Comparisons and Decisions

JavaScript

If Statement

When our programs need to make decisions, the usual go-to is the “if” statement.

If statements pose a statement that is tested for a true or false result.

If the statement is true, please execute (run) the following code block.

The syntax involves…

- The if keyword.
- Parentheses containing an expression to evaluate as either true or false.
- A code block; they are marked by curly braces “{}”. The lines of code inside are considered to be inside of that code block.

```
» if ( false ) {
    alert( "This code is ignored; evaluated to false!" );
  }
← undefined
» if ( true ) {
    alert( "This code runs; evaluated to true!" );
  }
```

Else Statement

Suppose you need something to happen if the evaluation was true… but if it isn't—you need an alternative set of instructions to run instead! This is where "else" comes in.

When you want one of two sets of instructions to be the only possibilities at that point in the program, you can write two "if" statements.

However, this is more difficult to maintain and harder to read than one of the available alternative solutions. Enter: the "else" statement!

Else is a default set of instructions that takes place only if your "if" condition evaluates to false. Think of it as a fallback.

Note: you cannot add an expression to test in parentheses for an else statement. An else must immediately follow an if code block.

```
» if ( false ) {
    alert( "This code is ignored; evaluated to false!" );
  } else {
    alert( "This code runs; else is the default if the above fails!" );
  }
```

Else If Statement

If you need to decide between more than two possibilities, you can add as many “else if” statements if you’d like between an if code block and an else statement.

“else if” statements are placed in between an if statement and else statement. You can chain many together if you need to!

They use the keywords “else if”, followed by parentheses containing an expression, and a code block with instructions.

Try alternative expressions, and see if you can get different statements to run or be ignored!

```
if ( 3 < 2 ) {
  alert( "3 isn't less than 2, so this won't run!" );
} else if ( 1 > 7 ) {
  alert( "1 isn't greater than 7, so this won't run either!" );
} else {
  alert( "Since the above failed, we'll see this alert message!" );
}
```

Ternary Operator

A ternary operator works much the same as an if/else statement.

The following is the syntax for a ternary operator:

```
Condition ? <expression if true> :
<expression if false>
```

For example:

```
if (myGuess == 1) {

     output(“You guessed right!”)

} else {

     output (“Wrong guess!”)

}
```

Is the same as:

```
myGuess == 1 ? output(“You guessed
right!”) : output(“Wrong guess!”)
```

Switch Statement

The switch statement is used when there are different conditions that will activate different code blocks. To prevent multiple conditions from activating, we must include a break statement. If no case conditions are met, the code under default is activated.

The following is the syntax for a ternary operator:

```
switch (condition) {
     case x:
          //code
          break;
     case y:
          //code
          break;
     default:
          //code
}
```