JavaScript and jQuery Course

Session 03

Control Structures

- block of programming code
- analyzes and makes a decision
- to do something / not do something
- **"if" conditional statement**

"if" Conditional Statement

- Performs an action(s)
- Based on a condition(s)
- Chooses between different possibilities

"if" Conditional Statement

```
if (condition)
{
    code to be executed if condition is true;
}
```

no semicolon

If the condition is true, ***execute*** the code

"if" rules:

- *condition* must be enclosed in ( )
- code to be executed *may* be enclosed in { }
- if more than one statement is to be executed, *must* use { }
  - ➥ it is usually best to use { } even if only 1 statement ➥

"if" Conditional Statement

Used with a comparison operator

==

Compares two things

are they the same - does not mean are they equal

Operators

Relational or Comparison Operators

!= == >= <= > <

Compare / analyze two or more factors

"if" Condition Example

```
var myName = "Teresa";

if (myName == "Teresa")
{
    alert("this is true");
}
```

Opening brace { on next line

Statements inside the { } block **are indented!**

```
var myName = "Teresa";

if (myName == "Teresa") {
    alert("this is true");
}
```

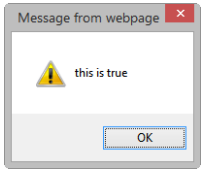
Opening brace { on same line

Closing brace } aligned under the if

"if" Condition Example

```
var myName = "Teresa";

if (myName = "Teresa")
{
    alert("this is true");
}
```

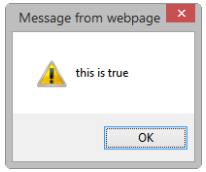
INCORRECT
Assignment will always return true



"if" Condition Example

```
var myName = "Teresa";

if ("Teresa" == myName)
{
    alert("this is true");
}
```

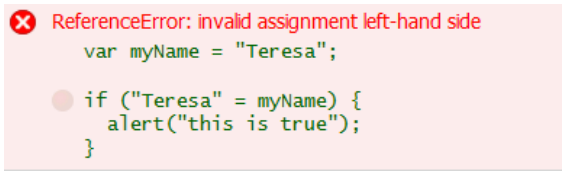
Can put the variable on the right



"if" Condition Example

```
var myName = "Teresa";

if ("Teresa" = myName)
{
    alert("this is true");
}
```

You will get an error message with only one =



"if" Condition Example

```
var myName = "Teresa";

if (myName == "teresa")
{
    alert("this is true");
}
```

- not true
- nothing will happen
- case does not match

JS performs case sensitive comparison

"if" Condition - Quiz

```
var x = 20;

if (x >= 20);
{
    alert("this is true");
}
```

True or False?

True / False – alert returns true / false

True vs False (Boolean data type)

If these are returned, they are considered False:

- False
- 0 (zero)
- Empty string
- Null
- Undefined
- NaN

Logical Operators

Logical Operators are used to create *compound conditions* (2 or more factors)

Operator	Name	Where found on keyboard
&&	AND	Above 7 key (top row)
||	OR	Above \ key (above the Enter key)
!	NOT	Above 1 key (top row)

Logical Operator: NOT

```
var x = 6;
var y = 3;

if ( !(x==y) )
{
    alert("this is true");
}
```

True or False?

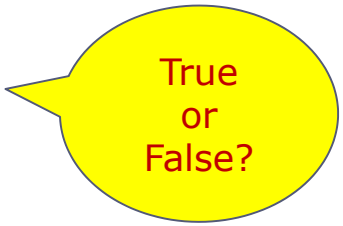
Logical Operator: OR

```
var x = 6;
var y = 3;

if ( x==5 || y==3 )
{
    alert("this is true");
}
```

True or False?

Logical Operator: AND

```
var x = 6;
var y = 3;

if ( x==5 && y==3 )
{
    alert("this is true");
}
```



Compound Condition - Order of Precedence

```
var x = 6;
var y = 3;
var z = 2;

if ( x==5 && y==3 || z==2 )
{
    alert("this is true");
}
```

Question: is this evaluated as

[x AND y] OR z ☜

or

x AND [y OR z]

Order of Precedence - Parentheses

```
var x = 6;
var y = 3;
var z = 2;

if ( (x==5 && y==3) || z==2 )
{
    alert("this is true");
}
```

Evaluated as

[x AND y] OR z

[false AND true] OR true ↙
false OR true ↙
true

Order of Precedence - Parentheses

```
var x = 6;
var y = 3;
var z = 2;

if ( x==5 && (y==3 || z==2) )
{
    alert("this is true");
}
```

Evaluated as

x AND [y OR z]

false AND [true OR true] ↙
false AND true ↙
false

"if / else" Conditional Statement

one condition
2 possible outcomes

```
if (condition)
{
   // code to execute if condition is true
}
else
{
   // code to execute if condition is not true
}
```


"if / else if" Conditional Statement

several conditions
several possible outcomes

```
if (condition1)
{
  // code to execute if condition1 is true
}
else if (condition2)
{
  // code to execute if condition2 is true
}
else
{
  // code to execute if neither is true
}
```

stops at first "true"

"if / else if" brace examples

Opening { starts on next line, closing } aligned under if or else

```
if (condition1)
{
  // code to execute if condition1 is true
}
else if (condition2)
{
  // code to execute if condition2 is true
}
else
{
  // code to execute if neither is true
}
```

Statements inside the { } blocks **are indented!**

"if / else if" brace examples

Opening { starts on same line, closing } aligned under if or else

```
if (condition1) {
  // code to execute if condition1 is true
}
else if (condition2) {
  // code to execute if condition2 is true
}
else {
  // code to execute if neither is true
}
```

Statements inside the { } blocks **are indented!**

"if / else if" brace examples

```
if      (condition1) { // code for condition1 }
else if (condition2) { // code for condition2 }
else                 { // code for else       }
```

If the *code to execute* is a single statement for all of the conditions, you can "stack the code" like this

"switch" Conditional Statement

For Many Conditions

```
switch(condition to evaluate)
{
    case value1:
      // code to execute if case 1 is true
      break;

    case value2:
      // code to execute if case 2 is true
      break;

    case value3:
      // code to execute if case 3 is true
      break;

    default:
      //code to execute if none are true
      break;
}
```

stops at first "true"

Note the indentation style used with a switch/case block

"switch" Conditional Statement

```
var myName = "Teresa";

switch(myName)
{
    case "Maria":
      // code to execute if case 1 is true
      break;
    case "Sue":
      // code to execute if case 2 is true
      break;
    case "Teresa":
      // code to execute if case 3 is true
      break;
    default:
      //code to execute if none are true
      break;
}
```

The opening brace character { can be at the end of the switch statement or on the next line

Closing brace } aligned under the switch

Loops

- Run a block of code over and over again
  - **for**
    - loops through a block of code a specified number of times
  - **while**
    - loops through a block of code while a condition is true

"for" Loop

- Counter variable
- Conditional statement
- Increment
- Code that is executed for each loop

"for" Loop

```
var i;

for (i=0; i<5; i++)
{
    document.write("The number is " + i + "<br>");
}
```

counter condition increment

no semicolon

Code that is executed for each loop

"for" Loop

The opening brace character { can be at the end of the for statement or on the next line

```
for ( var i=0; i<5; i++ )
{
    document.write("The number is " + i + "<br>");
}
```

Closing brace } aligned under the for

Statements inside the { } block **are indented!**

"while" Loop

- Conditional statement – must be true at least once
- Code that is executed for each loop
- Increment at the bottom

Note: if the condition statement is always True, then you will never exit the loop!

while Loop

stops when the condition is false

```
var i = 0;
while (i < 5)
{
    document.write("The number is " + i + "<br>");
    i++;
}
```

condition

increment

The opening brace character { can be at the end of the while statement or on the next line

Closing brace } aligned under the while

Statements inside the { } block **are indented!**

"do while" Loop

Always runs at least once

```
var i = 0;
do
{
    document.write("The number is " + i + "<br>");
    i++;
}
while (i < 5);
```

condition

increment

The opening brace character { can be at the end of the do statement or on the next line

Closing brace } aligned under the do

Statements inside the { } block **are indented!**

Ternery operator

```
var a = 5;
var b = 3;

if (a > b)
{
  alert("true");
}
else
{
  alert("false");
}
```

```
var a = 5;
var b = 3;

(a > b) ? alert("true") : alert("false");
```

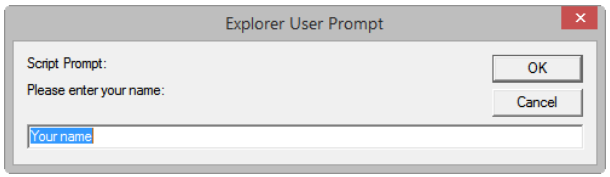
condition

Execute if true

Execute if false

Prompt – method of the window object

```
var theAnswer = prompt("Please enter your name:", "Your name");

alert(theAnswer);
```



The first argument is shown above the text box

The second argument is optional. If provided, it is displayed in the text box

Using the Document Object Model - DOM

```
window.onload = function() {   // begin function
    // code that references objects on the page
}   // end function
```

The anonymous function runs after the objects on the page are loaded

Using the Document Object Model - DOM

```
var theReply = "Hello World";

window.onload = function() { // begin function
    document.getElementById("xyz").innerHTML = theReply;
} // end function

<body>
    <p id="xyz">Hello World</p>
```