Introduction to Internet and Web

Taewoon Kim

Computer Science and Engineering

Outline

- JavaScript (JS) … continued
  - Introduction
  - Inserting JS code into web page
  - Data types and variables
  - Operators
  - Conditional statements
  - Iteration / loops
  - Function

Last class we learned…

- Writing JS code
- Generating HTML contents : document.write("msg")
- Three dialogs : prompt, confirm, alert
- Data types
- Variables
  - Scope and life of global, local, and block variables
  - Declaring variables with var and let

```
var score;                  // declare variable "score"
var year, month, day;       // declare three variables: year, month, day
var address = "Busan";      // declare variable "address", and initialize with "Busan"
```

```
let score;                  // declare variable "score"
let year, month, day;       // declare three variables: year, month, day
let address = "Busan";      // declare variable "address", and initialize with "Busan"
```

JavaScript (JS)

- Declaring variables with let and var
  - Both var and let are used to declare variables
  - Let is added later to avoid possible mistakes that can happen with var
- let prevents re-declaration

```
var x = 1;
var x = 2; // Allowed
           // previously defined x is
           // removed, and a new x is
           // declared
           // this, possibly, is a mistake
```

```
let x = 1;
let x = 2; // NOT allowed.
           // i.e., re-declaration of the same
           // identifier is not allowed
```

* let is preferred over var to prevent re-declaration mistakes

- let introduces a new scope : block scope

```
if(a == b) { // beginning of a block
   let x = 10; // x has block scope (if)
} // end of block
x = x + 1; // Now allowed
```

```
for(let n=0; n<10; n++) { // begin. of block
   let x = 10; // n and x have block scope (for)
} // end of block
x = x + 1; // Not allowed
n = n + 1; // Now allowed
```

JavaScript (JS)

- Constant variables (= constants)
  - Special variables that do not change values after initiation

```
const MAX = 10;  // constant MAX is initialized to 10
```

  - Update of value is not allowed

```
const MAX = 10;
MAX = 20; // Change of value is not allowed
```

  - Re-declaration is not allowed

```
const MAX = 10;
...
const MAX = 10; // Re-declaration is not allowed
```

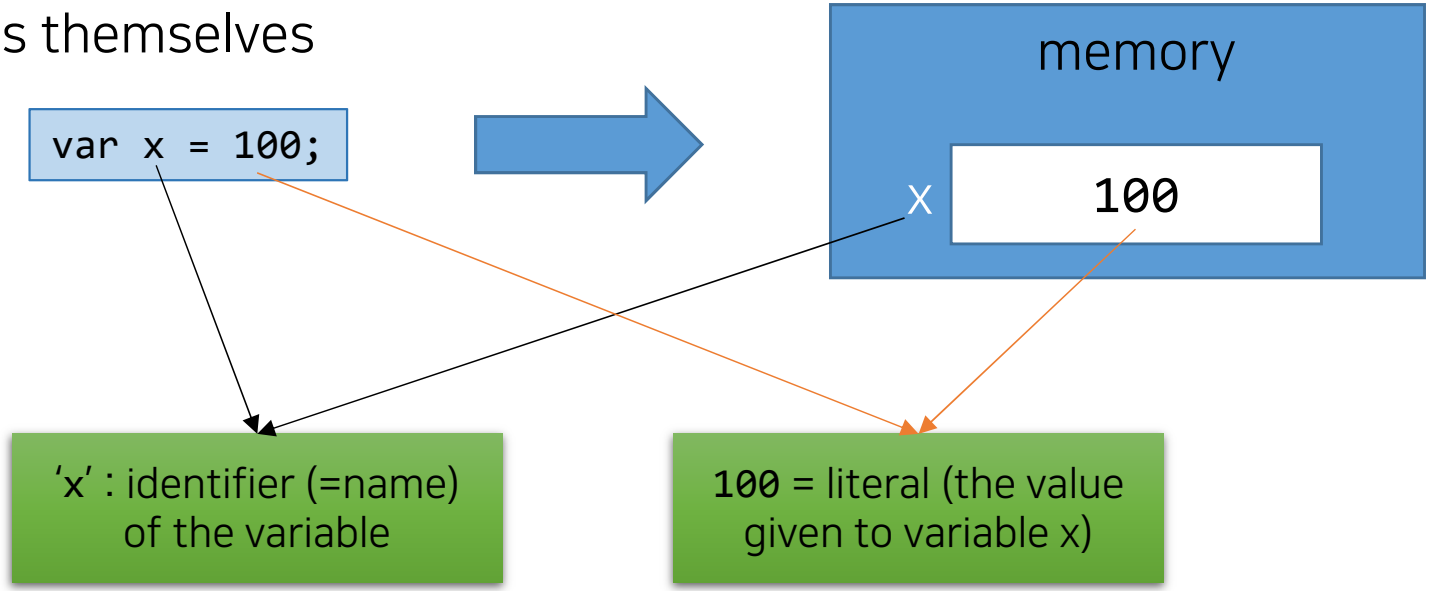
  - Scope is limited

```
if(a == b) {
  const MAX = 10;
  ...
}
let n = MAX; // Cannot access
             // outside the block
```

```
for(let n=0; n<10; n++) {
  const MAX = 10;
  ...
}
let m = MAX; // Cannot access
             // outside the block
```

JavaScript (JS)

- Literal : values themselves



- Examples:
  - integer literals : -100, 0, 50, ...
  - floating point number literals : -1.1, 3.14, ...
  - logical literals : true, false
  - string literals : "hello, yo", 'nice to meet you', ... (either " " and ' ' can be used)
  - etc. : null (i.e., absent), NaN (= not a number)

JavaScript (JS)

- JS operation and operators

operation	operator	operation	operator
arithmetic	+ - * / %	assign	= *= /= += -= &= ^= |= <<= >>= >>>=
increase/decrease	++ --	comparison	> < >= <= == !=
bit	& | ^ ~	logical	&& || !
shift	>> << >>>	conditional	? :

- Arithmetic operations
  - add(+), subtract(-), multiply(*), division(/), remainder(%)

```
let x = 32;
let total = 100 + x*2/4 - 3;  // total = 113
```

Simple quiz:
How to check if a number is even/odd?

  - Results are always real numbers

```
let div = 32/10;              // div = 3.2
```

JavaScript (JS)

- Arithmetic operations

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Arithmetic</title>
</head>
<body>
<h3> Arithmetic </h3>
<hr>
<script>
   let x=32;
   let total = 100 + x*2/4 - 3; // total = 113
   let div = x / 10; // div는 3.2
   let mod = x % 2; // remainder of x / 2 = 0

   document.write("x : " + x + "<br><br>");
   document.write("100 + x*2/4 - 3 = " + total + "<br>");
   document.write("x/10 = " + div + "<br>");
   document.write("x%2 = " + mod + "<br>");
</script>
</body>
</html>
```

Arithmatic

x : 32

100 + x*2/4 - 3 = 113
x/10 = 3.2
x%2 = 0

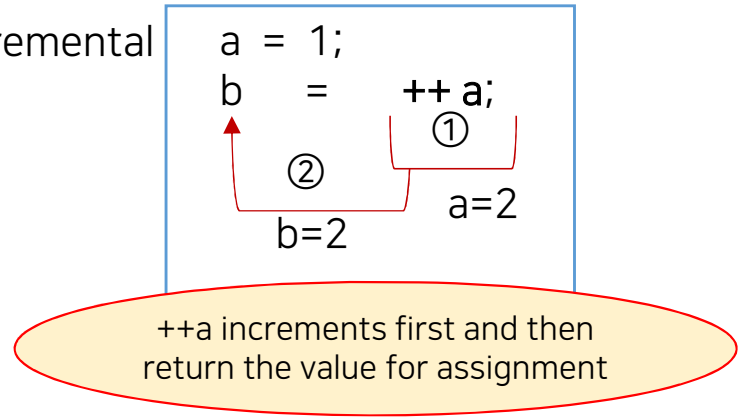
JavaScript (JS)

- Incremental, decremental operations : `++, --`
  - `++` is used to increase the number by 1
    - e.g.,

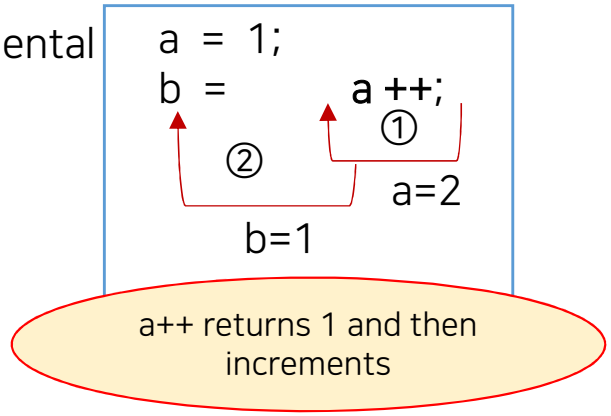
Both are the same	
`var a = 0;` `a++;`	`var a = 0;` `a = a + 1;` // increasement & assignment

    - NOTE: `++` and `--` operate in the same manner, so we focus on ++ in this lecture note
  - pre-increment and post-increment

(a) pre-incremental operation



(b) post-incremental operation



operator	operation	operator	operation
`a++`	returns a and then adds 1 to a	`++a`	adds 1 to a and then return a
`a--`	returns a and the subtract 1 from a	`--a`	subtracts 1 from a and then return a

JavaScript (JS)

- Incremental, decremental operations
  - Example

```html
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript Variables</h1>

<script>
var x = 10;
var y = 20;
document.write("x++ : " + (x++) + "<br>");
document.write("x : " + x);

document.write("<br><br>");

document.write("++y : " + (++y) + "<br>");
document.write("y : " + y);

</script>

</body>
</html>
```

JavaScript Variables

x++ : 10
x : 11

++y : 21
y : 21

JavaScript (JS)

- Assignment operation (a = b)
  - assign right-hand side results to the variable on the left-hand side

```
let a=1, b=3;
a = b;     // a becomes 3 now
a += b;    // equals a = a + b, and thus a becomes 6
```

- Shortened assignment and operation

operator	operation	operator	operation
a = b	assign b to a	a &= b	a = a & b
a += b	a = a + b	a ^= b	a = a ^ b
a −= b	a = a − b	a |= b	a = a | b
a *= b	a = a * b	a <<= b	a = a << b
a /= b	a = a / b	a >>= b	a = a >> b
a %= b	a = a % b	a >>>= b	a = a >>> b

JavaScript (JS)

- Assignment example

```
6  <body>
7  <h3>Assignment</h3>
8  <hr>
9  <script>
10     let x=3, y=3, z=3;
11     document.write("x=" + x + ", y=" + y);
12     document.write(", z=" + z + "<br><br>");
13
14     x += 3; // x=x+3 -> x=6
15     y *= 3; // y=y*3 -> y=9
16     z %= 2; // z=z%2 -> z=1
17
18     document.write("x += 3 >>>> x=" + x + "<br>");
19     document.write("y *= 3 >>>> y=" + y + "<br>");
20     document.write("z %= 2 >>>> z=" + z);
21 </script>
22 </body>
```

Assignment

x=3, y=3, z=3

x += 3 >>>> x=6
y *= 3 >>>> y=9
z %= 2 >>>> z=1

JavaScript (JS)

- Comparison
  - Compares two values, and returns true or false

```
let age = 25;
let result = (age > 20); // true will be assigned to result variable
```

- Comparison operators

operator	operation	operator	operation
`a < b`	returns true if a is less than b	`a >= b`	returns true if a is greater than or equal to b
`a > b`	returns true if a is greater than b	`a == b`	returns true if a equals b
`a <= b`	returns true if a is less than or equal to b	`a != b`	returns true if a is not equal to b

JavaScript (JS)

- Comparison example:

```
6   <body>
7   <h3>Comparison</h3>
8   <hr>
9   <script>
10      let x=13, y=7;
11      document.write("x=" + x + ", y=" + y + "<br><br>");
12      document.write("x == y : " + (x == y) + "<br>");
13      document.write("x != y : " + (x != y) + "<br>");
14      document.write("x >= y : " + (x >= y) + "<br>");
15      document.write("x > y : " + (x > y) + "<br>");
16      document.write("x <= y : " + (x <= y) + "<br>");
17      document.write("x < y : " + (x < y) + "<br>");
18  </script>
19  </body>
```

Comparison

x=13, y=7

x == y : false
x != y : true
x >= y : true
x > y : true
x <= y : false
x < y : false

JavaScript (JS)

- Logical operation : AND (&&), OR (||), NOT (!)

Operator		Operation
a && b	logical AND	• returns true if both a and b are true • returns false otherwise
a || b	logical OR	• returns true if at least one of a and b is true • returns false other wise (i.e., only when both a and b are false)
!a	logical NOT	• returns false if a is true • returns true if a is false

- AND, OR : takes in logical values on both sides and returns the resulting logical value
- NOT : takes in only a single logical value and returns the negation of it
- Example:

```
let score = 90;
let age = 20;
let res = ((score > 80) && (age < 25));      // res=true
```

JavaScript (JS)

- Logical operation example :

```
<script>
    let x=true; y=false;
    document.write("x=" + x + ", y=" + y + "<br><br>");
    document.write("x && y : "+ (x&&y) +"<br>");
    document.write("x || y : "+ (x||y) +"<br>");
    document.write("!x : " + (!x) +"<br>");
    document.write("<hr>");
    document.write("(3>2) && (3<4) : " + ((3>2)&&(3<4)) + "<br>");
    document.write("(3==-2) || (-1<0) : " + ((3==2)||(-1<0)));
</script>
```

x=true, y=false

x && y : false
x || y : true
!x : false

(3>2) && (3<4) : true
(3==-2) || (-1<0) : true

JavaScript (JS)

- String operations
  - String concatenation : +
    - string1 + string2
    - string3 = string1 + string2

```
<h1>JavaScript Variables</h1>

<script>
var firstname = "Daniel";
var lastname = "Kim";
var fullname = firstname + " " + lastname;

document.write(firstname + lastname);
document.write("<br>");

document.write(firstname + " " + lastname);
document.write("<br>");

document.write(fullname);
document.write("<br>");

document.write(12 + 34); document.write("<br>");
document.write("12" + "34"); document.write("<br>");
</script>
```

adding a space should be done manually

DanielKim
Daniel Kim
Daniel Kim
46
1234

- upper one is adding two integer numbers, while
- lower one is adding two strings

JavaScript (JS)

- String operations
  - mixing with integers
    - NOTE: addition occurs from left to right

```
<h1>JavaScript Variables</h1>

<script>
document.write("Pizza" + 9900);
document.write("<br>");

document.write(9900 + "Pizza");
document.write("<br>");

document.write("Pizza" + 9900 + 99);
document.write("<br>");

document.write(99 + 9900 + "Pizza");
document.write("<br>");

</script>
```

```
Pizza9900
9900Pizza
Pizza990099
9999Pizza
```

- When a string comes first, the following integer (9900) is added as a string (data type is casted from integer to string)
- The resulting term is a string, and thus, the following integer (99) is added as a string (type casting to string)

- The first two terms are integers, and the addition results in an integer sum.
- Finally, the sum of integer and string becomes the sum of two strings “9999” and “Pizza“, where “9999” is automatically type casted from integer to string

JavaScript (JS)

- String operations
  - comparison operator : `!=, ==, > , <, <=, >=`
  - comparison will be made lexicographically (alphabetically)

```
<h1>JavaScript Variables</h1>

<script>
document.write("Pizza" == "Pizza");
document.write("<br>");

document.write("Absolute" > "Banana");
document.write("<br>");

document.write("Canada" < "Denmark");
document.write("<br>");

</script>
```

true
false
true

- "A" comes before "B", then "A" is smaller than "B"
- "D" comes after "C", then "D" is larger than "C"

  - length
    - string length : string1.length

```
<script>
document.write("Pizza".length);
document.write("<br>");
</script>
```

- returns 5 (integer)

JavaScript (JS)

- Conditional statement
  - Used to perform different actions based on different conditions.
  - JS provides three ways to implement conditional statement
    - `?` (conditional operator)
    - `if-else`
    - `switch-case`

  - ? (conditional operator)
    - syntax : `condition ? expression_for_true : expression_for_false`
      - if "condition" is true, `expr_for_true` will be evaluated and the result will be returned
      - Otherwise, `expr_for_false` will be evaluated and the result will be returned

```
<script>
var v = 13;
var r = v > 10 ? v - 20 : v + 30;
document.write("Result (r) : " + r);
document.write("<br>");
</script>
```

- The given condition (`v > 10`) is true
- Thus, `v-20` will be evaluated, and the result will be returned (i.e., result will be assigned to `r`)

Result (r) : -7

JavaScript (JS)

- Conditional statement
  - `if`
    - syntax :

```
if( condition ) {
  … statement(s); // executed if condition is true
}
```

```
if( a > b ) {
  document.write("a is greater");
}
```

- "condition" is a condition to be evaluated to be either true or false
- When "condition" is true, the statement(s) within the following block will be executed.
- Otherwise, the block of code will not be executed

  - `if-else`
    - has the else block which is executed when the condition is false

```
if( condition ) {
  … statement(s); // when condition == true
}
else {
  … statement(s); // otherwise
}
```

```
if( a > b ) {
  document.write("a is greater");
}
else {
  document.write("a is NOT greater");
}
```

JavaScript (JS)

- Conditional statement
  - multiple `if-else`

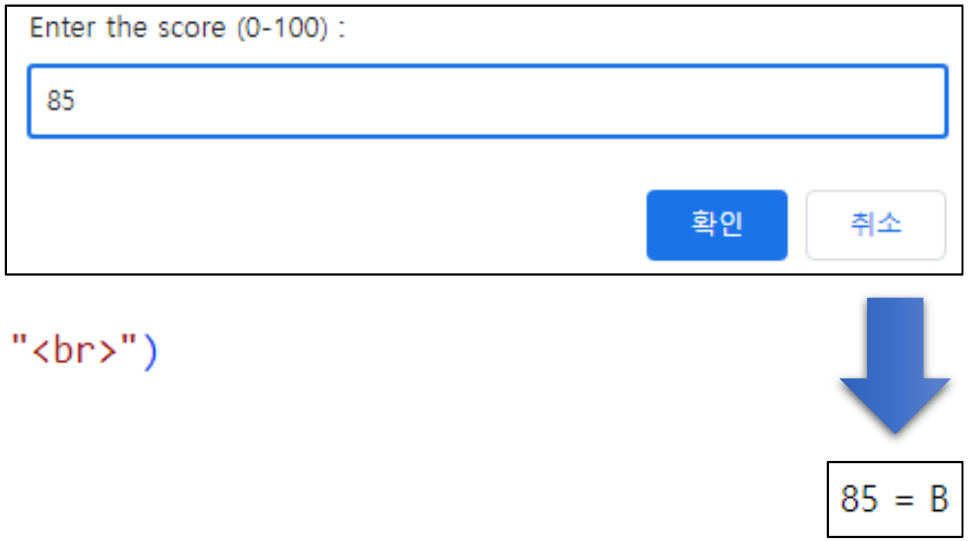
```
if( condition_1) {
  statement(s); // when condition_1 == true
}
else if( condition_2 ) {
  statement(s); // when condition_2 == true
}
... there can be more else-if blocks
else {
  statement(s); // none of the above is true
}
```

```
if( a > b ) {
  document.write("a is greater");
}
else if( a < b ) {
  document.write("b is greater");
}
else
  document.write("a is equal to b");
```

JavaScript (JS)

- Conditional statement : example

```
9   <script>
10      let grade;
11      let score = prompt("Enter the score (0-100) : ", 100);
12      score = parseInt(score); // change data type : string >> integer
13      if(score >= 90)
14          grade = "A";
15      else if(score >= 80)
16          grade = "B";
17      else if(score >= 70)
18          grade = "C";
19      else if(score >= 60)
20          grade = "D";
21      else // < 60
22          grade = "F";
23      document.write(score + " = " + grade + "<br>")
24  </script>
```



JavaScript (JS)

- Conditional statement
  - `switch-case`
    - similar to multiple if-else
    - syntax
      - the matching case to expression will only be executed
      - if there is no matching case, default block will be executed

```
switch( expression ) {
    case value _1: // if the expr. matches value_1
        statement(s);
        break;
    case value_2: // if the expr. matches value_2
        statement(s);
        break;
        ...
    case value_m:
        statement(s); // if the expr. matches value_m
        break;
    default: // none of the above matches
        statement(s);
}
```

```
let fruits="Apple";
switch(fruits) {
    case "Banana":
        price = 200; break;
    case "Apple":
        price = 300; break;
    case "Cherry":
        price = 400; break;
    default:
        document.write("Could not find any.");
        price = 0;
}

// result : price=300
```

JavaScript (JS)

- Conditional statement
  - switch-case
    - case values can only be either constant variables or literal

```
const MAX = 100;
...
...
case 1 :
case 2.7 :
case "Seoul" :
case MAX : // const variables are okay
case true :
case 2+3 : // simple expression with literals are fine
```

Accepted/valid case values

```
case a :      // Error (variable cannot be used for case value
case a > 3 :  // Error (expression with variables cannot be
used for case value)
```

mis-use of case values

JavaScript (JS)

- Conditional statement
  - switch-case
    - break statement determines where to stop and get out of the entire switch-case block

```
<script>
var guess = 3;
switch( guess ){
    case 1:
        document.write("1<br>");
    case 2:
        document.write("2<br>");
    case 3:
        document.write("3<br>");
    case 4:
        document.write("4<br>");
    case 5:
        document.write("5<br>");
    default:
        document.write("None<br>");
}
</script>
```

without break, all the remaining code will be executed

3
4
5
None

```
<script>
var guess = 3;
switch( guess ){
    case 1:
        document.write("1<br>");
        break;
    case 2:
        document.write("2<br>");
        break;
    case 3:
        document.write("3<br>");
        break;
    case 4:
        document.write("4<br>");
        break;
    case 5:
        document.write("5<br>");
        break;
    default:
        document.write("None<br>");
        break;
}
</script>
```

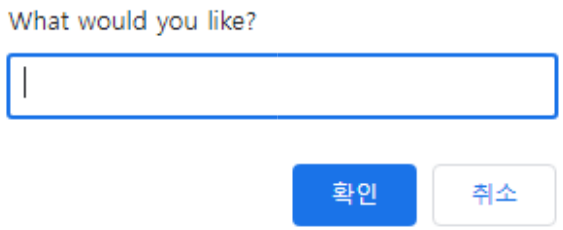
"break" tells when to stop

3

JavaScript (JS)

- Conditional statement
  - switch-case : example - ordering a coffee or tea

```
10  <script>
11      let price = 0; // initially set to zero
12      let coffee = prompt("What would you like?","");
13      switch(coffee) {
14          case "americano" :
15          case "iced americano" :
16              price = 2000; // for both hot and iced americano
17              break;
18          case "latte" :
19          case "caffe latte" :
20              price = 3000; // for both latte and caffe latte
21              break;
22          case "tea" :
23              price = 3500;
24              break;
25          default :
26              document.write("We do not have " + coffee + " for now.");
27      }
28      if(price != 0)
29          document.write(coffee + " is " + price + " won.");
30  </script>
```



JavaScript (JS)

- Loops
  - used to execute a block of code a number of times
  - for a sequence of repetitive task, loops can greatly reduce the coding burden
  - there are three types of loops in JS : for, while, do-while

  - **for** loop

```
for( initialization_stmt ; condition_stmt ; update_stmt ){
    statement(s);
}
```

    - `init_stmt` is executed only once at first
    - `condition_stmt` is evaluated on each iteration, determining when to stop
    - `update_stmt` is executed at the end of each iteration
    - when the `condition_stmt` is true, statement(s) will be executed
    - the execution order is: 1 > 2(true) > 3 > 4 > 2(true) > 3 > 4 > 2(false) > loop ends
    - example:

```
<script>
for( let num = 1; num <= 10 ; num++ ) {
    document.write(num + ",");
} // prints : 1,2,3,4,5,6,7,8,9,10,
</script>
```

JavaScript (JS)

- Loops
  - `while` loop

```
while( condition_stmt ){
    statement(s);
}
```

1

2

    - `condition_stmt` is evaluated on each iteration, determining when to stop
    - when the `condition_stmt` is true, statement(s) will be executed
    - the execution order is: 1(true) > 2 > 1(true) > 2 > ... > 1(false) > loop ends
    - example:

```
<script>
let num = 1;
while ( num <= 10 ) {
    document.write(num + ",");
    num++;
} // prints : 1,2,3,4,5,6,7,8,9,10,
</script>
```

JavaScript (JS)

- Loops
  - `do-while` loop

```
do{
    statement(s);
} while ( condition_stmt );
```

1 2

    - `condition_stmt` is evaluated on each iteration, determining when to stop
    - when the `condition_stmt` is true, statement(s) will be executed
    - the execution order is: 1 > 2(true) > 1 > 2(true) > ... > 2(false) > loop ends
    - example:

```
<script>
let num = 1;
do { // prints : 1,2,3,4,5,6,7,8,9,10,
    document.write(num + ",");
    num++;
} while ( num <= 10 );
</script>
```

JavaScript (JS)

- Loops
  - for loop example : changing font-size

```
10 <script>
11     for(let size=10; size<=35; size+=5) { // incr by 5
12         document.write("<span ");
13         document.write("style='font-size:" + size + "px'>");
14         document.write(size + "px");
15         document.write("</span>");
16     }
17 </script>
```

Changing fontsize : for-loop

10px15px20px25px30px35px

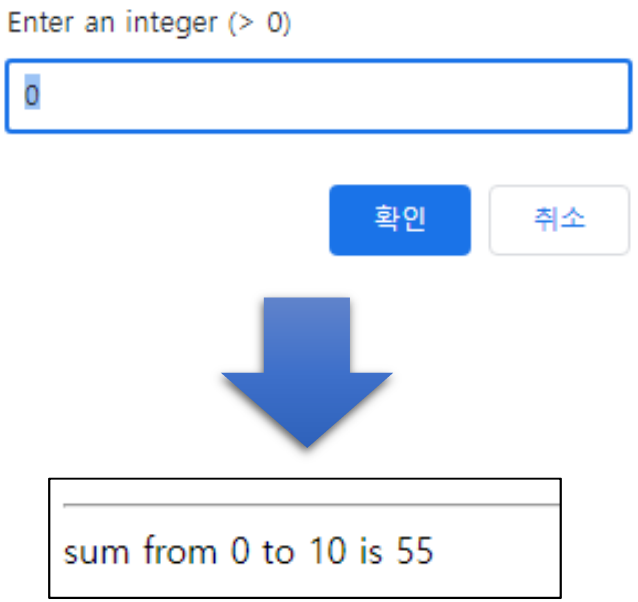
JavaScript (JS)

- Loops
  - while loop example : summing up from 0 to n

```
10  <script>
11      let n = prompt("Enter an integer (> 0)", 0);
12      n = parseInt(n); // change : string > int
13      let i=0, sum=0;
14      while(i<=n) { // from i=0 to i=n
15          sum += i;
16          i++;
17      }
18      document.write("sum from 0 to " + n + " is " + sum);
19  </script>
```

  - do-while loop example : summing up from 0 to n

```
10  <script>
11      let n = prompt("Enter an integer (> 0)", 0);
12      n = parseInt(n); // change : string > int
13
14      let i=0, sum=0;
15      do {
16          sum += i;
17          i++;
18      } while(i<=n); // from i=0 to i=n
19      document.write("sum from 0 to " + n + " is " + sum);
20  </script>
```



JavaScript (JS)

- Loops
  - break : breaks/stops the current loop (for, while, do-while)

```
for( … ) {
   ................
   break;
   ................
}
................
................
................
```

```
for( … ) {
   while( … ) {
      ................
      break;
      ................
   }
   ................
}

}
```

breaks the current loop

  - continue : stops the current iteration, and keep iterating

```
for(init; condition; update) {
   .........
   continue;
   .........
}
```

```
while(condition) {
   .........
   continue;
   .........
}
```

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

JavaScript (JS)

- Functions
  - Set of statements such that when the function is called, the statements within the function are executed
  - Syntax

```
function functionName(arg1, arg2,..., argN) {
    ...statements...;
    return statement;
}
```

beginning of function declaration

function name

arguments

```
function adder ( a,  b ) {
  let sum;
  sum = a + b;
  return sum;
}
```

return keyword

value to be returned

  - Calling a function

```
let n = adder(10, 20);
```

when called

```
function adder(a,  b) {
  let sum;
  sum = a + b;
  return sum;
}
```

❶ ❷ ❸

30

JavaScript (JS)

- Functions
  - example

```
6   <script>
7   function adder(a, b) { // Declaring a function
8       let sum;
9       sum = a + b;
10      return sum;
11  }
12  </script>
13  </head>
14  <body>
15  <h3>Function adder()</h3>
16  <hr>
17  <script>
18      let n = adder(24567, 98374); // calling a function
19      document.write("24567 + 98374 = " + n + "<br>");
20  </script>
21  </body>
```

Function adder()

24567 + 98374 = 122941

JavaScript (JS)

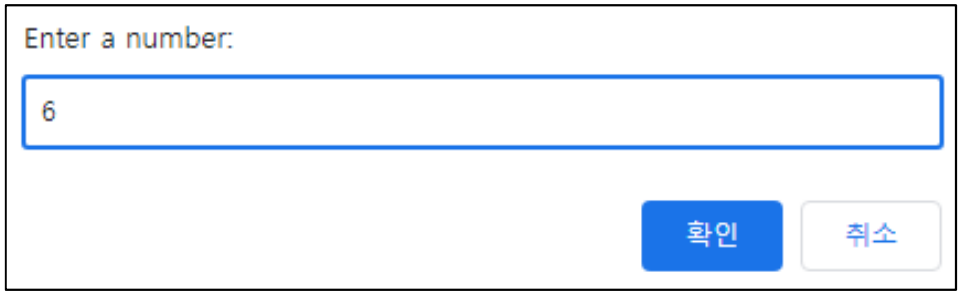
- Functions
  - Some useful functions

Function	Explain
eval(exp)	Evaluate the equation 'exp' and return the retuls ex) let res = eval("2*3+4*6"); // res = 30
parseInt(str)	convert the string 'str' to the corresponding integer
parseFloat(str)	convert the string 'str' to the corresponding real number

JavaScript (JS)

- Functions
  - Example

```
6   <script>
7   function gugudan(n) { // Declaring a function
8       let m = parseInt(n); // convert: string to integer
9       if(isNaN(m) || m < 1 || m > 9) {
10          alert("Incorrect input");
11          return;
12      }
13      for(let i=1; i<=9; i++) { // i iterates 1~9
14          document.write(m + "x" + i + "=" + m*i + "<br>");
15      }
16  }
17  </script>
18  </head>
19  <body>
20  <h3>Multiplication table</h3>
21  <hr>
22  <script>
23      let n = prompt("Enter a number: ", ""); // n : string
24      gugudan(n); // calling a function
25  </script>
26  </body>
```



Multiplication table

6x1=6
6x2=12
6x3=18
6x4=24
6x5=30
6x6=36
6x7=42
6x8=48
6x9=54

THE END