Bài 13

Toán tử và câu lệnh

Mục tiêu

- Giải thích về toán tử và các loại toán tử trong JavaScript
- Giải thích về biểu thức trong JavaScript
- Giải thích về các câu lệnh rẽ nhánh trong JavaScript

Giới thiệu

Một toán tử quy định cụ thể loại hình hành động được thực hiện trên các giá trị của biến và biểu thức.

JavaScript cung cấp các loại khác nhau của các toán tử để thực hiện việc tính toán đơn giản và phức tạp.

Một số toán tử cũng được sử dụng để xây dựng các câu lệnh logic và cấu trúc quan hệ. Các câu lệnh cho phép thực hiện việc rẽ nhánh và cấu trúc lặp.

Khái niệm cơ bản của các toán tử 1-2

Một phép toán là một hành động thực hiện trên một hoặc nhiều giá trị được lưu trữ trong các biến.

Các hành động cụ thể sẽ thay đổi giá trị của biến hoặc tạo ra một giá trị mới.

Một hoạt động đòi hỏi tối thiểu một biểu tượng và một số giá trị.

Biểu tượng được gọi là một toán tử và quy định cụ thể loại hành động được thực hiện trên các giá trị.

Giá trị hoặc biến mà trên đó các thao tác được thực hiện được gọi là một toán hạng.

Khái niệm cơ bản của các toán tử 2-2

- Ba loại chính của các toán tử như sau:

Toán tử 1 ngôi – Thực hiện trên một toán hạng.

Toán tử 2 ngôi – Thực hiện trên hai toán hạng.

Toán tử 3 ngôi- Thực hiện trên ba toán hạng.

Các kiểu toán tử khác nhau

Các toán tử giúp đơn giản hóa các biểu thức.

JavaScript cung cấp một loạt các toán tử được định nghĩa trước cho phép thực hiện các hoạt động khác nhau.

Toán tử trong JavaScript được phân thành sáu loại dựa trên loại hành động mà chúng thực hiện trên các toán hạng.

- Arithmetic operators (Số học)
- Relational operators (Quan hệ)
- Logical operators (Logic)
- Assignment operators(Gán)
- Bitwise operators(Bit)
- Special operators(Đặc biệt)

Toán tử số học 1-2

Là các toán tử hai ngôi.

Thực hiện phép tính số học cơ bản trên hai toán hạng.

Cho phép bạn thực hiện các tính toán trên giá trị số và chuỗi.

- Danh sách bảng số học.

Toán tử	Mô tả	Ví dụ
+ (Addition)	Thực hiện cộng hai số hoặc hai chuỗi	45 + 56
- (Subtraction)	Thực hiện trừ hai số	76-78
/ (Division)	Thực hiện chia hai số	24 / 8
% (Modulo)	Thực hiện chia hai số lấy phần dư	90 % 20
* (Multiplication)	Thực hiện nhân hai số	98 * 10

Toán tử số học 2-2

- Ví dụ

```
<SCRIPT>
  var loanAmount = 34500;
  var interest = 8;
  var interestAmount, totalAmount;
  interestAmount = loanAmount * (interest / 100);
  totalAmount = loanAmount + interestAmount;
  document.write("<B>Total amount to be paid ($):</B>" +
totalAmount + "<BR />");
</SCRIPT>
```

Các toán tử tăng và giảm 1-2

Tăng và giảm bớt toán hạng đi 1 đơn vị

Toán tử tăng (+ +) làm tăng 1 giá trị, trong khi các toán tử giảm (--) làm giảm 1 giá trị.

Các toán tử có thể được đặt trước hoặc sau toán hạng.

Operator nếu đặt trước toán hạng, thể hiện việc tăng trước hoặc giảm trước khi tính toán.
Operator nếu đặt sau các toán hạng, thể hiện việc tăng sau hoặc giảm sau khi tính toán.

- Bảng sau sẽ liệt kê các toán tử tăng giảm (numOne=2).

Biểu thức	Loại
numTwo = ++numOne;	Tăng trước
numTwo = numOne++;	Tăng sau
numTwo = --numOne;	Giảm trước
numTwo = numOne--;	Giảm sau

Các toán tử tăng và giảm 2-2

- Ví dụ

```
<SCRIPT>
  var number = 3;
  alert('Number after increment = ' + ++number);
  alert('Number after decrement = ' + number--);
</SCRIPT>
```

Các toán tử quan hệ 1-3

Là các toán tử hai ngôi để so sánh giữa hai toán hạng.

au khi thực hiện một sự so sánh, họ trở về một giá trị boolean cụ thể là, true hoặc false.

Biểu thức bao gồm một toán tử quan hệ được gọi là biểu thức quan hệ hoặc biểu thức điều kiện.

- Bảng dưới đây liệt kê các toán tử quan hệ.

Các toán tử quan hệ	Mô tả
== (Equal)	Kiểm tra xem hai toán hạng có bằng nhau không?
!= (Not Equal)	Kiểm tra hai toán hạng có khác nhau không?
=== (Strict Equal)	Kiểm tra hai toán hạng có bằng nhau và cùng kiểu không?
!== (Strict Not Equal)	Kiểm tra xem hai toán hạng có bằng nhau không cho dù cùng kiểu

Các toán tử quan hệ 2-3

Các toán tử quan hệ	Mô tả
> (Greater Than)	Kiểm tra xem toán hạng bên trái có lớn hơn bên phải không?
< (Less Than)	Kiểm tra xem toán hạng bên trái có nhỏ hơn bên phải không?
>= (Greater Than or Equal)	Kiểm tra xem toán hạng bên trái có lớn hơn hay bằng bên phải không?
<= (Less Than or Equal)	Kiểm tra xem toán hạng bên trái có nhỏ hơn hay bằng bên phải không?

Các toán tử quan hệ 3-3

- Ví dụ

```
<SCRIPT>
  var firstNumber = 3;
  var secondNumber = 4;
  document.write('First number is greater than the second
number: ' + (firstNumber > secondNumber));
  document.write('<br/>First number is less than the
second number: ' + (firstNumber < secondNumber));
  document.write('<br/>First number is equal to the second
number: ' + (firstNumber == secondNumber));
</SCRIPT>
```

Các toán tử logic 1-2

Là toán tử hai ngôi thực hiện kiểm tra logic trên hai toán hạng

Chúng thuộc nhóm toán tử quan hệ và trả về giá trị kiểu boolean (true hoặc false)

- Following table lists the logical operators.

Các toán tử logic	Mô tả
&& (AND)	Trả về true, nếu tất cả các toán hạng được đánh giá đúng. Nếu toán hạng đầu tiên để đánh giá sai, nó sẽ bỏ qua các toán hạng thứ hai
! (NOT)	Trả về false nếu toán hạng đúng và ngược lại
|| (OR)	Trả về true, nếu một trong các toán hạng được đánh giá đúng. Nếu toán hạng đầu tiên để đánh giá đúng, nó sẽ bỏ qua các toán hạng thứ hai

Các toán tử logic 2-2

- Ví dụ

```
<SCRIPT>
  var name = “John”;
  var age = 23;
  alert(‘John\’s age is greater than or equal to 23 years : ‘ +
((name==”John”) && (age >= 23)));
</SCRIPT>
```

Các toán tử gán

Toán tử gán gán giá trị của toán hạng bên phải cho toán hạng bên trái bằng cách sử dụng toán tử (=).

Toán tử gán đơn giản là '=' được sử dụng để gán một giá trị hoặc kết quả của một biểu thức cho một biến.

Phép gán gộp bao gồm một toán tử gán kết hợp với một phép toán.

- Ví dụ

Biểu thức	Mô tả
numOne += 6;	numOne = numOne + 6
numOne -= 6;	numOne = numOne – 6
numOne *= 6;	numOne = numOne * 6
numOne %= 6;	numOne = numOne % 6
numOne /= 6;	numOne = numOne / 6

Các toán tử Bitwise 1-2

Thể hiện các toán hạng là các bit (0 hoặc 1) và thực hiện thao tác trên chúng.

Chúng trả về các giá trị nhị phân.

Các toán tử Bitwise 2-2

- Bảng sau liệt kê các toán tử bit trong JavaScript

Các toán tử bit	Mô tả
& (Bitwise AND)	So sánh 2 bit, trả về 1 nếu cả hai là 1 ngược lại trả về 0
~ (Bitwise NOT)	Đảo ngược một bit
| (Bitwise OR)	So sánh 2 bit, trả về 1 nếu một trong 2 là 1 hoặc cả 2 là 1, còn lại trả về 0

- Ví dụ

```
//(56 = 00111000 and 28 = 00011100)
alert("56" + ' & ' + "28" + ' = ' + (56 & 28));
//(56 = 00111000 and 28 = 00011100)
alert("56" + ' | ' + "28" + ' = ' + (56 | 28));
```

Các toán tử đặc biệt 1-2

Có một số toán tử trong JavaScript mà không thuộc về bất kỳ loại nào gọi là toán tử đặc biệt.

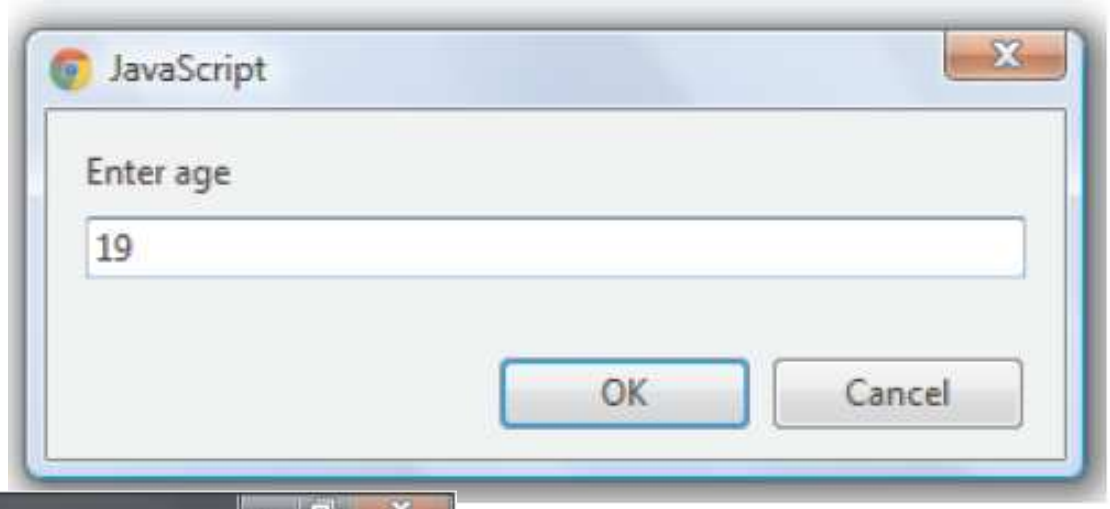
- Bảng sau liệt kê một số toán tử đặc biệt

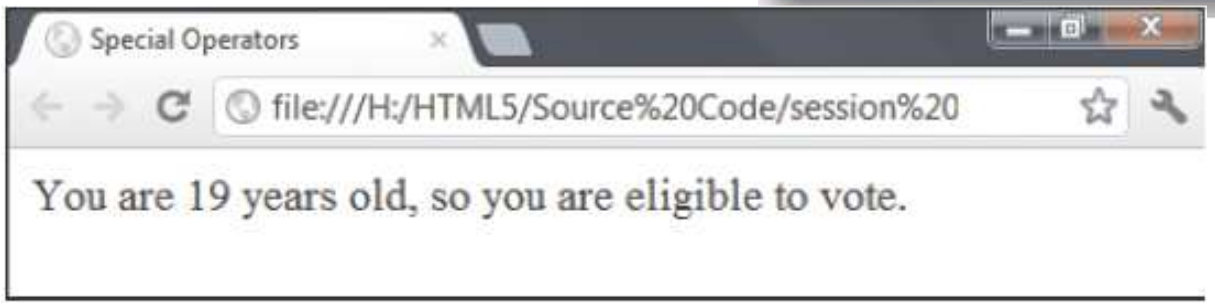
Các toán tử đặc biệt	Mô tả
, (comma)	Kết hợp nhiều biểu thức thành một biểu thức duy nhất, thao tác từ trái qua phải và trả về giá trị của biểu thức bên phải.
?: (conditional)	Thao tác trên ba toán hạng mà kết quả phụ thuộc vào điều kiện. Nó cũng được gọi là toán tử ba ngôi và có dạng điều kiện,? value1: value2. Nếu điều kiện là đúng thì nó trả về value1 ngược lại là value2.
typeof	Trả về một chuỗi cho biết kiểu của các toán hạng. Toán hạng có thể là một chuỗi, biến, từ khóa, hoặc một đối tượng.

Các toán tử đặc biệt 2-2

- Ví dụ

```
<SCRIPT>
  var age = parseInt(prompt("Enter age", "Age"))
  status = ((typeof(age) == "number" && (age >= 18))
  ?"eligible" : "not eligible";
  document.write('You are ' + age + ' years old, so you are '
+status + ' to vote.');
</SCRIPT>
```





Thứ tự của các toán tử

- Bảng sau liệt kê thứ tự các toán hạng và sự liên kết của chúng

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, and Greater than or equal	Left to Right
7	==, ===, !=, !==	Equal to, Strict equal to, Not equal to, and Strict not equal to	Left to Right
8	&, |, ^, &&, ||	Bitwise AND, Bitwise OR, Bitwise XOR, Logical AND, and Logical OR	Left to Right
9	?:	Conditional operator	Right to Left
10	=, +=, -=, *=, /=, %=	Assignment operators	Right to Left
11	,	Comma	Left to Right

Các biểu thức quy tắc

Là một mẫu bao gồm tập hợp các chuỗi, để được khớp với một nội dung văn bản cụ thể.

Cho phép xử lý dữ liệu văn bản có hiệu quả vì nó cho phép tìm kiếm và thay thế chuỗi.

Cho phép xử lý các thao tác phức tạp và xác nhận nếu không có thể được thực hiện thông qua các kịch bản dài.

- Có hai cách tạo biểu thức quy tắc

Cú pháp dạng chuỗi

- var variable_name = /regular_expression_pattern/;

Gọi hàm tạo RegExp()

- var variable_name = new RegExp("regular_expression_pattern","flag");

Các phương thức và thuộc tính của RegEx 1-2

- Đối tượng RegExp hỗ trợ các phương pháp được sử dụng để tìm kiếm các mẫu trong một chuỗi, chúng bao gồm:

test(string) - So sánh một chuỗi cho khớp với một mẫu và trả về một giá trị Boolean true hoặc false. Phương pháp này thường được sử dụng để xác nhận.

exec(string) - Thực thi một chuỗi tìm kiếm mẫu phù hợp với bên trong nó. Phương thức trả về một giá trị null, nếu mẫu không phải là được tìm thấy. Trong trường hợp có nhiều giá trị khớp, nó sẽ trả về tập kết quả khớp.

Các phương thức và thuộc tính của RegEx 2-2

- Ví dụ

```
<SCRIPT>
  var zipcodepattern = /^\d{5}$/;
  var zipcode = zipcodepattern.exec(prompt('Enter ZIP Code:'));
  if(zipcode != null)
  {
    alert('Valid ZIP Code.');
    alert('Regular Expression Pattern: ' + zipcodepattern.source);
   }
  else
  {
    alert('Invalid ZIP Code – Format xxxxx.');
  }
</SCRIPT>
```

Các loại mẫu phù hợp

- Các loại khác nhau của mẫu ký tự khớp với mà được yêu cầu để tạo ra một mẫu

  biểu thức chính quy như sau:
  - Position Matching
  - Character Classes
  - Repetition
  - Alternation and Grouping

Position Matching

Ký tự hoặc ký hiệu trong thể loại này cho phép khớp với một chuỗi tồn tại ở một vị trí cụ thể trong một chuỗi.

- Bảng sau liệt kê các ký hiệu khớp với vị trí khác nhau.

Biểu tượng	Mô tả	Ví dụ
^	Biểu thị sự bắt đầu của một chuỗi	/^Good/ matches “Good” in “Good night”, but not in “A Good Eyesight”
$	Biểu thị kết thúc của một chuỗi	/art$/ matches “art” in “Cart” but not in “artist”
\b	Phù hợp với một từ ranh giới. Một tử ừranh giới bao gồm các vị trí giữa một từ và khoảng trống	/ry\b/ matches “ry” in “She is very good”
\B	Ngược lại so với \b	/\Ban/ matches “an” in “operand” but not in “anomaly”

Character Classes 1-2

Ký tự hoặc ký hiệu trong mục này được kết hợp để tạo thành lớp ký tự cho việc xác định các mẫu.

Các lớp này được hình thành bằng cách đặt một tập hợp các ký tự trong dấu ngoặc vuông.

- Bảng sau liệt kê lớp ký hiệu khác nhau.

Biểu tượng	Mô tả	Ví dụ
[xyz]	Phù hợp với một trong những ký tự được quy định trong bộ ký tự	/^Good/ matches “Good” in “Good night”, but not in “A Good Eyesight”
[^xyz]	Phù hợp với một trong những ký tự không quy định trong bộ ký tự	/[^BC]RT/ Matches “RRT” but, not “BRT” or “CRT”
.	Biểu thị một ký tự ngoại trừ các dòng mới và ký tự ngắt dòng.	/s.t/ Matches “sat”, “sit”, “set”, and so on
\w	Phù hợp với bảng chữ cái và chữ số cùng với gạch dưới	/\w/ Matches “600” in “600%”

Character Classes 2-2

Biểu tượng	Mô tả	Ví dụ
\W	Phù hợp với một biểu tượng không không phải ký tự.	/\W/ Matches “%” in “800%”
\d	Phù hợp với một chữ số giữa 0-9	/\d/ Matches “4” in “A4”
\D	Phù hợp với một biểu tượng không phải chữ số	/\D/ Matches “ID” in “ID 2246”
\s	Tìm kiếm bất kỳ ký tự khoảng trắng duy nhất bao gồm cả khẳng trắng, tab,…	/\s\w*/ Matches “ bar” in “scroll bar”
\S	Ngược so với \s	/\S\w*/ Matches “scroll” in “scroll bar”

Repetition

Ký tự hoặc ký hiệu trong mục này cho phép ký tự phù hợp mà xuất hiện thường xuyên trong một chuỗi.

- Bảng sau liệt kê các biểu tượng phù hợp với mẫu lặp lại khác nhau.

Biểu tượng	Mô tả	Ví dụ
{x}	Phù hợp với x số lần xuất hiện của một biểu thức quy tắc	/\d{6}/ Matches exactly 6 digits”
{x, }	Phù hợp với tối thiểu x lần xuất hiện của một biểu thức quy tắc	/\s{4,}/ Matches minimum 4 whitespace characters
{x,y}	Phù hợp với với số lần xuất hiện trong đoạn từ x-y của một biểu thức chính quy	/\d{6,8}/ Matches minimum 6 to maximum 8 digits
?	Phù hợp với số lần xuất hiện tối thiểu là 0 tối đa là 1	/l\s?m/ Matches “lm” or “l m”
*	Phù hợp với số lần xuất hiện tối thiểu là 0 tối đa vô cùng	/im*/ Matches “i” in “Ice” and “imm” in “immaculate”, but nothing in “good”

Alternation and Grouping

Ký tự hoặc ký hiệu trong mục này cho phép nhóm ký tự như một thực thể cá nhân hoặc thêm logic 'OR' cho mẫu phù hợp.

- Bảng dưới đây liệt kê các thay đổi luân phiên và nhóm các biểu tượng ký tự khác nhau.

Biểu tượng	Mô tả	Ví dụ
()	Tổ chức các ký tự với nhau trong một nhóm để xác định một tập hợp các ký tự trong một chuỗi	/(xyz)+(uvw)/ Matches one or more number of occurrences of “xyz” followed by one occurrence of “uvw”
|	Kết hợp bộ ký tự vào một biểu thức quy tắc duy nhất và sau đó phù hợp với bất kỳ ký tự	/(xy)|(uv)|(st)/ Matches “xy” or “uv” or “st”

Các câu lệnh quyết định 1-2

Câu lệnh được gọi là một tập hợp logic của các biến, các toán tử, và từ khóa mà thực hiện một hành động cụ thể để thực hiện một nhiệm vụ bắt buộc.

Câu lệnh giúp bạn xây dựng một luồng hợp lý của kịch bản.

Trong JavaScript, một câu lệnh kết thúc bằng một dấu chấm phẩy.

JavaScript được viết với nhiều câu lệnh, trong đó các câu lệnh có liên quan được nhóm lại với nhau được gọi là khối mã và được kèm theo trong dấu ngoặc xoắn.

Câu lệnh đưa ra quyết định cho phép thực hiện các quyết định hợp lý để thực hiện các khối khác nhau để có được những kết quả mong muốn.

Chúng thực hiện một khối lệnh phụ thuộc vào một điều kiện Boolean trả về true hoặc false.

Các câu lệnh quyết định 2-2

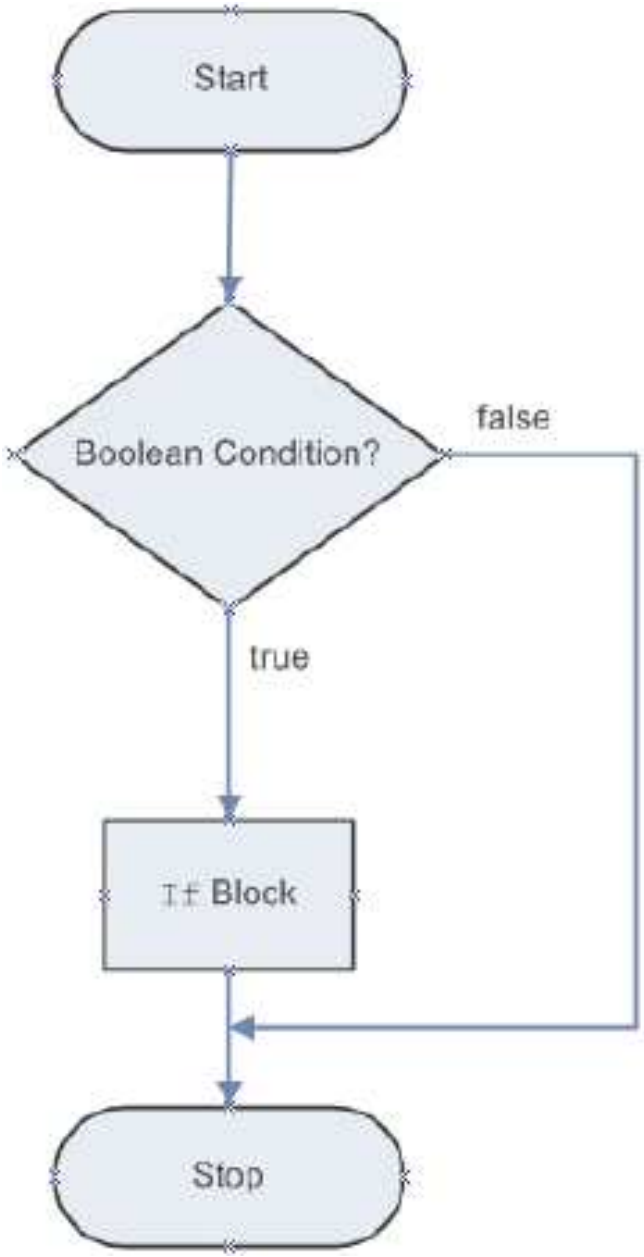
- JavaScript hỗ trợ bốn câu quyết định, cụ thể như sau:
  - `if`
  - `if-else`
  - `if-else if`
  - `switch`

Câu lệnh if 1-2

Thực hiện một khối lệnh dựa trên điều kiện Boolean logic.

Nếu điều kiện này là đúng, khối theo sau câu lệnh if được thực thi.

Nếu điều kiện là sai, khối sau khi câu lệnh if không được thực hiện và câu lệnh ngay lập tức sau khi khối được thực thi.



Câu lệnh if 2-2

- Ví dụ.

```
<SCRIPT>
  var quantity = prompt('Enter quantity of product:',0);
  if(quantity < 0 || isNaN(quantity))
  {
    alert('Please enter a positive number.');
  }
</SCRIPT>
```

Câu lệnh if-else 1-2

Câu lệnh if xác định một khối câu lệnh được thực hiện khi điều kiện trong câu lệnh if là đúng.

Đôi khi cần định nghĩa một khối lệnh được thực hiện khi điều kiện được đánh giá là sai.

Câu lệnh if-else bắt đầu với khối if, tiếp theo là các khối else.

Khối else bắt đầu bằng từ khóa else theo sau là một khối lệnh được thực hiện khi điều kiện sai.

Câu lệnh if-else 2-2

- Ví dụ.

```
<SCRIPT>
  var firstNumber = prompt('Enter first number:',0);
  var secondNumber = prompt('Enter second number',0);
  var result = 0;
  if (secondNumber == 0)
  {
    alert('ERROR Message: Cannot divide by zero.');
  }
  else
  {
    result = firstNumber/secondNumber;
    alert("Result: " + result);
  }
</SCRIPT>
```

Câu lệnh if-else-if 1-2

- Cho phép bạn kiểm tra nhiều điều kiện và chỉ định một khối khác nhau được thực hiện cho từng trường hợp.
- Luồng thực thi của câu lệnh bắt đầu với câu lệnh if tiếp theo nhiều else if và cuối cùng một khối else tùy chọn.
- Điểm vào thực hiện trong các báo cáo bắt đầu với câu lệnh if.
- Nếu điều kiện trong câu lệnh if là sai, điều kiện trong else if được đánh giá.
- Cấu trúc này còn được gọi là cấu trúc bậc thang.

Câu lệnh if-else-if 2-2

- Ví dụ

```
<SCRIPT>
  var percentage = prompt('Enter percentage:',0);
  if (percentage >= 60)
  {
    alert ('You have obtained the A grade.');
  }
  else if (percentage >= 35 && percentage < 60)
  {
    alert ('You have obtained the B class.');
  }
  else
  {
    alert ('You have failed');
  }
</SCRIPT>
```

Câu lệnh if lồng nhau 1-2

Bao gồm nhiều câu lệnh if trong một câu lệnh if.

Luồng thực hiện của các câu lệnh if lồng nhau bắt đầu với câu lệnh if, được gọi là câu lệnh if bên ngoài.

Câu lệnh if bên trong được thực thi khi câu lệnh if bên ngoài đúng.

Mỗi câu lệnh bên trong được thực thi khi câu lệnh if trước đó là đúng.

Câu lệnh if lồng nhau 2-2

- Ví dụ.

```
<SCRIPT>
  var username = prompt('Enter Username:');
  var password = prompt('Enter Password:');
  if (username != "" && password != "")
  {
    if (username == "admin" && password == "admin123")
    {
      alert('Login Successful');
    }
    else
    {
     alert ('Login Failed');
    }
  }
</SCRIPT>
```

Câu lệnh switch-case 1-2

Một chương trình trở nên khá khó hiểu khi có nhiều câu lệnh if.

Để đơn giản hóa mã và tránh sử dụng nhiều câu lệnh if, câu lệnh switch case có thể được sử dụng.

Câu lệnh switch-case cho phép so sánh một biến hoặc biểu thức với nhiều giá trị.

Câu lệnh switch-case 2-2

- Ví dụ

```
<SCRIPT>
  var designation = prompt('Enter designation:');
  switch (designation)
  {
    case 'Manager':
      alert ('Salary: $21000');
      break;
    case 'Developer':
      alert ('Salary: $16000');
      break;
    default:
      alert ('Enter proper designation.');
      break;
  }
</SCRIPT>
```

Tổng kết

- Một toán tử quy định cụ thể loại thao tác được thực hiện trên các giá trị của biến và biểu thức.
- Các toán tử JavaScript được phân thành sáu loại dựa trên loại hành động chúng thực hiện trên các toán hạng.
- Có 6 loại toán tử có tên như sau, Arithmetic, Relational, Logical, Assignment, Bitwise, và Special operators.
- Toán tử trong JavaScript có mức độ ưu tiên nhất định dựa trên tự thực hiện của chúng được xác định.
- Một biểu thức quy tắc là một mẫu bao gồm tập hợp các chuỗi mà chúng được khớp với một nội dung văn bản cụ thể.
- Trong JavaScript, có hai cách để tạo ra biểu thức thông thường cụ thể là, dùng chuỗi biểu thức và hàm RegExp().
- Câu lệnh đưa ra quyết định cho phép thực hiện các quyết định hợp lý để thực hiện các khối khác nhau để có được những kết quả mong muốn.