JavaScript Operators

JavaScript operators are symbols that are used to perform operations on operands. For example:

1. var sum=10+20;

Here, + is the arithmetic operator and = is the assignment operator.

There are following types of operators in JavaScript.

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

JavaScript Arithmetic Operators

Arithmetic operators are used to perform arithmetic operations on the operands. The following operators are known as JavaScript arithmetic operators.

Operator	Description	Example
+	Addition	10+20 = 30
-	Subtraction	20-10 = 10
*	Multiplication	10*20 = 200
/	Division	20/10 = 2
%	Modulus (Remainder)	20%10 = 0
++	Increment	var a=10; a++; Now a = 11
--	Decrement	var a=10; a--; Now a = 9

JavaScript Comparison Operators

The JavaScript comparison operator compares the two operands. The comparison operators are as follows:

Operator	Description	Example
==	Is equal to	10==20 = false
===	Identical (equal and of same type)	10==20 = false
!=	Not equal to	10!=20 = true
!==	Not Identical	20!==20 = false
>	Greater than	20>10 = true
>=	Greater than or equal to	20>=10 = true
<	Less than	20<10 = false
<=	Less than or equal to	20<=10 = false

JavaScript Bitwise Operators

The bitwise operators perform bitwise operations on operands. The bitwise operators are as follows:

Operator	Description	Example
&	Bitwise AND	(10==20 & 20==33) = false
|	Bitwise OR	(10==20 | 20==33) = false
^	Bitwise XOR	(10==20 ^ 20==33) = false
~	Bitwise NOT	(~10) = -10
<<	Bitwise Left Shift	(10<<2) = 40

>>	Bitwise Right Shift	(10>>2) = 2
>>>	Bitwise Right Shift with Zero	(10>>>2) = 2

JavaScript Logical Operators

The following operators are known as JavaScript logical operators.

Operator	Description	Example
&&	Logical AND	(10==20 && 20==33) = false
||	Logical OR	(10==20 || 20==33) = false
!	Logical Not	!(10==20) = true

JavaScript Assignment Operators

The following operators are known as JavaScript assignment operators.

Operator	Description	Example
=	Assign	10+10 = 20
+=	Add and assign	var a=10; a+=20; Now a = 30
-=	Subtract and assign	var a=20; a-=10; Now a = 10
*=	Multiply and assign	var a=10; a*=20; Now a = 200
/=	Divide and assign	var a=10; a/=2; Now a = 5
%=	Modulus and assign	var a=10; a%=2; Now a = 0

JavaScript Special Operators

The following operators are known as JavaScript special operators.

Operator	Description
(?:)	Conditional Operator returns value based on the condition. It is like if-else.
,	Comma Operator allows multiple expressions to be evaluated as single statement.
delete	Delete Operator deletes a property from the object.
in	In Operator checks if object has the given property
instanceof	checks if the object is an instance of given type
new	creates an instance (object)
typeof	checks the type of object.
void	it discards the expression's return value.
yield	checks what is returned in a generator by the generator's iterator

JavaScript If-else

The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

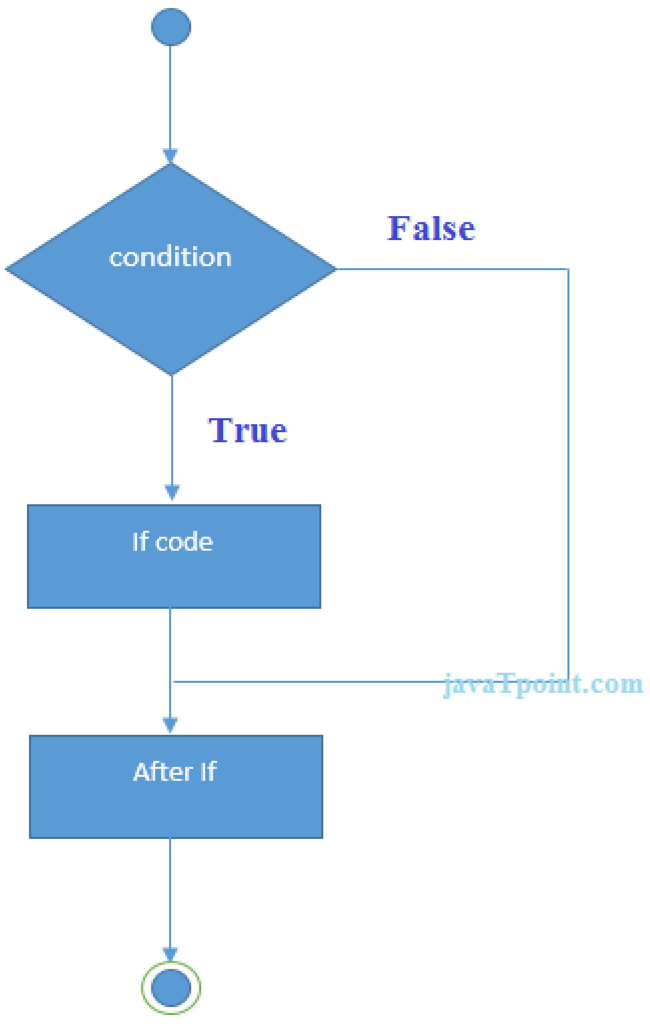
JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
```

}

Flowchart of JavaScript If statement



Let's see the simple example of if statement in javascript.

```
<script>
var a=20;
if(a>10){
document.write("value of a is greater than 10");
}
</script>
```

Output of the above example

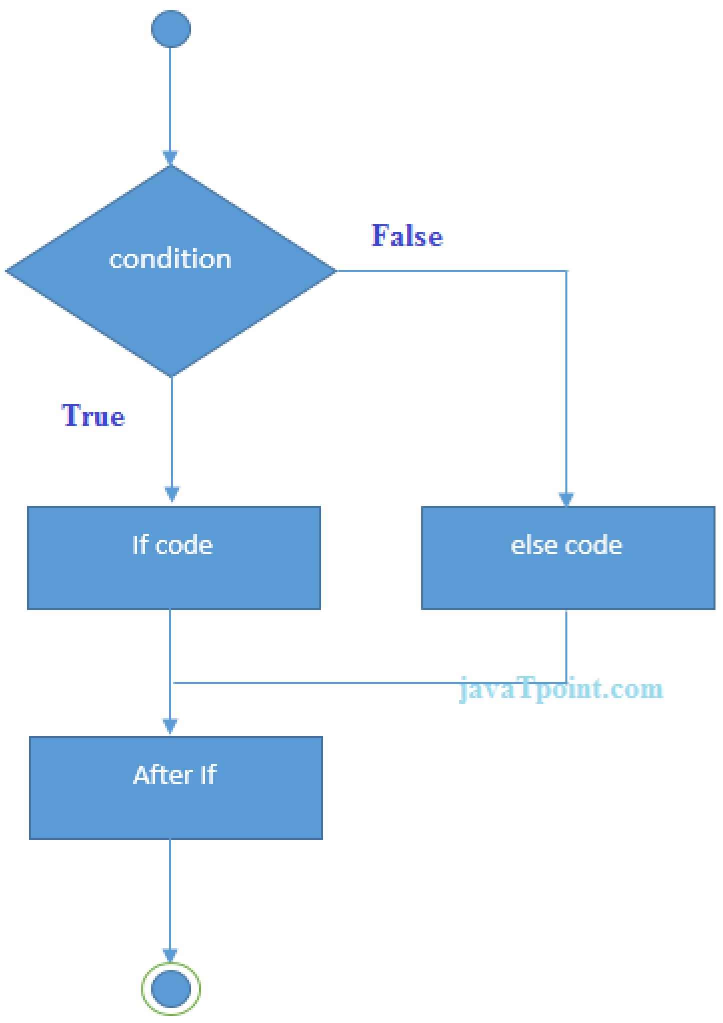
value of a is greater than 10

JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

1. if(expression){
2. //content to be evaluated if condition is true
3. }
4. else{
5. //content to be evaluated if condition is false
6. }

Flowchart of JavaScript If...else statement



Let’s see the example of if-else statement in JavaScript to find out the even or odd number.

```
<script>
var a=20;
if(a%2==0){
document.write("a is even number");
}
else{
document.write("a is odd number");
}
</script>
```

Output of the above example

a is even number

JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

```
if(expression1){
//content to be evaluated if expression1 is true
}
else if(expression2){
//content to be evaluated if expression2 is true
}
else if(expression3){
//content to be evaluated if expression3 is true
}
else{
//content to be evaluated if no expression is true
}
```

Let’s see the simple example of if else if statement in javascript.

```
<script>
var a=20;
if(a==10){
document.write("a is equal to 10");
}
else if(a==15){
document.write("a is equal to 15");
}
else if(a==20){
document.write("a is equal to 20");
}
```

```
else{
document.write("a is not equal to 10, 15 or 20");
}
</script>
```

Output of the above example

```
a is equal to 20
```

JavaScript Switch

What is Switch Case in JavaScript?

In JavaScript, a switch case is a conditional statement that is used so it can provide a way to execute the statements based on their conditions or different conditions.

A switch case statement is used to perform different actions based on the conditions that are present in the program or given by the user.

In other words, a switch statement evaluates the condition to match the condition with the series of cases and if the condition is matched then it executes the statement. If the condition is not matched with any case then the default condition will be executed.

In JavaScript, a switch case statement is an alternative approach for if else conditions.

The JavaScript switch statement is used to execute one code from multiple expressions. But it is convenient that if else if because it can be used with numbers, characters etc.

Syntax

Let's see the syntax of the Switch case statement in JavaScript:

```
switch (expression){
case value1:
Code to be executed;
break;
Case value2:
code to be executed;
break;
.........
default:
Code to be executed if the above values are not matched;
}
```

Break Statementx

In JavaScript, a break keyword is used within the switch case statement this statement indicates the end of a particular case of the switch statement. In a switch statement if a break statement is out then the interpreter would continue executing each of the statements in each following case.

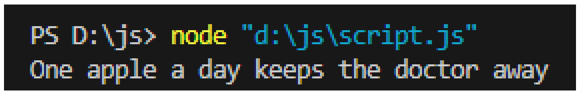
Default Statement

In JavaScript, a default keyword is used in a switch statement to define the default expression. In JavaScript, when any case doesn't match with the expression of the switch-case statement then it will execute the default code block.

Example

```
const expression = 'Apple';
switch (expression){
  case 'Oranges':
    console.log('I prefer Oranges');
    break;
  case 'watermelons':
    console.log('I prefer watermelon too!');
    break;
  case 'Apple':
    console.log('One apple a day keeps the doctor away);
    break;
  default:
    console.log(`Sorry, we are out of ${expression}`);
}
```

Output:



```
PS D:\js> node "d:\js\script.js"
One apple a day keeps the doctor away
```

How does Switch statements work?

Evaluation

In JavaScript, inside the switch case statement, we evaluate the expression once.

Comparison

In JavaScript, when we use the switch statement the value of the expression is compared with each case with the use of strict equality.

Execution

In JavaScript, a switch case expression match is found then the corresponding code block following the matching expression will be executed. If there is no match of expression then the execution will jump to the default case otherwise it will continue with the next statement after the switch block.

Break Statement

After the execution of the code block in a program break statement helps us so it terminates the switch statement and prevents the execution from falling through to subsequent cases.

In JavaScript, when the break statement is suppressed then the execution will continue to the next switch case and it is also known as fall through.

Default Case

In JavaScript, the default case is optional means it depends on the programmer if he wants to use it or not. If no match is found in the code then the code block under default case is executed.

Difference between if-else and switch statements

Syntax

Switch statement: In JavaScript, it uses the switch keyword, followed by parentheses in which an expression inside the parentheses and a series of case statements.

If-else statement: It uses the if keyword and is followed by a condition in parentheses and an optional else clause.

Purpose

Switch statement: In JavaScript, switch statements check a single expression with multiple cases and it executes the statement based on which case matches with the statement.

If-else statement: In JavaScript, the if statement checks the single condition and executes the different code blocks that are based on whether the condition is true or false.

Cases

Switch statement: In JavaScript, the switch statement can have multiple cases and each of the cases can have a different value to match with the expression.

If-else statement: In the if-else statement we have only two possible code blocks one for the true condition and another for the when it's false.

Default

Switch statement: In JavaScript, we can include a default statement to execute if in the program we don't have any case statements that match the expression.

If-else statement: In JavaScript, we can include the else statement to execute the condition is false.

Efficiency

Switch Statement: It can be more efficient if there are more cases to check the statement compared to the if-else statement.

If-else statement: It can be less efficient compared to the switch statement because of multiple cases or conditions.

Readability

Switch statement: In JavaScript, the switch statement can be easier to read compared to an if-else statement if there are multiple statements to check.

If-else statement: In JavaScript, the if-else condition can be more flexible and powerful if you compare it with the switch condition in the case of a complex condition to check.