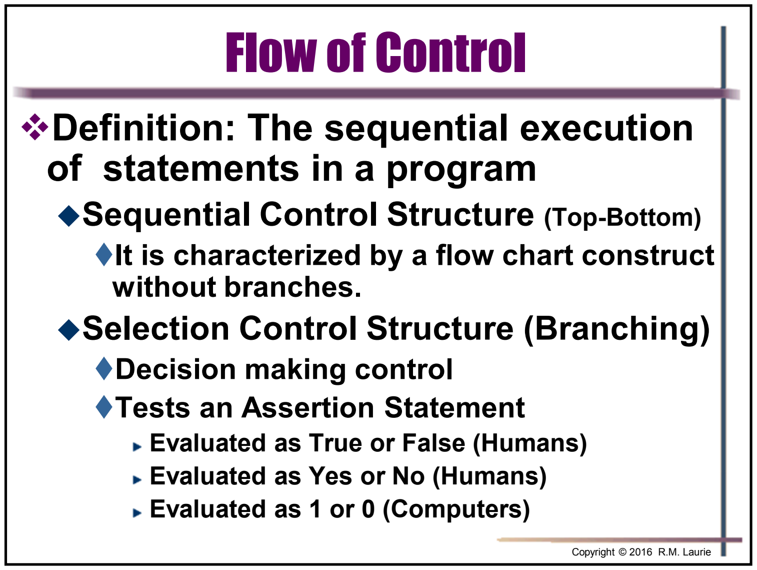
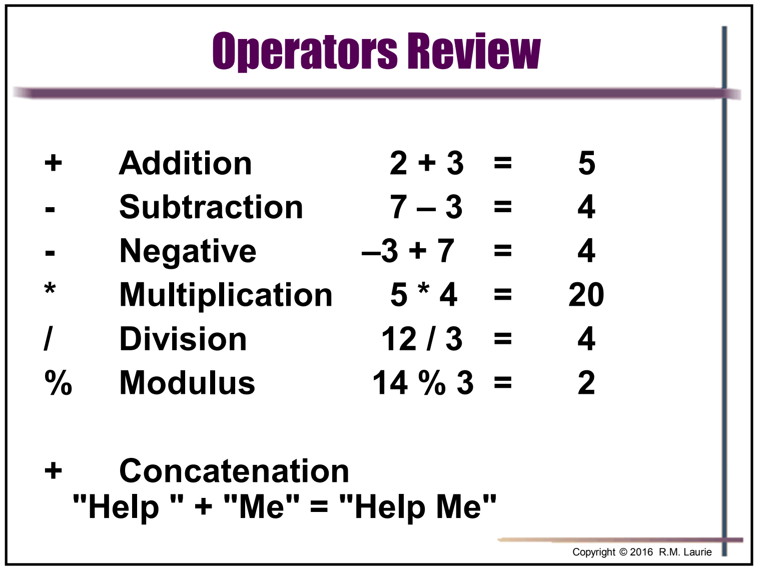


Flow of Control

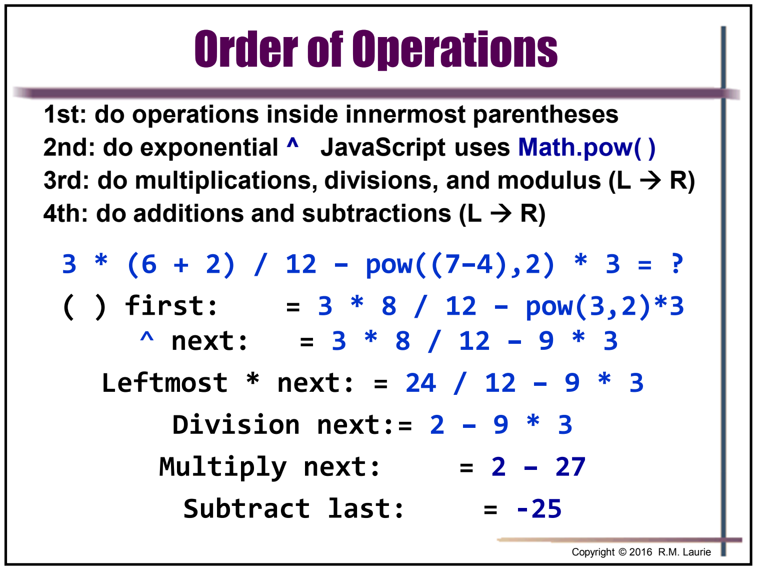
- ❖Definition: The sequential execution of statements in a program
  - ◆Sequential Control Structure (Top-Bottom)
    - ♦It is characterized by a flow chart construct without branches.
  - ◆Selection Control Structure (Branching)
    - ♦Decision making control
    - ♦Tests an Assertion Statement
      - ▸ Evaluated as True or False (Humans)
      - ▸ Evaluated as Yes or No (Humans)
      - ▸ Evaluated as 1 or 0 (Computers)



Operators Review

+	Addition	2 + 3	=	5
-	Subtraction	7 – 3	=	4
-	Negative	–3 + 7	=	4
*	Multiplication	5 * 4	=	20
/	Division	12 / 3	=	4
%	Modulus	14 % 3	=	2

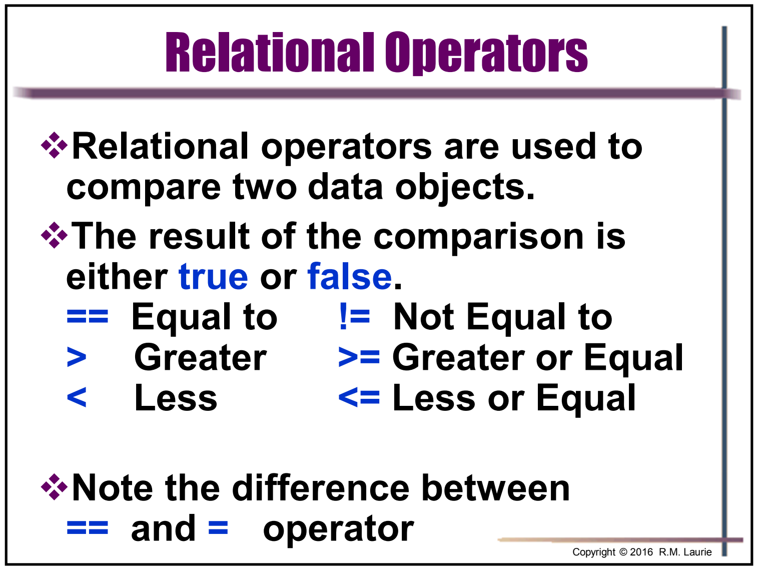
+ Concatenation
"Help " + "Me" = "Help Me"



Order of Operations

1st: do operations inside innermost parentheses
2nd: do exponential ^ JavaScript uses Math.pow( )
3rd: do multiplications, divisions, and modulus (L → R)
4th: do additions and subtractions (L → R)

```
3 * (6 + 2) / 12 – pow((7-4),2) * 3 = ?
( ) first:     = 3 * 8 / 12 – pow(3,2)*3
     ^ next:    = 3 * 8 / 12 – 9 * 3
   Leftmost * next: = 24 / 12 – 9 * 3
       Division next:= 2 – 9 * 3
     Multiply next:     = 2 – 27
      Subtract last:     = -25
```



Relational Operators

- ❖Relational operators are used to compare two data objects.
- ❖The result of the comparison is either true or false.

==	Equal to	!=	Not Equal to
>	Greater	>=	Greater or Equal
<	Less	<=	Less or Equal

- ❖Note the difference between == and = operator

Relational Operator Examples

- **Given:** A = 23, B = 16, Entry = 'y'
- **Then:**

```
A > B is true
A < B is false
A >= B is true
A <= B is false
A != B is true
A == B is false
(A < 5) && (B > 10) is false
(Entry=='y') || (Entry=='Y') is true
```

Logical Operators

- **Used in while and if assertions true/false**
- **There are three logical operators**
  - **AND** &&
  - **OR** ||
  - **NOT** !

A	B	A && B
F	F	F
F	T	F
T	F	F
T	T	T

A	B	A || B
F	F	F
F	T	T
T	F	T
T	T	T

A	!A
F	T
T	F

Note on Precedence: Evaluate relational first and then logical

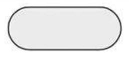
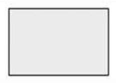
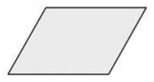
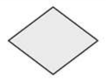
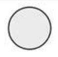
Arithmetic Operators Precedence

(Highest to Lowest)

( )	Defines order of operation
-	Minus (unary)
* / %	Multiply, Division, Remainder
+ -	Addition, Subtraction
< <= > >=	Relational Operators
== !=	Relational Operators
&& || !	Logical Operators
=	Assignment

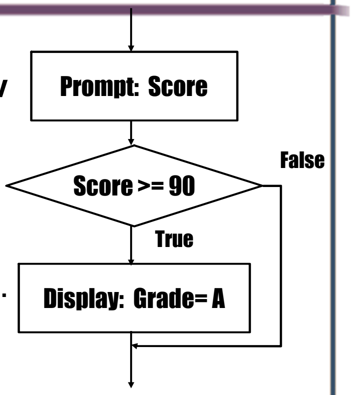
http://www.w3schools.com/jsref/jsref_operators.asp

Flowchart Symbols

Symbol	Name	Description
	Terminator	Represents the start or end of a program or module
	Process	Represents any kind of processing function; for example, a computation
	Input/output	Represents an input or output operation
	Decision	Represents a program branch point
	Connector	Indicates an entry to, or exit from, a program segment

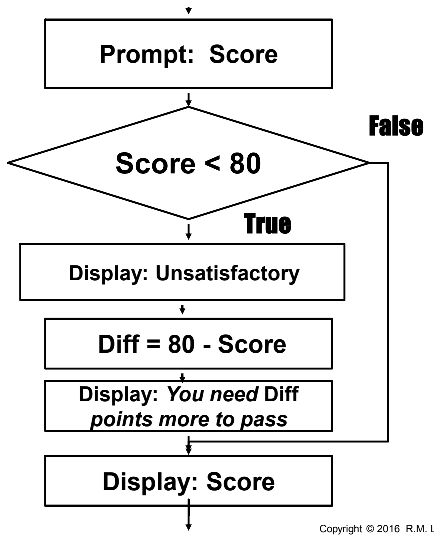
if Selection Control Structure

❖Characterized by a diamond shaped flow chart construct, containing an assertions with two possible outcomes branches (True or False).



```
if(Score >= 90)
     document.write("Grade = A");
```

if Compound Selection Control Structure



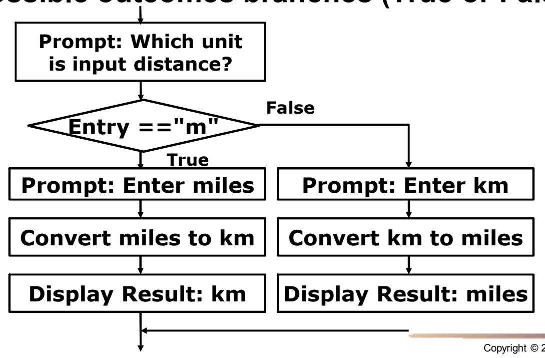
if Selection Control Structure

(Compound statement syntax)

```
var fScore=parseFloat(window.prompt("Enter Score","0" ));
if(fScore < 80)
{
    document.write("<h2 style=\"color: #CC0000\">"
        + "Exam Result Unsatisfactory</h2>");
    var fDiff = 80 - fScore;
    document.write("<p>You need " + fDiff
        + " more points to continue to next chapter</p>");
}
document.write("<p>Your Exam Score was " + fScore
    + "</p>");
```

if - else Selection Structure

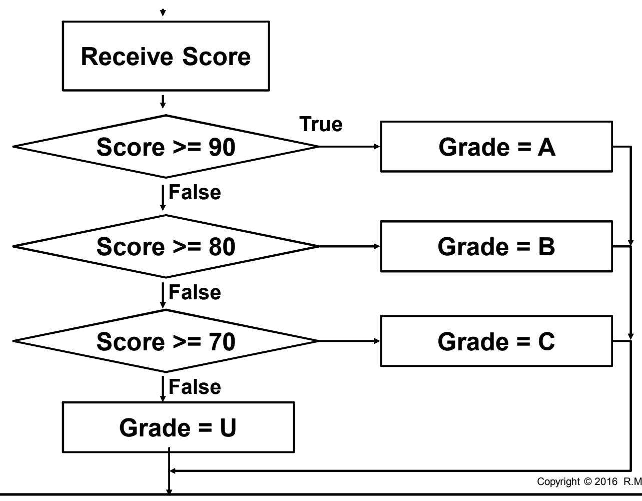
❖Characterized by a diamond shaped flow chart construct, containing an assertions with two possible outcomes branches (True or False).



if - else Selection Structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Miles or Kilometers Converter</title>
      <script type="text/javascript">
      var sEntry, fEntry, fResult;
      sEntry = window.prompt("Is input distance miles or km? (m or k)","m");
      if(sEntry == "m")
      {
            fEntry = parseFloat(window.prompt("Enter miles: ", "0"));
            fResult = Entry * 1.609;
            document.write("<p>"+fEntry+" miles = "+fResult+" km</p>");
      }
      else
      {
            fEntry = parseFloat(window.prompt("Enter kilometers: ", "0"));
            fResult = fEntry / 1.609;
            document.write("<p>"+ fEntry +" km = "+ fResult +" miles</p>");
      }
      document.write("<p>Reload for another conversion</p>");
      </script>
  </head>
  <body> </body>
</html>
```

if - else if - else Structure



if - else if – else Selection Structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Grade Determination</title>
    <script  type="text/javascript">
      var fScore, cGrade;
      fScore = parseFloat(window.prompt( "Enter Score", "0" ));
      if(fScore >= 90)
        cGrade = "A";
      else if(fScore >= 80)
        cGrade = "B";
      else if(fScore >= 70)
        cGrade = "C";
      else
        cGrade = "U";
      document.write("<h2>For the score = " + fScore
       + " <br>Your letter grade is " + cGrade + "</h2>" );
    </script>
  </head>
  <body></body>
</html>
```

Program Style Practices

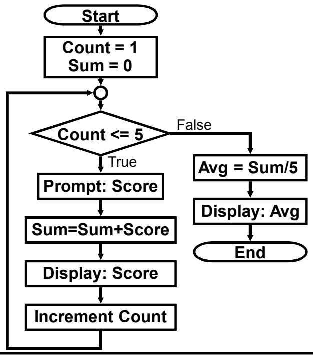
- Write structured and modular programs
  - Use descriptive variable names
  - Provide a welcome message for the user
  - Use a prompt before all input
  - Provide well designed program output
  - Document programs using comments
- Test your program thoroughly
  - Write test data to test all selection paths
  - Does output support user expectations

Flow of Control

- Definition: The sequential execution of statements in a program
  - Sequential Control Structure (Top-Bottom)
  - Selection Control Structure (Decisions)
  - Repetition Control Structure (Looping)
    - Loop back and repeats code execution
    - Relational and Logical Operators
    - Tests an Assertion (T/F) to loop again or exit
    - Counter controlled or Sentinel controlled loops
    - Keywords: while do while for
    - Computers Never Get Bored
      - Best for iterative well structured processing
      - Not well suited for creative problem solving

Repetition (Loop) Structure

- Repeat a sequence of instructions in a loop
- The simplest loop structure is the while( )
- Beware of infinite loops, exit must occur!



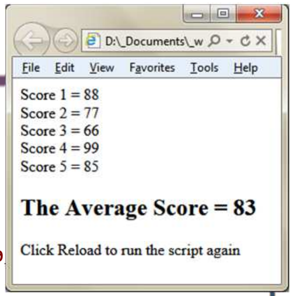
```
var fScore, nCount = 1, fSum = 0;
while(nCount <= 5)
{
  fScore = parseFloat(window.prompt(
    "Enter Score "+ nCount,"0"));
  fSum = fSum + fScore;
  document.write("<p>Score "+nCount
     +" = " + fScore+"</p>");
   nCount = nCount + 1;
}
document.write("<h3>Average = "
+(fSum/5) + "</h3>");
```

while statement loop control

- Contents of loop executed repeatedly while(*assertion)* is true
- Loop terminated when while(assertion) is false
- Counter-Controlled Repetition Structure
  - Initialize a counter to count loops
  - Increment or decrement counter
  - while(assertion) checks for total loops reached
- Sentinel-Controlled Repetition Structure
  - while(assertion) checks for a sentinel termination value

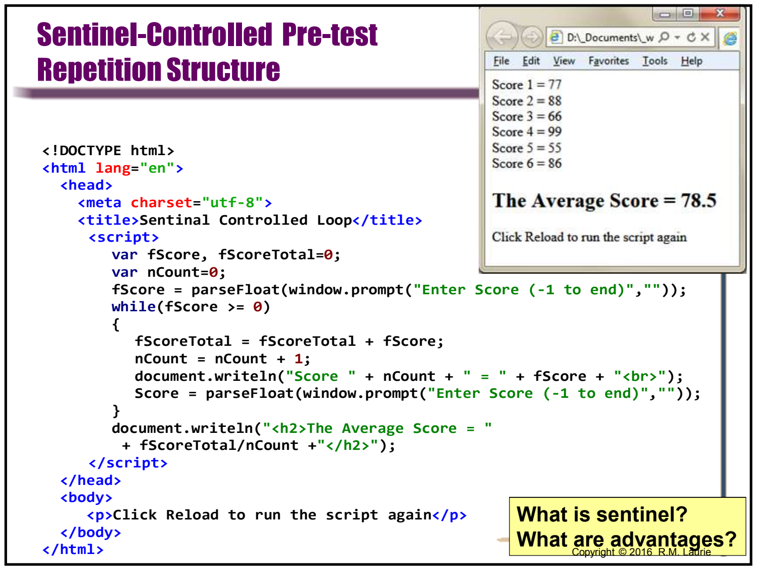
Counter-Controlled Pre-test Repetition Structure

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Counter Controlled Loop</title>
    <script>
        var nScore=0, nScoreTotal=0, nCount=0
        while(nCount < 5)
        {
           nScore=parseInt(window.prompt("Enter Score",""));
           nScoreTotal = nScoreTotal + nScore;
           nCount = nCount + 1;
           document.write("Score " + nCount + " = " + nScore + "<br>");
        }
        document.write("<h2>The Average Score = "+ nScoreTotal/5 +"</h2>");
    </script>
  </head>
  <body>
     <p>Click Reload to run the script again</p>
  </body>
</html>
```

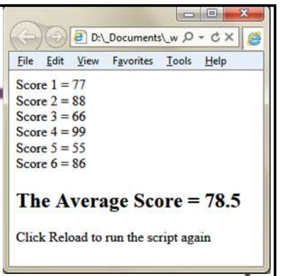


```
Score 1 = 88
Score 2 = 77
Score 3 = 66
Score 4 = 99
Score 5 = 85

The Average Score = 83

Click Reload to run the script again
```

1. Define counter
2. Initialize counter
3. Increment counter



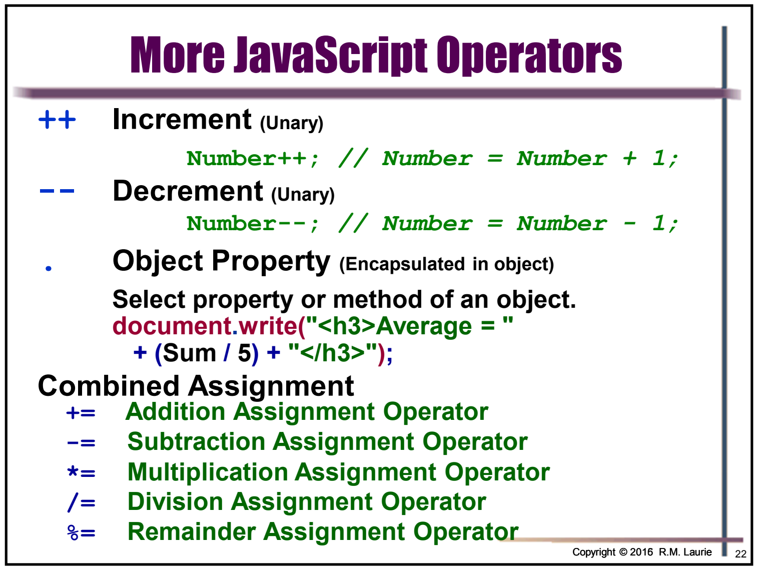
Sentinel-Controlled Pre-test Repetition Structure



```
Score 1 = 77
Score 2 = 88
Score 3 = 66
Score 4 = 99
Score 5 = 55
Score 6 = 86

The Average Score = 78.5

Click Reload to run the script again
```

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Sentinal Controlled Loop</title>
     <script>
        var fScore, fScoreTotal=0;
        var nCount=0;
        fScore = parseFloat(window.prompt("Enter Score (-1 to end)",""));
        while(fScore >= 0)
        {
           fScoreTotal = fScoreTotal + fScore;
           nCount = nCount + 1;
           document.writeln("Score " + nCount + " = " + fScore + "<br>");
           Score = parseFloat(window.prompt("Enter Score (-1 to end)",""));
        }
        document.writeln("<h2>The Average Score = "
         + fScoreTotal/nCount +"</h2>");
     </script>
  </head>
  <body>
     <p>Click Reload to run the script again</p>
  </body>
</html>
```

What is sentinel?
What are advantages?



More JavaScript Operators

++ Increment (Unary)

```
Number++; // Number = Number + 1;
```

-- Decrement (Unary)

```
Number--; // Number = Number - 1;
```

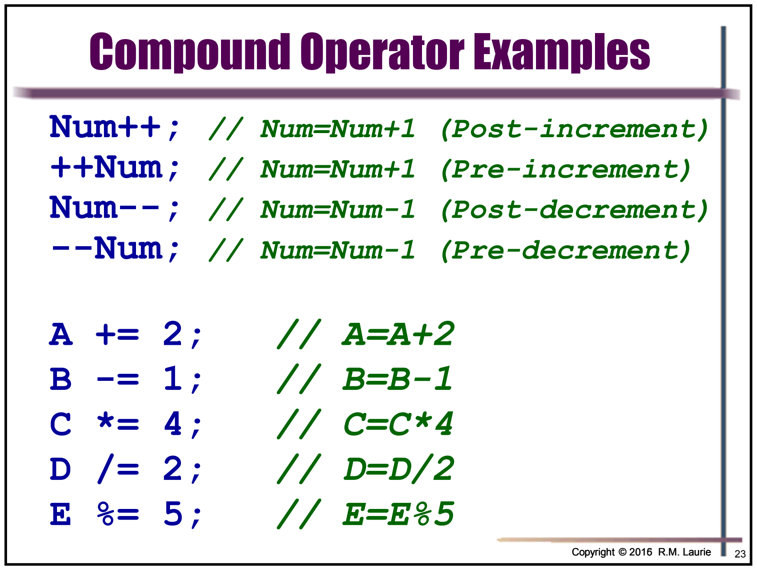
. Object Property (Encapsulated in object)

Select property or method of an object.
document.write("<h3>Average = "
+ (Sum / 5) + "</h3>");

Combined Assignment

+=	Addition Assignment Operator
-=	Subtraction Assignment Operator
*=	Multiplication Assignment Operator
/=	Division Assignment Operator
%=	Remainder Assignment Operator

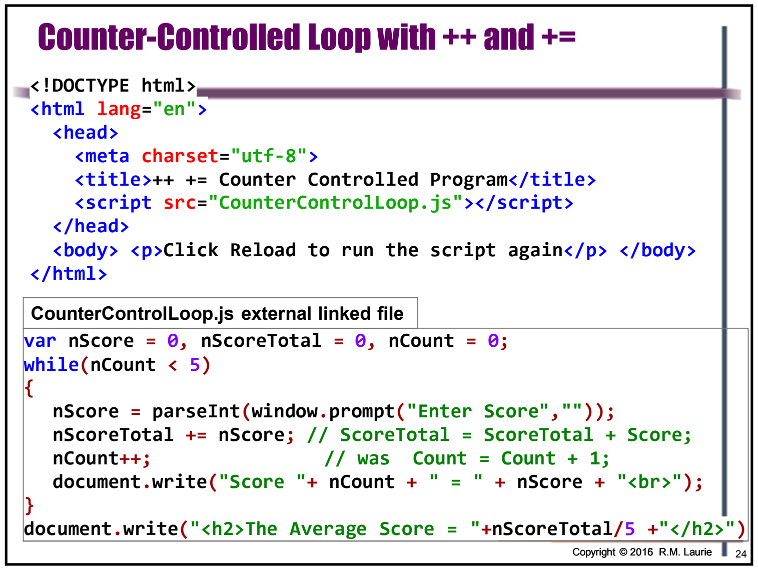
22



Compound Operator Examples

```
Num++;  // Num=Num+1  (Post-increment)
++Num;  // Num=Num+1  (Pre-increment)
Num--;  // Num=Num-1  (Post-decrement)
--Num;  // Num=Num-1  (Pre-decrement)

A += 2;    // A=A+2
B -= 1;    // B=B-1
C *= 4;    // C=C*4
D /= 2;    // D=D/2
E %= 5;    // E=E%5
```

23



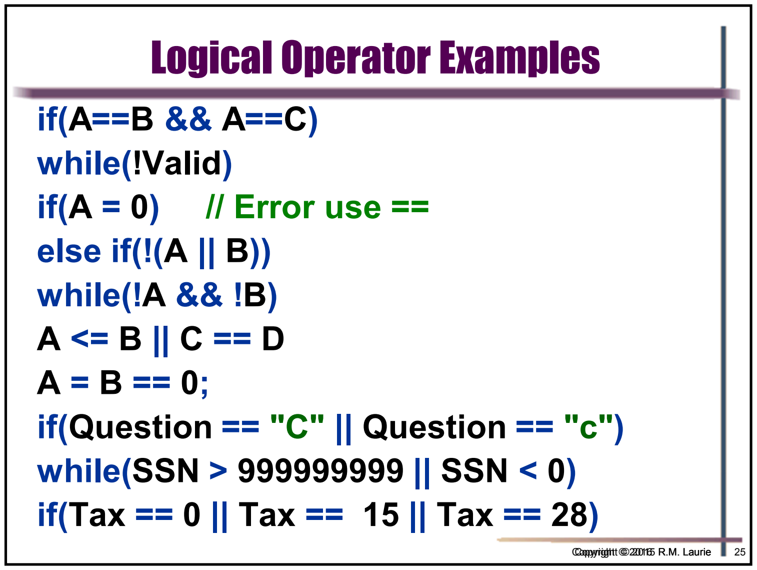
Counter-Controlled Loop with ++ and +=

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>++ += Counter Controlled Program</title>
    <script src="CounterControlLoop.js"></script>
  </head>
  <body> <p>Click Reload to run the script again</p> </body>
</html>
```

CounterControlLoop.js external linked file

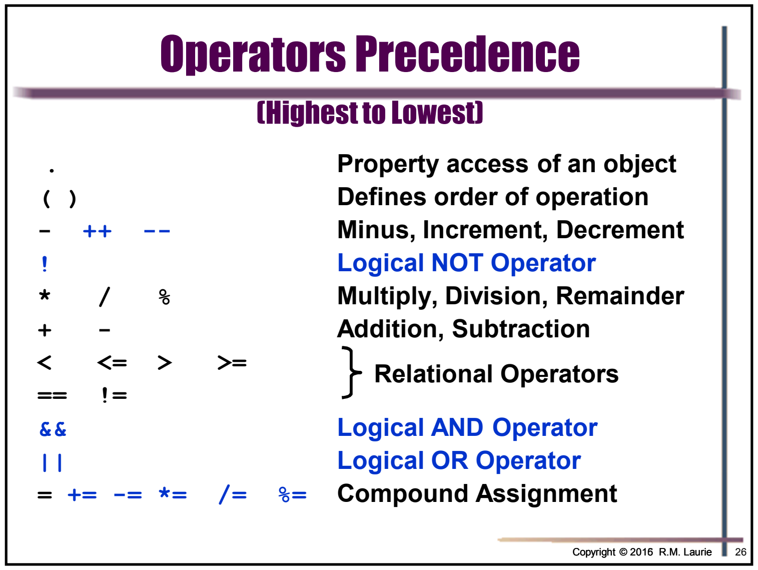
```
var nScore = 0, nScoreTotal = 0, nCount = 0;
while(nCount < 5)
{
  nScore = parseInt(window.prompt("Enter Score",""));
  nScoreTotal += nScore; // ScoreTotal = ScoreTotal + Score;
  nCount++;                    // was  Count = Count + 1;
  document.write("Score "+ nCount + " = " + nScore + "<br>");
}
document.write("<h2>The Average Score = "+nScoreTotal/5 +"</h2>")
```

24



Logical Operator Examples

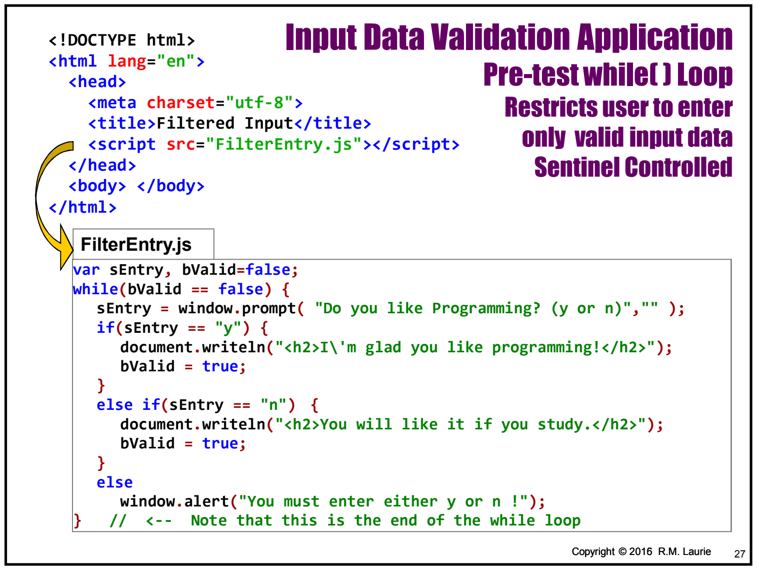
```
if(A==B && A==C)
while(!Valid)
if(A = 0)     // Error use ==
else if(!(A || B))
while(!A && !B)
A <= B || C == D
A = B == 0;
if(Question == "C" || Question == "c")
while(SSN > 999999999 || SSN < 0)
if(Tax == 0 || Tax ==  15 || Tax == 28)
```



Operators Precedence

(Highest to Lowest)

`.`	Property access of an object
`( )`	Defines order of operation
`-  ++  --`	Minus, Increment, Decrement
`!`	Logical NOT Operator
`*  /  %`	Multiply, Division, Remainder
`+  -`	Addition, Subtraction
`<  <=  >  >=`	} Relational Operators
`==  !=`	
`&&`	Logical AND Operator
`||`	Logical OR Operator
`=  +=  -=  *=  /=  %=`	Compound Assignment



Input Data Validation Application

Pre-test while( ) Loop

Restricts user to enter only valid input data

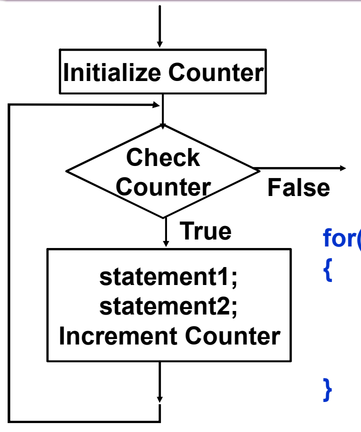
Sentinel Controlled

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Filtered Input</title>
    <script src="FilterEntry.js"></script>
  </head>
  <body> </body>
</html>
```

FilterEntry.js

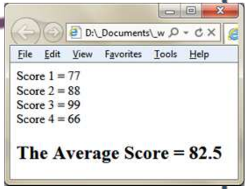
```
var sEntry, bValid=false;
while(bValid == false) {
   sEntry = window.prompt( "Do you like Programming? (y or n)","" );
   if(sEntry == "y") {
      document.writeln("<h2>I\'m glad you like programming!</h2>");
      bValid = true;
   }
   else if(sEntry == "n")  {
      document.writeln("<h2>You will like it if you study.</h2>");
      bValid = true;
   }
   else
      window.alert("You must enter either y or n !");
}   //  <--  Note that this is the end of the while loop
```

for loop Flow Chart



```
for(Initialize; Test; Increment)
{
      statement1;
      statement2;
      statement3;
}
```

For( ) Counter Controlled Loop Example

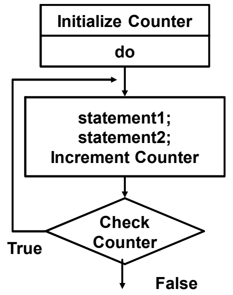


```
Score 1 = 77
Score 2 = 88
Score 3 = 99
Score 4 = 66

The Average Score = 82.5
```

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Average Calculation 2</title>
      <script>
         var nScore, nScoreTotal = 0, nCount, nQty;
         nQty = parseInt(window.prompt("How Many Scores?",""));
         for(nCount = 1; nCount <= nQty; nCount++)
         {
            nScore = parseInt(window.prompt("Enter Score",""));
            nScoreTotal = nScoreTotal + nScore;
            document.write("Score " + nCount + " = "
               + nScore + "<br/>");
         }
         document.write("<h2>The Average Score = "
            + (nScoreTotal / nQty) + "</h2>");
     </script>
   </head>
   <body>
   </body>
</html>
```

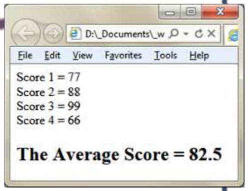
do - while Post-test Structure

- A loop structure that guarantees the loop body is executed once.
- Condition is tested at bottom of loop
- Don't forget the semicolon for while(...);



```
Initialize Counter;
do
{
        statement1;
        statement2;
        Increment Counter;
}while(Check Counter);
```

Sentinel Controlled Loop Example



```
Score 1 = 77
Score 2 = 88
Score 3 = 99
Score 4 = 66

The Average Score = 82.5
```

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <meta charset="utf-8">
      <title>Average Calculation 2</title>
      <script>
         var nScore, nCount=0, nTotal = 0;
         do
         {
           nScore = parseInt(window.prompt("Enter Score or [Q]=quit","Q"));
            if(isNaN(nScore));  // Score is Not a Number
            else if(nScore < 0)
               window.alert("Score cannot be negative");
            else
            {
               nTotal += nScore;
               nCount++;
               document.write("<p>Score " + nCount+" = " + nScore + "</p>");
            }
         }while(!isNaN(nScore));
         document.write("<h2>Average Score = " + nTotal/nCount + "</h2>");
     </script>
   </head>
   <body>
   </body>
</html>
```

http://www.w3schools.com/jsref/jsref_obj_global.asp