INTRODUCTION TO JAVASCRIPT

Dr. S.V.Shirbhate
Assistant Professor
DCPE HVPM Amravati

Why Study JavaScript?

- 1. HTML to define the content of web pages

  2. CSS to specify the layout of web pages

  3. JavaScript to program the behavior of web pages

JAVASCRIPT

- **JavaScript is used in millions of Web pages to improve the design, validate forms, detect browsers, create cookies, and much more.**
- **JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Mozilla, Firefox, Netscape, Opera.**

WHAT IS JAVASCRIPT?

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language (a scripting language is a lightweight programming language)
- A JavaScript consists of lines of executable computer code
- A JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)
- Everyone can use JavaScript without purchasing a license

Are Java and JavaScript the Same?

- NO!
- Java and JavaScript are two completely different languages in both concept and design!
- Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

How to Put a JavaScript Into an HTML Page?

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!")
</script>
</body>
</html>
```

Ending Statements With a Semicolon?

- With traditional programming languages, like C++ and Java, each code statement has to end with a semicolon (;).
- Many programmers continue this habit when writing JavaScript, but in general, semicolons are **optional**! However, semicolons are required if you want to put more than one statement on a single line.

JavaScript Variables

- Variables are used to store data.
- A variable is a "container" for information you want to store. A variable's value can change during the script. You can refer to a variable by name to see its value or to change its value.
- Not use any javascript reserved words
- Rules for variable names:
  - Variable names are case sensitive
  - They must begin with a letter or the underscore character not start with a numerical (0-9) e.g 123test is invalid and _123test is valid
    - strname – STRNAME (not same)

JavaScript Data Types

JavaScript variables can hold many data types: numbers, strings, objects and more

for e.g.

```
var length = 16;                // Number
var lastName = "Johnson";       // String
var x = {firstName:"John", lastName:"Doe"};
// Object
```

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

```
var x = 16 + "Volvo";
```

Note:When adding a number and a string, JavaScript will treat the number as a string.

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

```
var x = 16 + 4 + "Volvo";
```

```
var x = "Volvo" + 16 + 4;
```

JavaScript Types are Dynamic

```
var x;              // Now x is undefined
x = 5;              // Now x is a Number
x = "John";         // Now x is a String
```

Javascript Operators

In Javascript there are following types of operators

- Arithmetic Operators
- Assignment Operators
- String Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Type Operators

JavaScript Arithmatic Operators

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation (ES2016)
/	Division
%	Modulus (Remainder)
++	Increment
--	Decrement

Operator	Description	Example	Result
+	Addition	x=2 y=2 x+y	4
-	Subtraction	x=5 y=2 x-y	3
*	Multiplication	x=5 y=4 x*y	20
/	Division	15/5 5/2	3 2.5
%	Modulus (division remainder)	5%2 10%8 10%2	1 2 0
++	Increment	x=5 x++	x=6
--	Decrement	x=5 x--	x=4

JavaScript AssignmentOperators – 2

Assignment Operators

```
<!DOCTYPE html>
<html>
<body>
<h2>The += Operator</h2>
<script>
var x = 10;
x += 6;
document.write(x) ;
</script>
</body>
</html>
```

Operator	Example	Is The Same As
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

Javascript String Operator

The + operator can also be used to add (concatenate) strings.

```
<!DOCTYPE html>
<html>
<body>
<h2>JavaScript Operators</h2>
<p>The + operator concatenates (adds) strings.</p>
<script>
var txt1 = "John";
var txt2 = "Doe";
document.write(txt1+txt2);
</script>
</body>
</html>
```

JavaScript Comparison Operators

Comparison Operators

Operator	Description	Example
==	is equal to	5==8 returns false
===	is equal to (checks for both value and type)	x=5 y="5" x==y returns true x===y returns false
!=	is not equal	5!=8 returns true
>	is greater than	5>8 returns false
<	is less than	5<8 returns true
>=	is greater than or equal to	5>=8 returns false
<=	is less than or equal to	5<=8 returns true

JavaScript Logical Operators

Logical Operators

Operator	Description	Example
&&	and	x=6 y=3 (x < 10 && y > 1) returns true
||	or	x=6 y=3 (x==5 || y==5) returns false
!	not	x=6 y=3 !(x==y) returns true

JavaScript Type Operators

Operator	Description
typeof	Returns the type of a variable
instanceof	Returns true if an object is an instance of an object type

JavaScript Bitwise Operators

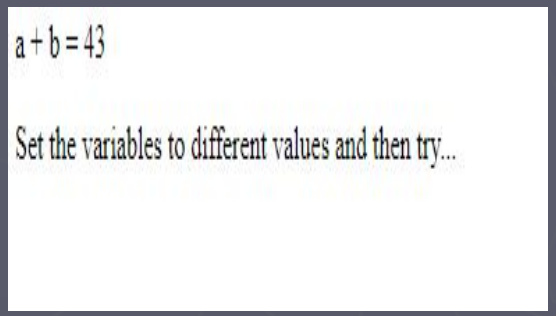
Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	Zero fill left shift	5 << 1	0101 << 1	1010	10
>>	Signed right shift	5 >> 1	0101 >> 1	0010	2
>>>	Zero fill right shift	5 >>> 1	0101 >>> 1	0010	2

Example

```
<html>
<body>
<script type="text/javascript">
var a = 33;
var b = 10;
var c = "Test";
var linebreak = "<br />";
document.write("a + b = ");
result = a + b;
document.write(result);
document.write(linebreak);
</script>
<p>Set the variables to different values and then try...</p>
</body>
</html>
```



JavaScript Basic Examples

```
<script>
document.write("Hello World!")
</script>
```
⇒ format text with HTML code - heading

```
<script>
alert("Hello World!")
</script>
```

Example

```
<script>
x="Hello World!"
document.write(x)
</script>

<script>
x="Shirbhate"
document.write("Sandhya" +x)
</script>
```

⇒ use line break html code

pop up boxes in JavaScript

In Javascript, popup boxes are used to display the message or notification to the user. There are three types of pop up boxes in JavaScript namely

- Alert Box
- Confirm Box and
- Prompt Box.

JavaScript Alert Box

- Alert Box
  - An alert box is often used if you want to make sure information comes through to the user.
  - When an alert box pops up, the user will have to click "OK" to proceed.

```
<script>
alert("Hello World!")
</script>
```

JavaScript Confirm Box

- Confirm Box
  - A confirm box is often used if you want the user to verify or accept something.
  - When a confirm box pops up, the user will have to click either "OK" or "Cancel" to proceed.
  - If the user clicks "OK", the box returns true. If the user clicks "Cancel", the box returns false.

example of Confirm box

```
<html>
<body>
<script>
if (confirm("Press a button!")) {
  txt = "You pressed OK!";
} else {
  txt = "You pressed Cancel!";
}
</script>
</body>
</html>
```

JavaScript Prompt Box

- Prompt Box
  - A prompt box is often used if you want the user to input a value before entering a page.
  - When a prompt box pops up, the user will have to click either "OK" or "Cancel" to proceed after entering an input value.
  - If the user clicks "OK", the box returns the input value. If the user clicks "Cancel", the box returns null.

Prompt Box Example

```
<html>
<body>
<script>
x=prompt("Enter your Name", "");
document.write(x+"  " + "Shirbhate");
</script>
</body>
</html>
```

Prompt Box example

```
<html>
<body>
<script>
//y="<br/>"
x=prompt("Enter value of x", "");
y=prompt("Enter value of y", "");
result= Number(x)+Number(y);
document.write("addition of two nos is="+result);
document.write("<br/>");
result= Number(x)*Number(y);
document.write("multiplication is"+result);
</script>
</body>
</html>
```

JS Examples -1

Y=20x+12

```
<script>
x=3
y=20*x+12
alert(y)
</script>
```

Examples -2

```
<script>
s1=12
s2=28
add=s1+s2
document.write("Addition of two nos:"+add)
</script>
```

Examples -3

s1=12, s2=28

```
<script>
s1=12
s2=28
Add=s1+s2
sub=s1-s2
mult=s1*s2
div=s1/s2
document.write("<br>Demo of Arithmatic operator..<br>")
document.write("<br>Addition of two nos: "+Add)
document.write("<br>Subtraction of two Nos: "+sub)
document.write("<br>multiplication of two Nos: "+mult)
document.write("<br>division of two nos: "+div)
alert("Arithmatic operator")
</script >
```

Conditional Statements

Very often when you write code, you want to perform different actions for different decisions. You can use conditional statements in your code to do this.

In JavaScript we have the following conditional statements:

- **if statement** - use this statement if you want to execute some code only if a specified condition is true
- **if...else statement** - use this statement if you want to execute some code if the condition is true and another code if the condition is false
- **if...else if....else statement** - use this statement if you want to select one of many blocks of code to be executed
- **switch statement** - use this statement if you want to select one of many blocks of code to be executed

Conditional Statements - 2

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

---

if (*condition*)
{
code to be executed if condition is true
}
else
{
code to be executed if condition is not true
}

Conditional Statements Examples

```
<html>
<body>
<script>
c=prompt(" Enter the number", "")
c= Number(c)
if(c>0)
alert(" The Number is positive")
document.write("Given Number is ="+c)
</script>
</body>
</html>
```

Conditional Statements Examples - 2

```
<html>
<body>
<script>
c=prompt(" Enter your age", "")
c= Number(c)
document.write("Your age is ="+c)
if(c>18)
alert(" You are elligible for Voting")
else
alert(" You are not elligible for Voting")
</script>
</body>
</html>
```

syntax of if...else if....else statement

```
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

Example

```
<html>
<body>
<script>
time= prompt( "enter the time","");
time=Number(time);
if(time< 12)
document.write(" Good Morning")
else if(time< 17)
document.write("Good Afternoon")
else if(time< 20)
document.write("Good Evening")
else
document.write("Good Night")
</script>
</body>
</html>
```

The JavaScript Switch Statement

Use the switch statement to select one of many code blocks to be executed. Syntax

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

example

```
<!DOCTYPE html>
<html>
<body>
<script>
day=prompt("enter Day values shouldbe 0,1,...6","");
day= Number(day);
switch (day) {
  case 0:
    day = "Sunday";
    break;
  case 1:
    day = "Monday";
    break;
  case 2:
    day = "Tuesday";
    break;
```

Example Continue

```
 case 3:
    day = "Wednesday";
    break;
  case 4:
    day = "Thursday";
    break;
  case 5:
    day = "Friday";
    break;
  case  6:
    day = "Saturday";
default:
day= "wrong data"
}
document.write(day);
</script>
</body>
</html>
```

Loops in JavaScript

- Looping in programming languages is a feature which facilitates the execution of a set of instructions/functions repeatedly while some condition evaluates to true.
- For example, suppose we want to print “Hello World” 10 times. This can be done in two ways as shown below:
- Iterative Method
- Iterative method to do this is to write the document.write() statement 10 times.

Using Loops

- Using Loops
- In Loop, the statement needs to be written only once and the loop will be executed 10 times as shown below:

```
<script type = "text/javascript">
var i;
for (i = 0; i < 10; i++)
{     document.write("Hello World!\n");
}
< /script>
```

Types of loops

There are mainly two types of loops:

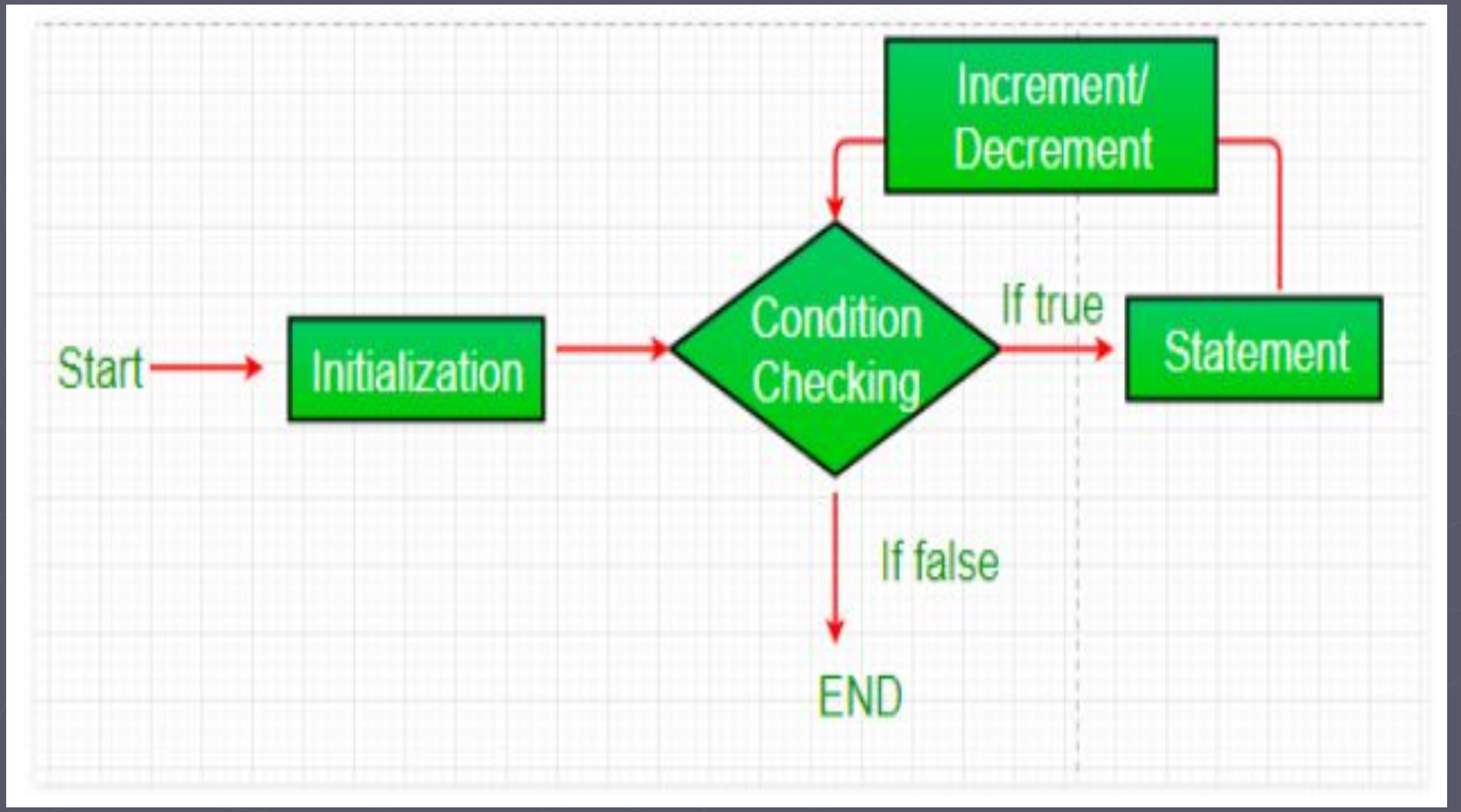
- Entry Controlled loops: In this type of loops the test condition is tested before entering the loop body. For Loop and While Loop are entry controlled loops.
- Exit Controlled Loops: In this type of loops the test condition is tested or evaluated at the end of loop body. Therefore, the loop body will execute atleast once, irrespective of whether the test condition is true or false. do – while loop is exit controlled loop.

for loop:

- for statement consumes the initialization, condition and increment/decrement in one line thereby providing a shorter, easy to debug structure of looping.
- Syntax:

```
for (initialization condition; testing condition;
                                increment/decrement)
{
    statement(s)
}
```

Flowchart


Increment/
Decrement
Start
Initialization
Condition
Checking
If true
Statement
If false
END

for loop execution

- Initialization condition: Here, we initialize the variable in use. It marks the start of a for loop. An already declared variable can be used or a variable can be declared, local to loop only.
- Testing Condition: It is used for testing the exit condition for a loop. It must return a boolean value. It is also an Entry Control Loop as the condition is checked prior to the execution of the loop statements.
- Statement execution: Once the condition is evaluated to true, the statements in the loop body are executed.
- Increment/ Decrement: It is used for updating the variable for next iteration.
- Loop termination:When the condition becomes false, the loop terminates marking the end of its life cycle.

Example

```
<!doctype html>
<html>
<body>
<script type = "text/javaScript">
// JavaScript program to illustrate for loop
      var x;
    // for loop begins when x=2
    // and runs till x <=4
    for (x = 1; x <= 5; x++)
    {
     document.write("Value of x:" + x + "<br />");
 }
  </script>
</body>
</html>
```

Value of x:1
Value of x:2
Value of x:3
Value of x:4
Value of x:5

for...in loop

JavaScript also includes another version of for loop also known as the for..in Loops.

The for..in loop provides a simpler way to iterate through the properties of an object.

This loop is seen to be very useful while working with objects.

Syntax:

```
for (variableName in Object)
{
    statement(s)
}
```

Example

```
<!doctype html>
<html>
<body>
<script type = "text/javaScript">
// JavaScript program to illustrate for..in loop
      // creating an Object
    var lang= { first : "C", second : "Java",
                    third : "Python", fourth : "PHP",
                    fifth : "JavaScript" };
      // iterate through every property of the
    // object languages and print all of them
    // using for..in loops
    for (itr in lang)
    {         document.write(lang[itr] + "<br >");
    }
  </script>  </body> </html>
```

C
Java
Python
PHP
JavaScript

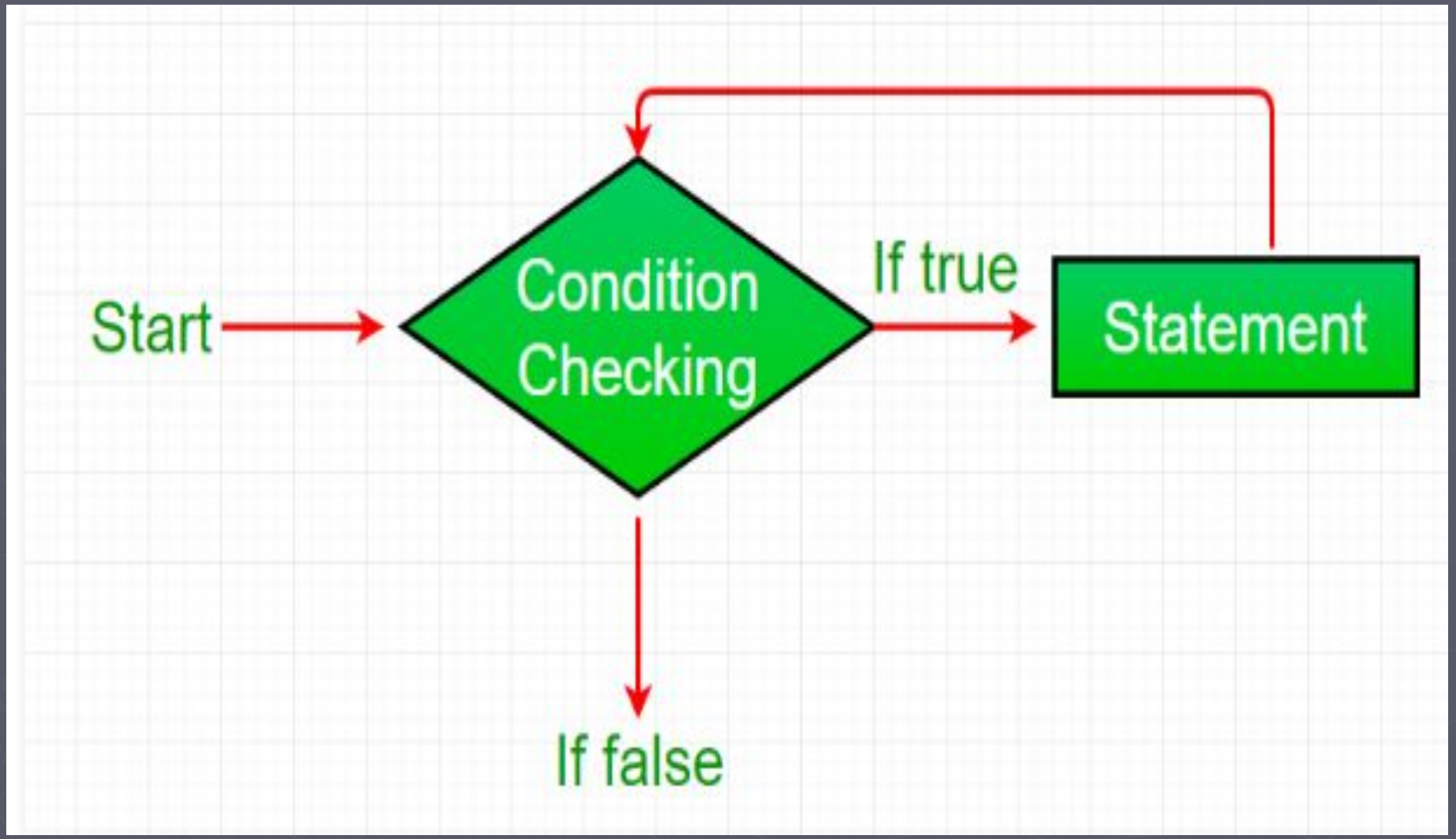
while loop

while loop: A while loop is a control flow statement that allows code to be executed repeatedly based on a given Boolean condition. The while loop can be thought of as a repeating if statement.

Syntax :

```
while (boolean condition)
{
   loop statements...
}
```

Flowchart



EXECUTION

While loop starts with the checking of condition. If it evaluated to true, then the loop body statements are executed otherwise first statement following the loop is executed. For this reason it is also called Entry control loop

Once the condition is evaluated to true, the statements in the loop body are executed. Normally the statements contain an update value for the variable being processed for the next iteration.

When the condition becomes false, the loop terminates which marks the end of its life cycle.

EXAMPLE

```
<!dOCTYPE HTML>
<HTML>
<BODY>
<script type = "text/javaScript">
// JavaScript program to illustrate while loop
      var x = 1;
      // Exit when x becomes greater than 4
    while (x <= 4)
    {
        document.write("Value of x:" + x + "<br />");
          // increment the value of x for
        // next iteration
        x++;
    }
  </script>
</BODY> </HTML>
```

Value of x:1
Value of x:2
Value of x:3
Value of x:4

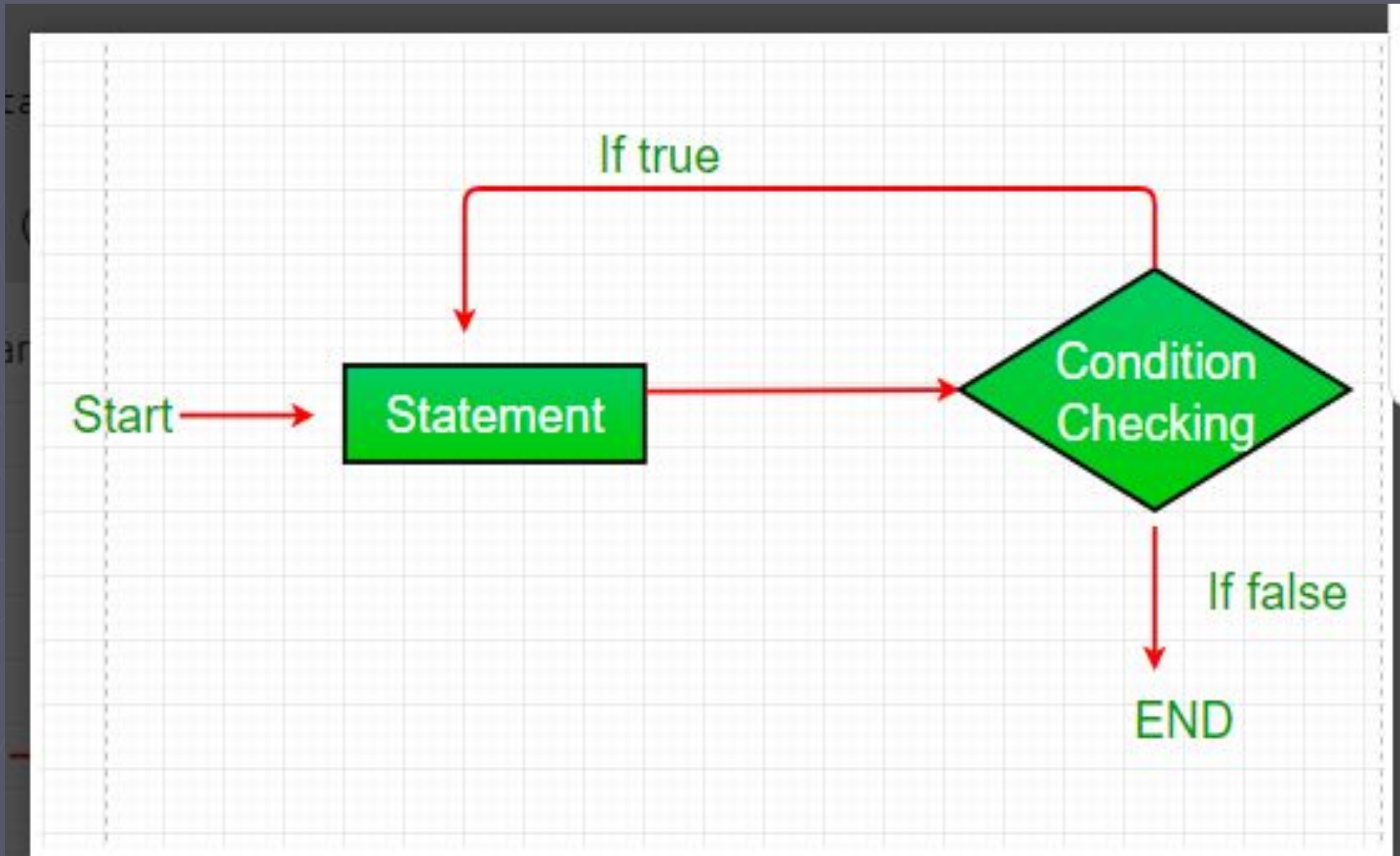
do while

do while loop is similar to while loop with only difference that it checks for condition after executing the statements, and therefore is an example of Exit Control Loop.

Syntax:

```
do
{
    statements..
}
while (condition);
```

Flowchart:



Execution

- do while loop starts with the execution of the statement(s). There is no checking of any condition for the first time.
- After the execution of the statements, and update of the variable value, the condition is checked for true or false value. If it is evaluated to true, next iteration of loop starts.
- When the condition becomes false, the loop terminates which marks the end of its life cycle.
- It is important to note that the do-while loop will execute its statements atleast once before any condition is checked, and therefore is an example of exit control loop.

Example

```
!Doctype Html>
<html>
<body>
<script type = "text/javaScript">
// JavaScript program to illustrate do-while loop
      var x = 21;
      do
    {
        // The line while be printer even
        // if the condition is false
        document.write("Value of x:" + x + "<br />");
          x++;
    } while (x < 20);
  </script>
</body>
</html>
```