Syntax

1 JavaScript – Operators

1.1 What is an operator?

在表達式中 **4 + 5** 4 和 5 稱為**operands** ，而 + 是 **operator**.

- Arithmetic Operators
- Comparision Operators
- Logical (or Relational) Operators
- Assignment Operators
- Conditional (or ternary) Operators

1.2 Arithmetic Operators

JavaScript 支援的數學運算子:

假定A是10，B是20.

Sr.No	Operator and Description
1	+ (加) Ex: A + B 結果30
2	- (減) Ex: A - B will give -10
3	* (乘) Ex: A * B will give 200
4	/ (除) Divide the numerator by the denominator Ex: B / A will give 2
5	% (Modulus) Ex: B % A will give 0
6	++ (Increment) Ex: A++ will give 11

7	-- (Decrement) Ex: A-- will give 9

Note – + 這個運算元除了用於數字外，如果用在字串上，則意義是串連，例如，"a" + 10 結果為 "a10".

Example

The following code shows how to use arithmetic operators in JavaScript.

```
<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);

            document.write("a - b = ");
            result = a - b;
            document.write(result);
            document.write(linebreak);

            document.write("a / b = ");
            result = a / b;
            document.write(result);
            document.write(linebreak);

            document.write("a % b = ");
            result = a % b;
            document.write(result);
            document.write(linebreak);

            document.write("a + b + c = ");
            result = a + b + c;
            document.write(result);
            document.write(linebreak);

            a = ++a;
            document.write("++a = ");
            result = ++a;
            document.write(result);
            document.write(linebreak);

            b = --b;
            document.write("--b = ");
            result = --b;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>
```

```
     Set the variables to different values and then try...
   </body>
</html>
```

```
Output
a + b = 43
a - b = 23
a / b = 3.3
a % b = 3
a + b + c = 43Test
++a = 35
--b = 8
```

semantic: 數字 + 文字 會被自動設定為 轉換成文字（數字） + 文字。

1.3 Comparison Operators

比較運算子，參看下表，假定假定A是10，B是20.

Sr.No	Operator and Description
1	= = (Equal) : 比較兩邊的運算元是否相等。Ex: (A == B) is not true. <參考> = = 會對兩邊做型別轉換，然後比較。而===則不做類別轉換。例如，a="12",b=12 ，則 a==b 和 a===b 結果不一樣。前者是true,後者是false。
2	!= (Not Equal) Checks if the value of two operands are equal or not, if the values are not equal, then the condition becomes true. Ex: (A != B) is true.
3	> (Greater than) Checks if the value of the left operand is greater than the value of the right operand, if yes, then the condition becomes true. Ex: (A > B) is not true.
4	< (Less than) Checks if the value of the left operand is less than the value of the right operand, if yes, then the condition becomes true. Ex: (A < B) is true.
5	>= (Greater than or Equal to) Checks if the value of the left operand is greater than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A >= B) is not true.
6	<= (Less than or Equal to)

	Checks if the value of the left operand is less than or equal to the value of the right operand, if yes, then the condition becomes true. Ex: (A <= B) is true.

Example

The following code shows how to use comparison operators in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write("(a == b) => ");
            result = (a == b);
            document.write(result);
            document.write(linebreak);

            document.write("(a < b) => ");
            result = (a < b);
            document.write(result);
            document.write(linebreak);

            document.write("(a > b) => ");
            result = (a > b);
            document.write(result);
            document.write(linebreak);

            document.write("(a != b) => ");
            result = (a != b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >= b) => ");
            result = (a >= b);
            document.write(result);
            document.write(linebreak);

            document.write("(a <= b) => ");
            result = (a <= b);
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      Set the variables to different values and different operators and then try...
   </body>
</html>
```

```
Output
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
```

```
(a >= b) => false
a <= b) => true
```

1.4 Logical Operators

JavaScript supports the following logical operators −
Assume variable A holds 10 and variable B holds 20, then −

Sr.No	Operator and Description
1	&& (Logical AND) If both the operands are non-zero, then the condition becomes true. Ex: (A && B) is true.
2	|| (Logical OR) If any of the two operands are non-zero, then the condition becomes true. Ex: (A || B) is true.
3	! (Logical NOT) Reverses the logical state of its operand. If a condition is true, then the Logical NOT operator will make it false. Ex: ! (A && B) is false.

求值的最短路徑

由於邏輯表達式是由左往右求值，計算結果時，儘可能以“最短路徑”求值：

`false` && *anything* 中的 false 是求值的最短路徑。

`true` || *anything* 中的 true 是求值的最短路徑。

Example

Try the following code to learn how to implement Logical Operators in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = true;
            var b = false;
            var linebreak = "<br />";

            document.write("(a && b) => ");
            result = (a && b);
            document.write(result);
            document.write(linebreak);

            document.write("(a || b) => ");
            result = (a || b);
            document.write(result);
            document.write(linebreak);

            document.write("!(a && b) => ");
            result = (!(a && b));
            document.write(result);
```

```
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
  try...</p>
   </body>
</html>
```

```
Output
(a && b) => false
(a || b) => true
!(a && b) => true
```

1.5 Bitwise Operators

如果A=2,B=3

Sr.No	Operator and Description
1	& (Bitwise AND) Ex: (A & B) is 2. A=00000010 B=00000011
2	| (BitWise OR) Ex: (A | B) is 3.
3	^ (Bitwise XOR) 不是兩個都一樣，則1。兩個都一樣是0。也就是 0 ^ 0 => 0 0 ^ 1 => 1 1 ^ 0 => 1 1 ^ 1 => 0 Ex: (A ^ B) is 1.
4	~ (Bitwise Not) Ex: (~B) is -4. note: logical not "!"
5	<< (Left Shift) 左移填零。 Ex: 5 << 1 結果為10　　因為，0101 << 1 結果 1010 Ex: (A << 1) is 4.
6	>> (Right Shift) 右移的時候，以最左邊的值填入空缺。 Ex: (A >> 1) is 1. Ex: -1>>1 結果 -1 解釋 53 --> 00110101 -53 --> 11001010 + 1 = 11001011 1->0001 (4位元為例)

	所以 -1->1110+1 以32bit為例 -1 -> 11111111111111111111111111111111 -1>>1 右移1位 ?11111111111111111111111111111111 在問號上，補1.
7	>>> (右移填零) Ex: (A >>> 1) is 1. Ex: -1>>>1 結果2147483647 (32bit chrome) 01111111111111111111111111111111

Example

Try the following code to implement Bitwise operator in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 2; // Bit presentation 10
            var b = 3; // Bit presentation 11
            var linebreak = "<br />";

            document.write("(a & b) => ");
            result = (a & b);
            document.write(result);
            document.write(linebreak);

            document.write("(a | b) => ");
            result = (a | b);
            document.write(result);
            document.write(linebreak);

            document.write("(a ^ b) => ");
            result = (a ^ b);
            document.write(result);
            document.write(linebreak);

            document.write("(~b) => ");
            result = (~b);
            document.write(result);
            document.write(linebreak);

            document.write("(a << b) => ");
            result = (a << b);
            document.write(result);
            document.write(linebreak);

            document.write("(a >> b) => ");
            result = (a >> b);
            document.write(result);
            document.write(linebreak);
```

```
        //-->
     </script>

     <p>Set the variables to different values and different operators and then
  try...</p>
   </body>
</html>
```

```
(a & b) => 2
(a | b) => 3
(a ^ b) => 1
(~b) => -4 //<1>
(a << b) => 16
(a >> b) => 0
```

1
0100
1011
1100--> -4
0011->3

<註>

(2).toString(2) 轉成2進位

</註>

1.6 Assignment Operators

JavaScript supports the following assignment operators −

Sr.No	Operator and Description
1	= (Simple Assignment) Assigns values from the right side operand to the left side operand Ex: C = A + B will assign the value of A + B into C
2	+= (Add and Assignment) It adds the right operand to the left operand and assigns the result to the left operand. Ex: C += A is equivalent to C = C + A
3	−= (Subtract and Assignment) It subtracts the right operand from the left operand and assigns the result to the left operand. Ex: C -= A is equivalent to C = C - A

4	*= (Multiply and Assignment) It multiplies the right operand with the left operand and assigns the result to the left operand. Ex: C *= A is equivalent to C = C * A
5	/= (Divide and Assignment) It divides the left operand with the right operand and assigns the result to the left operand. Ex: C /= A is equivalent to C = C / A
6	%= (Modules and Assignment) It takes modulus using two operands and assigns the result to the left operand. Ex: C %= A is equivalent to C = C % A

Note − Same logic applies to Bitwise operators so they will become like <<=, >>=, >>=, &=, |= and ^=.

Example

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 33;
            var b = 10;
            var linebreak = "<br />";

            document.write("Value of a => (a = b) => ");
            result = (a = b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a += b) => ");
            result = (a += b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a -= b) => ");
            result = (a -= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a *= b) => ");
            result = (a *= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a /= b) => ");
            result = (a /= b);
            document.write(result);
            document.write(linebreak);

            document.write("Value of a => (a %= b) => ");
            result = (a %= b);
            document.write(result);
            document.write(linebreak);
         //-->
```

```
      </script>

      <p>Set the variables to different values and different operators and then
  try...</p>
   </body>
</html>
```

```
Output
Value of a => (a = b) => 10
Value of a => (a += b) => 20
Value of a => (a -= b) => 10
Value of a => (a *= b) => 100
Value of a => (a /= b) => 10
Value of a => (a %= b) => 0
Set the variables to different values and different operators and then try...
```

1.7 Miscellaneous Operator

We will discuss two operators here that are quite useful in JavaScript: the **conditional operator** (? :) and the **typeof operator**.

Conditional Operator (? :)

The conditional operator first evaluates an expression for a true or false value and then executes one of the two given statements depending upon the result of the evaluation.

Sr.No	Operator and Description
1	? : (Conditional) If Condition is true? Then value X : Otherwise value Y

Example

Try the following code to understand how the Conditional Operator works in JavaScript.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 10;
            var b = 20;
            var linebreak = "<br />";

            document.write ("((a > b) ? 100 : 200) => ");
            result = (a > b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);

            document.write ("((a < b) ? 100 : 200) => ");
            result = (a < b) ? 100 : 200;
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
```

```
  try...</p>
   </body>
</html>
```

```
Output
((a > b) ? 100 : 200) => 200
((a < b) ? 100 : 200) => 100
Set the variables to different values and different operators and then try...
```

1.8 typeof Operator

The **typeof** operator is a unary operator that is placed before its single operand, which can be of any type. Its value is a string indicating the data type of the operand.
The *typeof* operator evaluates to "number", "string", or "boolean" if its operand is a number, string, or boolean value and returns true or false based on the evaluation.
Here is a list of the return values for the **typeof** Operator.

Type	String Returned by typeof
Number	"number"
String	"string"
Boolean	"boolean"
Object	"object"
Function	"function"
Undefined	"undefined"
Null	"object"

Example

The following code shows how to implement **typeof** operator.

```
<html>
   <body>

      <script type="text/javascript">
         <!--
            var a = 10;
            var b = "String";
            var linebreak = "<br />";

            result = (typeof b == "string" ? "B is String" : "B is Numeric");
            document.write("Result => ");
            document.write(result);
            document.write(linebreak);

            result = (typeof a == "string" ? "A is String" : "A is Numeric");
            document.write("Result => ");
```

```
            document.write(result);
            document.write(linebreak);
         //-->
      </script>

      <p>Set the variables to different values and different operators and then
  try...</p>
   </body>
</html>
```

```
Output
Result => B is String
Result => A is Numeric
Set the variables to different values and different operators and then try.
```

問題

6 ^ 3 為多少

<註>

6 ^ 3, meaning 110 XOR 011 gives 101 (5).

</註>