Conditionals: How to Test Whether Specific Conditions are True

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 &lt; 10</td>
<td>true</td>
</tr>
<tr>
<td>7 &gt; 10</td>
<td>false</td>
</tr>
<tr>
<td>7 &lt;= 7</td>
<td>true</td>
</tr>
<tr>
<td>7 &gt;= 7</td>
<td>true</td>
</tr>
<tr>
<td>2 &gt;= 3</td>
<td>false</td>
</tr>
<tr>
<td>2 == 3</td>
<td>false</td>
</tr>
<tr>
<td>2 != 3</td>
<td>true</td>
</tr>
</tbody>
</table>

- `<`  Less than
- `>`  Greater than
- `<=` Less than or Equal
- `>=` Greater than or Equal
- `==` Equal
- `!=` Not Equal
Compound Tests in ActionScript

- Can test more than one condition...

- `<test>  ||  <test>` True if either test is true; false otherwise
- `<test>  &&  <test>` True if both tests are true; false otherwise
- `! <test>` True if the test is false; false if test is true

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 &lt; 10</td>
<td></td>
</tr>
<tr>
<td>7 &lt; 10</td>
<td></td>
</tr>
<tr>
<td>7 &lt; 10 &amp;&amp; 10 &gt; 15</td>
<td>false</td>
</tr>
<tr>
<td>7 &lt; 10 &amp;&amp; !(10 &gt; 15)</td>
<td>true</td>
</tr>
</tbody>
</table>
Practice Compound Tests

```javascript
var n1 = 10;
var n2 = 15;
var n3 = 20;
```

<table>
<thead>
<tr>
<th>Expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>`n1 &gt; n2</td>
<td></td>
</tr>
<tr>
<td>`n2 &gt; n3</td>
<td></td>
</tr>
<tr>
<td><code>n1 &gt;= 0 &amp;&amp; n1 &lt;= 20</code></td>
<td></td>
</tr>
<tr>
<td><code>!(n1 &lt; n3 &amp;&amp; !(n2 &gt; n3))</code></td>
<td></td>
</tr>
</tbody>
</table>
If and If-Else Statements

- **Purpose:** to specify what the program should do given a condition
- For example:

\[
\text{If Statement}
\]

Decide whether or not to execute a set of instructions

\[
\text{if ( … something is true … )}
\]

\[
\{
\text{… execute these lines of code …}
\}
\]
If Statement

- **Purpose:** Specify whether or not to execute a set of instructions based on a condition

**Example:**

```c
// Check if first player has won the game
if (firstPlayerScore > 200)
{
    winner = firstPlayer;
}
```

**General Syntax:**

```
if (< test > )  // If the test is true,
{
    < instructions >  // Instructions are done once; otherwise, not done
}
```
Purpose: Specify an alternative set of instructions to execute when the If-Statement does not hold

```java
if ( ... something is true ... )
{
    ... execute these lines of code ...
}
else
{
    ... otherwise execute these lines of code ...
}
```
If-Else Statement

Example: At the end of a game, determines who has won

```
if ( firstPlayerScore > secondPlayerScore )
{
    winner = firstPlayer;
}
else
{
    winner = secondPlayer;
} // end if-else
```

General Syntax:

```
if ( < test > )    // If the test is true,
{
    < instructions >    // the 1st set of instructions are done once
}
else    // Otherwise 2nd set of instructions are done once
{
    <instructions >    }
```
Nested If-Else Statements

- **Purpose:** Specify multiple alternative sets of instructions to execute when the above IF and IF-ELSE statements do not hold

```plaintext
if ( ... first condition is true ... )
{
    ... execute these first lines of code ...
}
else if ( ... second condition is true ... )
{
    ... execute these second lines of code ...
}
else
{
    ... else execute these lines of code ...
}
```
Nested If-Else Statements

Example:

// At the end of a game, determines who has won
if ( firstPlayerScore > secondPlayerScore )
{
    winner = firstPlayer;
}
else if ( secondPlayerScore > firstPlayerScore )
{
    winner = secondPlayer;
}
else
{
    winner = "No Winner: A Tie";
}
**Boolean Variables**

- **Purpose:** To store whether something is true of false

```javascript
var gameOver: Boolean;  // Creates a Boolean variable
gameOver = false;      // Sets variable to hold value false
    // (At start of game, it’s not over)

...  // Spend time playing the game and then check score
if( p1Score > 30 )
{
    gameOver = true;  // Sets variable to hold value true
}
if( gameOver == true )  // Checks value of Boolean variable
{
    trace( “Game Over!” );
}
```
How are these different?

```javascript
if( p1Score > 30 || p2Score > 30 )
{
    trace( "Game Over!" );
    stop( );
}

if( p1Score > 30 && p2Score > 30 )
{
    trace( "Game Over!" );
    stop( );
}
```
Loops!

- In programming, loops are a way to execute lines of code repeatedly, until a certain condition is false

- There are two types of loops in ActionScript:
  - While Loops
  - For Loops
While Loops

- **Purpose:** Allow instructions to be repeated *while* a certain condition is true
  - The block of code is executed top to bottom until the condition is false
  - When the condition is false, we exit the loop and continue running the rest of code in the program

```// Repeat the set of instructions while something is true
while( ... something is true ... )
{
    ... execute these lines of code ...
}
```
While Loops

Example: var i = 0;
while ( i < 5 )
{
    trace( i );
    i ++;
}

Output:
0
1
2
3
4

General Syntax:
while ( < test > ) // While the test is true, instructions are
{ < instructions > // repeated. Once test is false, repetition stops
}
}
While Loops

Example: Add $1+2+3+4+5$

```
var sum = 0;
var i = 1;
while ( i <= 5 ) {
    sum = sum + i;
    i ++;
}
// end while
trace( sum );
```

Tracing:

<table>
<thead>
<tr>
<th>sum</th>
<th>i</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>6</td>
</tr>
</tbody>
</table>

Nested While Loops

while ( … something is true … )
{
    while ( … something is true … )
    {
        … execute these lines of code …
    }  // end of inner while loop
}  // end of outer while loop
Next Week

- Functions (need for assignment 5)
- Timer Loops (need for assignment 5)
- More on MovieClips