Fundamental notions of programming (NFP)

Class 5

Evaluating conditions

As we already have seen, control structures such as IF or loops are based on conditions evaluating at least one variable in parenthesis. There are different types of evaluations that can be made using relational (comparison) operators.

Relational operators

Operator	Description
==	**Equal:** Checks if the values of two operands are equal and returns a boolean (true/false).
===	**Equal in value and type:** Checks if the values of two operands are of equal value and type (string or number) and returns a boolean.
!=	**Not equal:** Checks if the values of two operands are not equal and returns a boolean.
>	**Greater than:** Checks if the value of the left operand is superior to the value of the right operand and returns a boolean.
<	**Less than:** Checks if the value of the left operand is inferior to the value of the right operand and returns a boolean.
>=	**Greater than or equal:** Checks if the value of the left operand is superior or equal to the value of the right operand and returns a boolean.
<=	**Less than or equal:** Checks if the value of the left operand is inferior or equal to the value of the right operand and returns a boolean.

10 == 20
Does 10 equals 20? // False

10 != 20
Is 10 not equal to 20? // True

10 > 20
Is 10 greater than 20? // False

10 <= 20
Is 10 less or equal to 20? // True

10 === Hello
Does 10 equals Hello in value and type? // False

Logical operators

Logical operators are typically used with Boolean (logical) values and its result are return whether the condition is verified or not, true or false.

Operator	Description
!	**Not:** Used to exclude a value.
&&	**And:** Used to verify two or more conditions.
||	**Or:** Used to verify one of several conditions.

! (Not)

```
let number = 10

number != 10          // False: 10 equals 10
number != 20          // True: 10 does not equal 20
```

&& (And)

```
let x = 7;
let y = 4;

(x < 10 && y > 1);     // True
(x < 10 && y < 1);     // False: y is not less than 1
```

|| (Or)

```
let x = 7;
let y = 4;

(x == 5 || y == 5);    // False: x and y does not equal 5
(x == 7 || y == 0);    // True
(x == 0 || y == 4);    // True
(x == 7 || y == 4);    // True
```

The loops

In programming, loops are used to make code shorter by using some sorts of automatic repetitions. These repetitions (or iterations) continue as long as a condition is verified, as long as it returns a *true* boolean. The loop structures contain instructions which will be repeated as long as the condition is verified as true.

while

While is the most common and simple loop in JavaScript. It basically says: as long as the condition is true, repeat the block of instructions.

Syntax :

```
while (condition) {
        Instructions...
}
```

For instance, it is possible to create an instruction that will repeat a certain number of times using ***i++*** each time the loop is repeated.

```
<script>
let i = 1;                              // Defines iteration #1
while(i<=5) {                           // As long as 5 or under
        document.write("Iteration number: " + i + "<br />");
        ++i;                            // Adds 1 to the count
}
</script>
```

Result :

```
Iteration number: 1
Iteration number: 2
Iteration number: 3
Iteration number: 4
Iteration number: 5
```

for

for is used more often as it is shorter and includes the start value of i, the condition and the iteration's incrementation all at once.

```
<script>
for(i=1; i<=5; i++) {
        document.write("Iteration number: " + i + "<br />");
}
</script>
```

Assignment 5 : loops

Create pseudocode and flowcharts first, and then create the following programs using a for loop :

- A secret number variable between 1 and 100 is set.
- The program requests the user to choose a number between 1 and 100.
- If the chosen number if higher or lower than the secret number,, it is told to the user that tries again until the secret number is found.
- If/when the chosen number is the same as the secret number, a congratulation message is output to screen such as the following :

> Congratulations!
> *x* was the secret number.
> You found it in *y* attempts