements look at the value of a variable, or perhaps compare two variables, and then execute certain lines of code if the result is what you expect. If you wish, you can also execute *other* lines of code if the result goes the other way. The important point is that each decision has only two possible outcomes. You have a certain test or comparison to be done; **IF** the test passes, **THEN** you execute certain code. If you wish, you can execute other code in the *"test did not pass"* (**ELSE**) case. This code would look like this:

```
if(myName == "Martin")
{
    console.log("Hey there Martin, how's it going?");
}
```

The **IF** statement is followed by parentheses that contain our *test*. Remember that we need to use a double-equals to create a comparison, and here we compare the value of variable **myName** to the string **"Martin".** If the comparison passes, then we execute the next section of code within the curly braces. Otherwise, we skip that code. What if we want to greet users with a cheerful comment, even if they are not Martin? You can execute other code in this case (called the **ELSE**). We might write code like this:

```
if(myName == "Martin")
{
    console.log("Hey there Martin, how's it going?");
}
else
{
    console.log("Greetings Earthling. Have a great day!");
}
```

Note: the "test" between the **IF**'s parentheses is an *expression* that results in a *Boolean* (a **true** or **false** value). This expression is evaluated when execution reaches the **IF**. If it evaluates to **true**, then your code enters the **IF** statement; if it evaluates to **false**, you enter the **ELSE** (if there is one).

Here is a brief code-based definition of how IF and ELSE statements operate:

```
if (EXPRESSION) // EXPRESSION is evaluated upon reaching this line
{
    // body of 'IF': code runs only if EXPRESSION evaluates to true
}
else
{
    // body of 'ELSE': code runs only if EXPRESSION evaluates to false
}

```