Scripting language I (LS1)

Course 3

Conditions

Condition brings the possibility for a program to offer alternatives. They somehow consist into tests able to verify if a criteria has been met or not. Depending on the result of the test, it is possible to program adapted instructions.

Basically, a condition allows to say : if the criteria is not met, do this; if it is met, do that. The answer to a condition that is returned can only be yes or no, or actually, in programming : ***true*** or ***false***. These returned values are called booleans.

Booleans

Booleans offer an alternative allowing to verify if a condition is *true* or *false*.

Conditions are testing values and they are of two different types :

- Logical value
- Comparison value

Comparison operators

Comparison operators allow to compare values.

Operators	Signification
==	Equals to
!=	Differs from
===	Content and type equal to
!==	Content and type different from
>	Higher than
>=	higher or equal to
<	Lower than
<=	Lower or equals to

```
<script>
        var number1 = 2, number2 = 2, number3 = 4, result;
        result = number1 == number2;

        alert(result);
</script>
```

Explanation:

In the preceding example, variables *number1* and *number2* are equals. The returned Boolean is therefor *true*. The condition is *verified*.

```
<script>
        var number1 = 2, number2 = 2, number3 = 4, result;
        result = number1 == number3;

        alert(result);
</script>
```

Explanation:

In the preceding example, variables *number1* and *number3* are not equals. The returned Boolean is therefor *false*. The condition is not *verified*.

```
<script>
        var number1 = 2, number2 = 2, number3 = 4, result;
        result = number1 < number3;

        alert(result);
</script>
```

Explanation:

In the preceding example, variable *number1*'s value is lower than *number3's value*. The returned Boolean is therefor *true*.

Logical operators

There are three logical operators :

Operators	Logic type	Usage
&&	AND	value1 && value2
||	OR	value1 || value2
!	NO	!value

&& (AND)

This operator verifies the condition when all values returned are *true*. If one of the values returns *false*, the condition is not verified.

```
<script>
        let result = true && true;
        alert(result);                          // Displays : « true »
</script>
```

|| (OR)

This operator verifies the condition if one of the returned value is *true*.

```
<script>
        result = true || false;
        alert(result);                  // Displays : « true »
</script>


<script>
        result = false || false;
        alert(result);                  // Displays : « false »
</script>
```

! (NO)

Applying to only one value at the time, this operator inverts the returned value; *true* then becomes *false* and conversely.

```
<script>
        let result = false;
        result = !result;           // Inverts « result »

        alert(result);              // Displays : « true »
</script>
```

Combining operators

Comparison operators all accept two values and return one Boolean. Logical operators accept several Booleans and return one. It is then possible to couple the comparison operators' returned values with those of logical operators.

Shorten the code :

```
<script>
        let condition1, condition2, result;
        condition1 = 2 > 8 ;                // false
        condition2 = 8 > 2 ;                // true

        result =  condition1 &&  condition2;

        alert(result);                      // Displays : « false »
</script>
```

```
<script>
        let result = 2 > 8 && 8 > 2;

        alert(result);                      // Displays : « false »
</script>
```

The « if » conditional structure

The condition structure of ***if*** allows instructions' execution if one or more conditions are verified.

A condition if formed of three components :

- The conditional structure.
- The parenthesis, containing the condition(s) to evaluate (Booleans returned by the conditional operators).
- The brace brackets, containing the instructions to execute if the condition is verified.

```
if (condition verified) {
        instructions to be executed;
}
```

Note: Instructions will be executed only if the condition is verified (true Boolean returned). An unverified will keep the instructions to be executed.

```
if ((condition1)&&(condition2)) {
        instructions to be executed;
}
```

Note: If both conditions are verified, the instructions will be executed.

```
if ((condition1)||(condition2)) {
        instructions to be executed;
}
```

Note: If one of the two conditions is verified, the instructions will be executed.

If there is only one instruction, the brace brackets are optional :

```
if (x==2) document.write("X equals 2");
```

Exercise 4

Using « if » conditional structure, create a program that will display a message adapted to the user input data based on its age range :

Age range	**Comment example**
1 to 17	You're not major yet.
18 to 49	You are major, but not senior yet.
50 to 64 ans	You are senior, but not retired yet.

Program's flow :

- User loads the pages
- User is asked to supply its age
- The proper comment is displayed

The confirm() function

This function returns a Bolean (*true* or *false*) based on confirmation or refusal of the user.

```
<script>
if (confirm('Do you want to test this program?')) {
        alert('Instruction has been correctly executed!');
}
</script>
```

The « else » conditional structure

This structure is the equivalent of saying « otherwise » or « if not ». It allows to execute specific instructions if the prior condition isn't verified. It is more simple than using two consecutive « if » structures :

```
<script>
if (confirm(' Click OK if you're 18 or over')) {
        alert('Wlecome to our site.');
} else {
        alert("Sorry, come back when you'll be 18.");
}
</script>
```

To avoid any confusion, use the following layout :

```
<script>
if ( condition ) {
        instruction;
} else {
        instruction;
}
</script>
```

The « else if » conditional structure

« *else if* » makes it possible to provide a sequence of specific conditions. If a condition is not verified, *else if* will proceed with another instruction.

Syntax :

First condition to evaluate
Second condition to evaluate if the first one is not verified
If none are verified, *else* is executed

```
<script>
let floor = parseInt(prompt("Enter desired level (from -2 to 30) :"));
if (floor == 0) {
            alert("You are already at ground level.");
} else if  (-2 <=  floor &&  floor <= 30 ) {
            alert("Direction level " +  floor + " !" );
} else {
            alert("This level doesn't exist." );
}
</script>
```

The « switch » condition structure

Although very useful, *if else*, is not as efficient to work with independent elements of data. The *switch* structure may then be used. When *else if* is returning a Boolean (*true* or *false*), *switch* returns a predefined value or a default instruction, if necessary.

Example :

Let's create a price list for grocery shopping (fruits). User asks for a specific product and the program displays the price. If the product is not available, a default message is returned :

```
<script>
let fruits = prompt('What fruit do you want to buy? :');
switch (fruits) {
        case "Oranges":
        alert("Oranges : 0.59 $ le kilo.");
        break;

        default:
        alert("Sorry, we don't have " + fruits + ".");
}
</script>
```

Explanations :

The first code line creates a variable used to store the fruit the user wants to buy.

The second code line starts *switch*, in which each available item is defined starting with *case*, followed by the instruction to execute, and ending with *break;*.

To add a new available item, simply repeat the preceding sequence (*case*, instruction and *break;*).

Finally, using *default:* defines the instruction to be executed if a desired item is not available.

The ternary conditional structure

A ternary (three elements condition) is a short and fast way to code a condition equivalent to *else if*.

A simple syntax:

condition ? ifTrue : ifFalse

If the condition is *true*, the first expression is returned. If the condition is-*false*), the second expression is returned.

Example :

Here is a way to display the price of an item for a member and a non-member of a cooperative store :

"The price is : " + (member ? "25 $" : "50 $")

In the example, if member variable is *true*, the price displayed will be 25 $, if the condition is *false*, it will be 50 $.

Example :

Let's analyze a program that says «You are : 18+!» if the users confirms being major, and « You are : -18 » if the users does not confirm being major coded using *if* and *else*. Then, we'll check how it's done with ternaries.

Example using « if / else »

```
<script>
let         startMessage = "You are : ",
            endMessage,
            adult = confirm("Are you major of age ?");

if (adult) {
            endMessage = "18+";
} else {
             endMessage = "-18";
}
            alert(startMessage + endMessage);
</script>
```

Example using a ternary

```
<script>
let         startMessage = "You are : '"
            endMessage,
            adult = confirm("Are you major of age ?");
            endMessage = adult ? "18+" : "-18";

            alert(startMessage + endMessage);
</script>
```

Assignment 4

Create a program showing a specific message based on the age range supplied by the user :

Age ranges	**Comment examples**
1 to 17	You're not major yet.
18 to 49	You are major, but not senior yet.
50 to 64	You are senior, but not yet retired.

Program flow :

- User loads the web page
- The user is asked to enter his age
- A specific comment based on his age is displayed