college CD

Scripting language I (LS1)

Course 8

Objects and array

The object notion is quite an abstract one, but quite easy to understand. As we have already said, JavaScript possesses native objects we can directly use. We already did, actually : strings, numbers, booleans...

Example :
let myString = "This is a string";

Explanation :
The variable *myString* counts an object : the string. Variables always count the same thing : an object that can be of different types. .

Objects components :

Objects are formed with three components : a *constructor*, *properties* and *methods*.

The constructor
A constructor is a code used to create an object. This is done automatically when using native objects. To some extent, it can be viewed as a normal function which declares a value.

Properties
A property consists in data value contained in a variable.

Methods
Methods allows to modify objects. They are functions contained in objects which makes it possible to apply different operations on the object's content. For instance, there's a method which makes it possible to transform a string in uppercase.

Explanation :

An object is firstly created (declaration of a string). Then, using the method ***lenght***, the number of character is displayed in a message window. With the method *toUpperCase()*, it would display the string in uppercase.

Example :

```
let myString = "This is a string";
        alert(myString.length);
```

Note : A dot is used between the variable and the method

Array (matrix)

Arrays are variables containing several values. It is possible to access these values using index numbers (starting at zero).

Index

Since items numbering starts with zero, there is always a shift to take in consideration (first item is index 0, second item is index 2, etc.).

Declaring an array

let (or *var*) is used to declare an array, just like for any variables. But the syntax used to declare the several values (the items) is a specific one. The content of an array is placed between straight brackets and a colon is used between the indexes (using quote marks since they are strings) :

Example :

```
let myAnimals = ["dog", "cat", "bird", "fish"];
```

Explanation :

In the preceding example, the array (myAnimals) contains 4 indexes (items numbered 0 to 3) associated to four different animals (values).

Note : There is also a long version of array declaration that you will see here and there :

```
let myAnimals = new Array("dog", "cat", "bird", "fish");
```

Adding items the an array

push() :
This method adds data values at the end of an array.

unshift() :

This method adds data values at the beginning of an array.

```
let myAnimals = ["dog", "cat"];
myAnimals.push("bird", "fish");
alert(myAnimals);
```

Note : The new items will follow the existing ones in the array.

Delete items from an array

***shift()*:**
This method deletes the first data value of an array.

***pop()*:**
This method deletes the last data value of an array.

```
let myAnimals = ["dog", "cat", "bird", "fish"];
myAnimals.shift();                    // Would delete : « dog »
myAnimals.pop();                      // Would delete : « fish »
alert(myAnimals);
```

Converting a string in an array

The method ***split()*** makes it possible to convert a string into an array's index based on blank space.

```
let     myAnimals = "dog cat bird fish",
        myArray = myAnimals.split(" ");
alert(myAnimals);
alert(myArray);
```

Nota : A string may be cut into array's items using ***toString()***. It is also possible to do the opposite (to make a string out of an array's items) using ***.join()***.

Example :

```
let     myAnimals = "dog cat bird fish",
        myArray  = myAnimals.split(" ");
alert(myAnimals);
alert(myArray);
var myAnimals_2 = myArray.join('-');
alert(myAnimals_2);
```

Retrieving data values from an array

Retrieving values from an array makes it possible, for instance, to display its items, apply modifications to them or execute instructions based on its content.

Retrieving values :

```
let myAnimals = ["dog", "cat", "bird", "fish"];
alert(myAnimals[1]);                                    // Displays : « cat »
```

Retrieving items of an array using a « for » loop

The principle is quite simple. A loop must be created and repeated as many times as it contains items. In order to know how many items an array contains, the *.length* method is used. the i++ iteration variable (i) will keep the iterations up to date.

Example :

```
for (let i = 0; i < myArray.length; i++) {
        alert(myArray[i]);
}
```

Another more performing way:

```
for (let i = 0, c = myArray.length; i < c; i++) {
        alert(myArray[i]);
}
```

Literal objects

As we just have seen, it is possible to access an array's items using index numbers. It is also possible to access items using a ***name***.

Exemple :

```
let family = {
        me: "John",
        sister:"Christine",
        brother: "Martin",
        cousin_1: "Ann",
        cousin_2: "Robert"
};
```

In the preceding example, we created an array listing family members. The item named « sister » would return «Christine ».

In the following examples, the declaration will create something similar to an array, but each items can be returned using its name which are, in facts, properties.

Accessing items

```
family.sister;
OR
family["sister"];
OR
let member ="sister";
alert(family[member]);          // Would display : « Christine »
```

Note: objects' syntax

Array : `let myArray = [ ];`

Objects : `let myObject = { };`

Define objects' item :

```
let myObject = {
        item_1: "string 1",
        item_2: "string 2",
        item_3: "string 3"
};
```

Adding items to literal objects

All there is to do, to add literal objects items, is to specify a non-existing name and a value :

```
family["uncle"] = "Bob";
```

OR

```
family.uncle = "Bob"
```

Retrieving objects content using « *for in* » loop

The *for in* loop is quite similar to arrays in the way it works. Although, the object you want to retrieve needs to be stored in an ID (variable) and the object's name to be specified.

The loop will repeat for as long as there are items left. It will then stop and execute the following instructions.

```
let result = "";
let family = {
        me: "John",
        sister:"Christine",
        brother: "Martin",
        cousin_1: "Ann",
        cousin_2: "Robert"
};

for (let id in family) {
        result = result + family[id];
}

alert(result);        // Result would be the family members list
```

Assignment 6

Create a program that will ask the user to enter as many first names a wanted. When the user is done supplying name (clicking the button with an empty field), a message window displays the list of names.

Program's flow :

- User is asked to supply as many names as wanted.
- When a name is supplied, user is asked to supply another name.
- When done, user submits an empty input field.
- A message window displays the name list of names (using blank spaces in between, not colons).