

Chương 09

Mục tiêu bài học

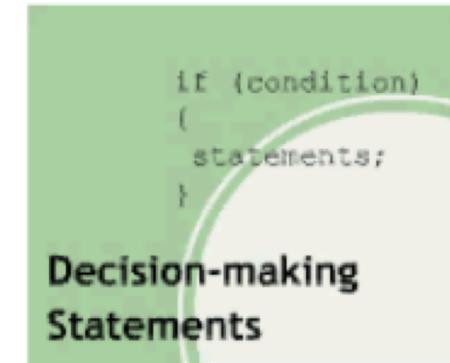
- Toán tử cơ bản
- Toán tử nâng cao
- Câu lệnh quyết định
(Decision-making statement)



Basics Operators



Advanced Operators



- Toán tử được sử dụng để thực hiện thao tác một hoặc nhiều giá trị được lưu trữ trong các biến. Thao tác có thể là thay đổi giá trị hoặc phát sinh ra giá trị mới.
- Có ba loại toán tử chủ yếu sau:
 - Toán tử một ngôi: Thao tác trên một toán hạng.
 - Toán tử hai ngôi: Thao tác trên hai toán hạng.

- Toán tử số học
- Toán tử quan hệ
- Toán tử logic
- Toán tử gán
- Toán tử đặc biệt
- Mức ưu tiên

- Toán tử số là toán tử hai ngôi cho thực hiện tính toán trên giá trị số hoặc chuỗi:
 - Cộng (+)
 - Trừ (-)
 - Nhân (*)
 - Chia (/)
 - Lấy phần dư (%)

Arithmetic Operator	Description	Example
+	Performs addition. In case of string values, it behaves as a string concatenation operator and appends a string at the end of the other.	45 + 56
/	Performs division. It divides the first operand by the second operand and returns the quotient.	24 / 8
%	Performs the modulo operation. It divides the first operand by the second operand and returns the remainder.	90 % 20
*	Performs multiplication. It multiplies the two operands.	98 * 10
-	Performs subtraction. If a larger value is subtracted from a smaller value, it returns a negative numeric value.	78 - 76

- Toán tử tăng giảm

- ++

- --

Expression	Type	Result
numTwo = ++numOne;	Pre-increment	numTwo = 3
numTwo = numOne++;	Post-increment	numTwo = 2
numTwo = --numOne;	Pre-decrement	numTwo = 1
numTwo = numOne--;	Post-decrement	numTwo = 2
- (numOne)	-	-2

- Là toán tử hai ngôi dùng để so sánh giữa hai toán hạng và trả về giá trị boolean dựa trên sự so sánh có đúng hay không.
- Toán tử so sánh bao gồm:
 - So sánh bằng (==)
 - So sánh khác (!=)
 - So sánh lớn hơn (>)
 - So sánh lớn hơn hoặc bằng (>=)
 - So sánh nhỏ hơn (<)
 - So sánh nhỏ hơn hoặc bằng (<=)
 - Toán tử === dùng để xác minh xem hai toán hạng có bằng nhau và cùng kiểu hay không

- Các toán tử logic dùng để kết hợp các so sánh trong một biểu thức điều kiện.
- Bao gồm:

Logical Operator	Description	Example
&& (AND)	Returns true if both the operands are evaluated to true or returns false. If first operand evaluates to false, it will ignore the second operand and will return false.	$(x == 2) \&\& (y == 5)$ Returns false.
!	Returns false if the expression is true and vice-versa.	$!(x==3)$ Returns true.
 (OR)	Returns true if either of the operands are evaluated to true. If first operand evaluates to true, it will ignore the second operand and will return true.	$(x == 2) (y == 5)$ Returns true.

- Toán tử chuỗi lấy các toán tử chuỗi như các toán hạng và tạo một chuỗi mới, kết quả là một chuỗi kết hợp các chuỗi con.

Ví dụ:

```
x = 'yellow';  
y = 'green';  
z = x + y + 'white'; //which means z is "yellowgreenwhite"  
w = y + 9; //which means w is "green9"
```

- Toán tử gán được chia thành hai nhóm
 - Toán tử gán đơn: là toán tử '=' dùng để gán giá trị hay biểu thức cho biến.
 - Toán tử gán phức hợp: là toán tử gán đơn đi cùng với một toán tử toán học. Lấy giá trị của biến ra tính toán rồi gán trở lại

Expression	Description	Result
numOne += 6;	numOne = numOne + 6	numOne = 12
numOne -= 6;	numOne = numOne - 6	numOne = 0
numOne *= 6;	numOne = numOne * 6	numOne = 36
numOne %= 6;	numOne = numOne % 6	numOne = 0
numOne /= 6;	numOne = numOne / 6	numOne = 1

Bitwise Operator	Description	Example
& (Bitwise AND)	Compares two bits and returns 1 if both of them are 1 or else returns 0.	00111000 & 00011100 Returns 00011000.
~ (Bitwise NOT)	Inverts every bits of the operand and is a unary operator.	~00010101 Returns 11101010.
 (Bitwise OR)	Compares two bits and returns 1 if the corresponding bits of either or both the operands is 1.	00111000 00011100 Returns 00111100.
^ (Bitwise XOR)	Compares two bits and returns 1 if the corresponding bit of either but not both the operands is 1.	00111000 00011100 Returns 00100100.

- Các toán tử chỉnh bao gồm:
 - Toán tử điều kiện
(condition) ? trueVal : falseVal

Gán một giá trị xác định vào một biến nếu điều kiện đúng, trường hợp còn lại thì gán vào biến còn lại.
ví dụ.

```
status = (age >= 18) ? "adult" : "minor"
```

- Toán tử typeof
Toán tử typeof trả về một chuỗi cho biết kiểu của toán hạng.
ví dụ.

```
var x = 5;  
document.write(typeof(x));
```

Mức ưu tiên của toán tử

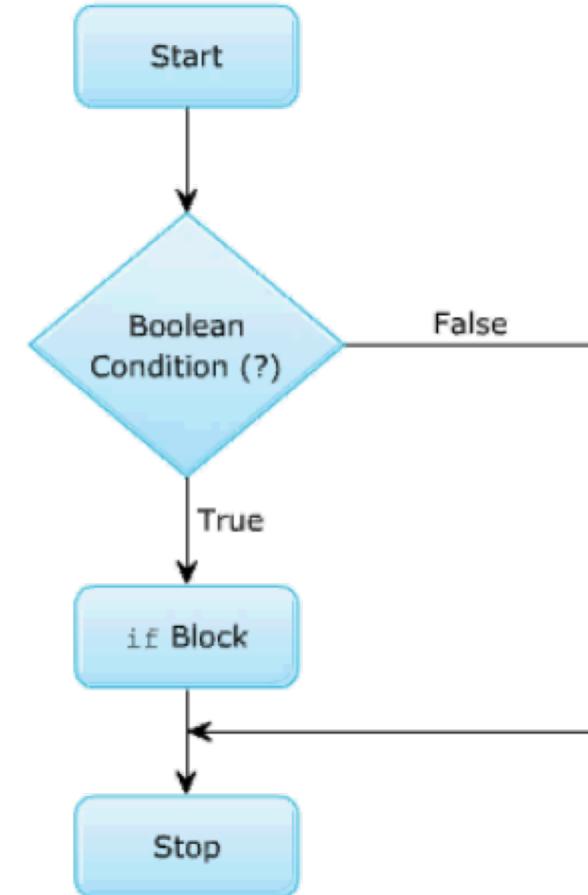
- Khi có nhiều toán tử được tính toán trong một biểu thức, mức ưu tiên của toán tử xác định thứ tự thực hiện của toán tử trong biểu thức đó.
- Bảng sau liệt kê mức ưu tiên của các toán tử từ thấp đến cao:

Precedence Order	Operator	Description	Associativity
1	()	Parentheses	Left to Right
2	++, --	Post-increment and post-decrement operators	Not Applicable
3	typeof, ++, --, ~, ~, !	Pre-increment and pre-decrement operators, Logical NOT, Bitwise NOT, and Unary negation.	Right to Left
4	*, /, %	Multiplication, Division, and Modulo	Left to Right
5	+, -	Addition and Subtraction	Left to Right
6	<, <=, >, >=	Less than, Less than or equal, Greater than, Greater than or equal	Left to Right
7	==, ===, !=, !==	Equal to, Strict equal to, Not equal to, Strict not equal to	Left to Right
8	&, , ^, &&,	Bitwise AND, Bitwise OR, Bitwise XOR, Logical AND, Logical OR	Left to Right
9	?:	Conditional operator	Right to Left
10	=, +=, -=, *=, /=, %=	Assignment operators	Right to Left
11	,	Comma	Left to Right

- Câu lệnh điều kiện được dùng để kiểm tra điều kiện. Kết quả xác định câu lệnh hoặc khối lệnh được thực thi.
- Các câu lệnh điều kiện bao gồm:
 - if
 - if...else
 - if...else if...else
 - switch

- Cú pháp

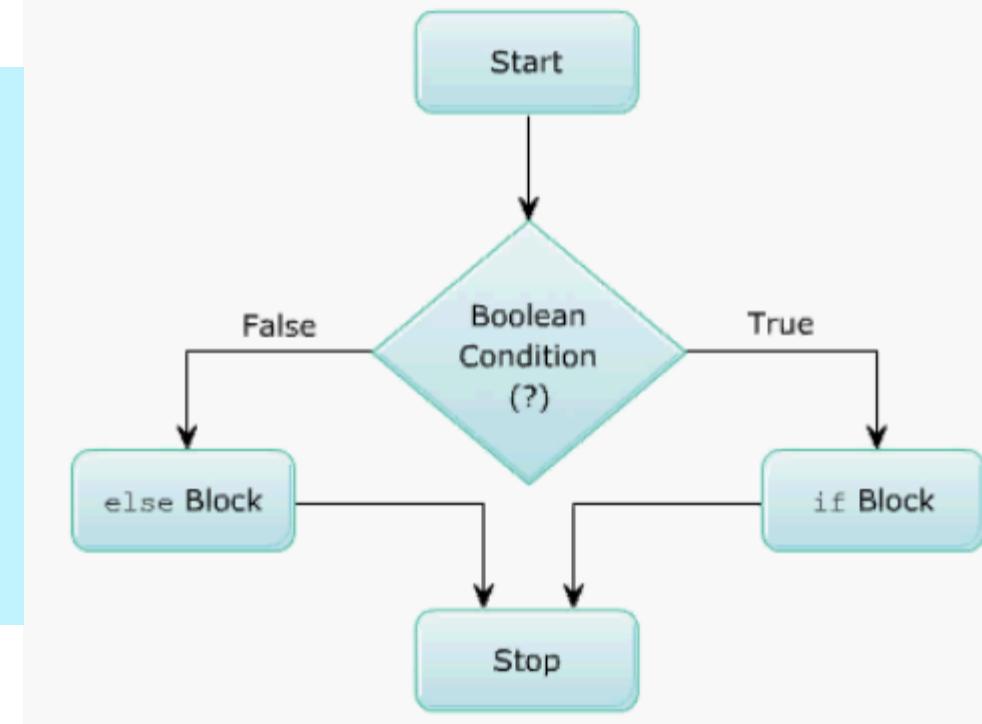
```
if (điều_kiện)
{
    //Một hoặc nhiều câu lệnh
}
```



Lệnh if – else

- Cú pháp

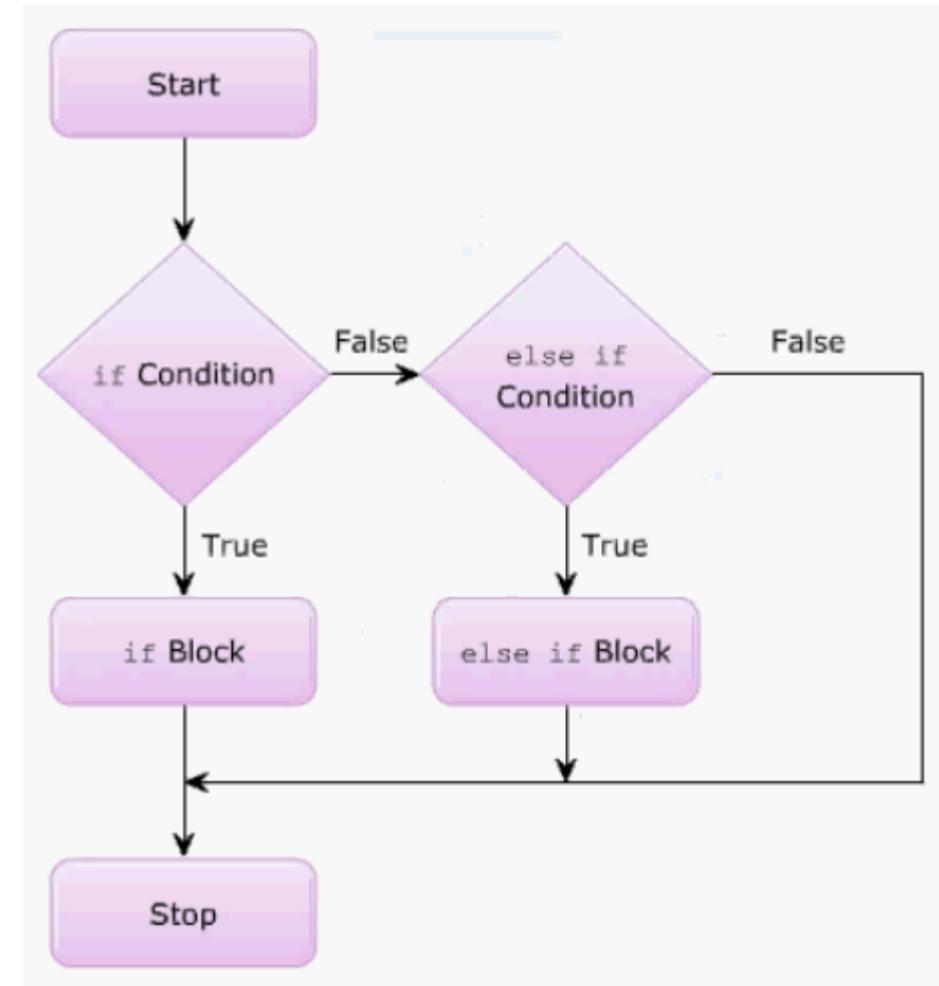
```
if (điều_kiện)
{
    //Một hoặc nhiều câu lệnh;
}
else
{
    //Một hoặc nhiều câu lệnh;
}
```



Lệnh if – else – if

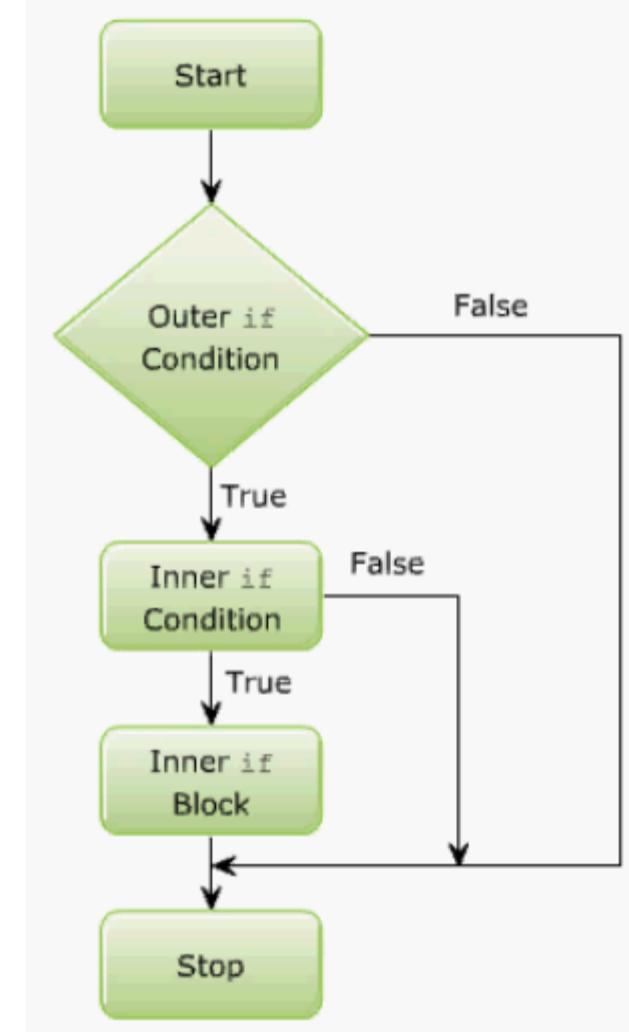
- Cú pháp

```
if (điều_kiện_1)
{
    câu lệnh_1;
}
else if (điều_kiện_2)
{
    câu lệnh_2;
}
else
{
    câu_lệnh_n+1;
}
```



Lệnh if lồng nhau

```
if (điều_kiện_1)
{
    câu lệnh;
    if (điều_kiện)
    {
        câu lệnh;
        if (điều_kiện)
        {
            câu lệnh;
        }
    }
}
```



- Cú pháp

```
switch (expression)
{
    case constant1:
        statement sequence
        break;
    case constant2:
        statement sequence
        break;
    case constant3:
        statement sequence
        break;

    .
    .
    .

    default:
        statement sequence
}
```

- Một biểu thức quy tắc là một kiểu được xác định trong việc tìm kiếm tương ứng các ký tự kết hợp của một chuỗi.
- Các biểu thức quy tắc có thể dùng để tìm kiếm các mẫu ký tự trong một chuỗi nhập vào từ người sử dụng.
- Biểu thức quy tắc bao gồm:
 - Các mẫu đơn giản
 - Các ký tự đơn giản và ký tự đặc biệt

- Các biểu thức quy tắc có thể được tạo bằng một trong hai cách:

- Sử dụng khởi tạo đối tượng

```
var tên_biến=/mẫu_biểu_thức/bổ_tù;
```

- Gọi hàm khởi tạo của đối tượng RegExp()

```
var tên_biến=new RegExp ("mẫu_biểu_thức",bổ_tù)
```

Bổ từ	Mô tả
<u>i</u>	Không phân biệt chữ hoa chữ thường
<u>g</u>	Thực hiện tìm tất cả các chuỗi con thỏa mãn mẫu Biểu Thức
<u>m</u>	Thực hiện tìm trên nhiều dòng

- Các phương thức sử dụng trong biểu thức quy tắc bao gồm:
 - exec(), test()

Character	Matches	Example
\d	Numeral 0 through 9	/\d\d\d/ matches "212" and "415" but not "B17"
\D	Non-numeral	/\D\D\D/ matches "ABC" but not "212" or "B17"
\s	Single white space	/over\sbite/ matches "over bite" but not "overbite" or "over bite"
\S	Single non-white space	/over\Sbite/ matches "over-bite" but not "overbite" or "over bite"
\w	Letter, numeral, or underscore	/A\w/ matches "A1" and "AA" but not "A+"
\W	Not letter, numeral, or underscore	/A\W/ matches "A+" but not "A1" and "AA"
.	Any character except newline	/.../ matches "ABC", "1+3", "A 3", or any three characters
[...]	Character set	/[AN]BC/ matches "ABC" and "NBC" but not "BBC"
[^...]	Negated character set	/[^AN]BC/ matches "BBC" and "CBC" but not "ABC" or "NBC"

Table 38-2 JavaScript Regular Expression Counting Metacharacters

Character	Matches Last Character	Example
*	Zero or more times	/Ja*vaScript/ matches "JavaScript", "JavaScript", and "JaaavaScript" but not "JovaScript"
?	Zero or one time	/Ja?vаСript/ matches "JavaScript" or "JavaScript" but not "JaaavaScript"
+	One or more times	/Ja+vaScript/ matches "JavaScript" or "JaavaScript" but not "JvaScript"
{n}	Exactly n times	/Ja{2}vaScript/ matches "JaavaScript" but not "JvaScript" or "JavaScript"
{n,}	n or more times	/Ja{2,}vaScript/ matches "JaavaScript" or "JaaavaScript" but not "JavaScript"
{n,m}	At least n, at most m times	/Ja{2,3}vaScript/ matches "JaavaScript" or "JaaavaScript" but not "JavaScript"

Table 38-3 JavaScript Regular Expression Positional Metacharacters

Character	Matches Located	Example
^	At beginning of a string or line	/^Fred/ matches "Fred is OK" but not "I'm with Fred" or "Is Fred here?"
\$	At end of a string or line	/Fred\$/ matches "I'm with Fred" but not "Fred is OK" or "Is Fred here?"

Symbol	Description	Example
()	Organizes characters together in a group to specify a set of characters in a string.	/ (xyz) + (uvw) / Matches one or more number of occurrences of "xyz" followed by one occurrence of <u>uvw</u> .
	Combines sets of characters into a single regular expression and then matches any of the character set.	/ (xy) (uv) (st) / Matches "xy" or "uv" or "st".

Table 38-1 JavaScript Regular Expression Matching Metacharacters

Character	Matches	Example
\b	Word boundary	<code>/\bor/</code> matches "origami" and "or" but not "normal" <code>/or\b/</code> matches "traitor" and "or" but not "perform" <code>/\bor\b/</code> matches full word "or" and nothing else
\B	Word non-boundary	<code>/\Bor/</code> matches "normal" but not "origami" <code>/or\B/</code> matches "normal" and "origami" but not "traitor" <code>/\Bor\B/</code> matches "normal" but not "origami" or "traitor"

Ví dụ:

```
<HTML>
```

```
    <HEAD>
```

```
        <SCRIPT LANGUAGE="JavaScript">
            re = /Time/
            str = re.test ("Time and Tide wait for none");
            window.alert(str);
        </SCRIPT>
```

```
    </HEAD>
```

```
</HTML>
```

