Scripting language I (LS1)

Course 1

Developer's role (programmer)

One of the programmer's many tasks consists into coding series of instructions in order for the program to achieve its different purposes correctly.

At first, a beginner will be able to code simple programs quite rapidly. However, this task will become more difficult as programs become more complexed. Experience, and a lot of practice, will be necessary to master programing logic and the specificities of the various languages you will learn.

Programs execution

In programing, we call *instruction* the request that is made to a computer to execute certain tasks. Whatever the language used, the program will need to be translated into *Assembly* (machine language) in order to be executed.

There are different types of programming languages :

Interpreted languages :
With certain programming languages, each line of *source code* are translated in *Assembly*. They need to be executed one after another by a specific program (software) named *interpreter*. These languages are called interpreted languages (e.g. JavaScript, Python or PHP).

Compiled languages :
Other languages allows to create an executable program (standalone, e.g. .exe files under Windows system). To do so, the source code needs to be compiled using a specific intermediary program called *compiler*. These are called compiled languages (e.g. C or C++).

Pre-compiled languages :
Finally, some languages, which source code will be treated with a *pre-compiler*, will generate a group of files that may be executed on all platform (e.g. Java, VB.NET, C#, etc.).

Programming logic

Except in some very rare occasions and for very simple tasks, we never start coding without a plan. This isn't a grocery list we're making! The problem must be very well analyzed and structured in order to know in advance the entire sequence of instructions needed.

Here's a simple example of pseudo-code (from *Alain Tarlowski*) showing different instructions for a program we'll call : Let's make pastas! :

```
START
    Get a cooking pot
    Add water in the pot
    Add salt in water
    Put pot on the stove
    Turn on heat to Max
    As long as water isn't boiling
        Wait
    Get pastas from the larder
    Put pastas in the water
    As long as the pastas aren't done
        Wait
    Put pastas in the strainer
    Strain pastas
    Put pastas in a bowl
    Taste pastas
    If pastas taste dull
        Add salt
    Taste pastas
    If we prefer butter instead of oil
        Add butter
    If not
        Add oil
END
```

In the preceding pseudo-code, there are three types of instructions :

- Simple instructions (e.g. get a cooking pot)
- Conditional instruction (e.g. If we prefer butter...)
- Loop instructions (e.g. As long as pastas aren't done...)

So, the complex problem of cooking pastas has been broken down to a sequence of simple instructions. That is what is called an *algorithm*. It is the very basics of programming, whatever the language used.

JavaScript

What is JavaScript

JavaScript is a web programming language invented in 1995 by *Brendan Eich* for Netscape (ancestor of Firefox browser). It is mainly used on the INternet along with web pages (HTML and XHTML) to create more dynamic pages. It is allows, for instance, to show or hide content, to create slide shows and add user interactions.

With a growing number of platforms and devices, JavaScript became more and more popular as its scripts can be executed on all of these platforms and devices. With JavaScript, it is now possible to create very fast applications and even to interact with databases.

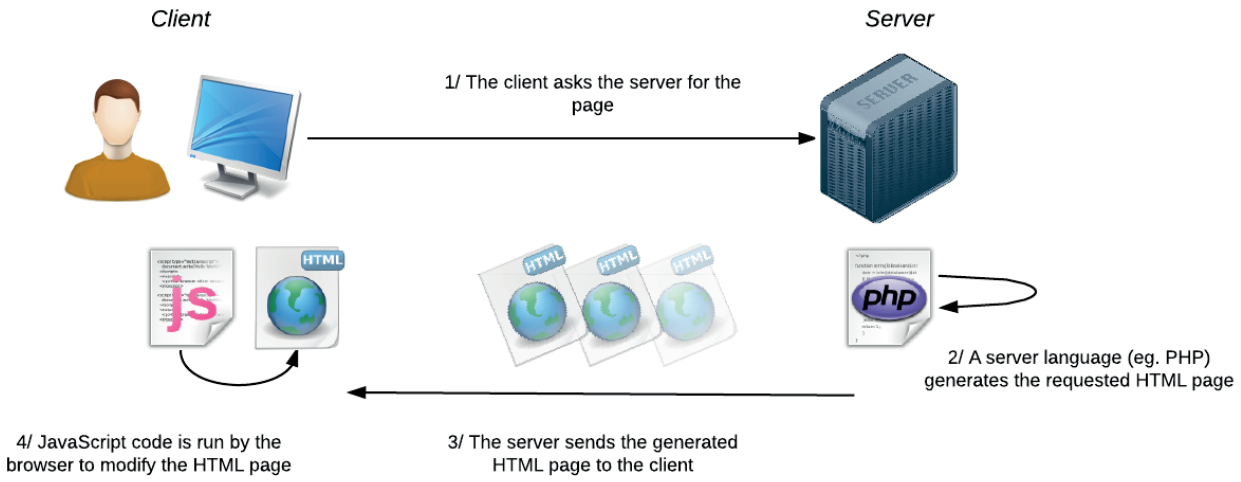
JavaScript specificities

Javascript is an object-oriented script language which programs are executed on client-side.

This means that the language creates scripts (programs) which instructions are associated to web pages. It doens't need to be compiled as it is interpreted by the web browsers using different internal interpreters (*JScript* on Explorer, *SpiderMonkey* on Mozilla, and *V8* on Chrome).

JavaScript contains various objects and specific usage modes. But it is also possible to create customized objects whenever needed.

When it is said that JavaScript is a client-side interpreted object, it is because the execution of its instructions are made on the user's computer and not on the server-side like other languages (e.g. PHP).



JavaScript program structure

It was already said that a computer program is made of a sequence of instructions which are executed by a computer. These instructions may be coded in text format within a HTML document, or within one or several external files. This is wath is called **source code**.

Each source code's text line is called **code line**. Empty code lines may be used as it is ignored when executed.

A program's command is called instruction, and it should be followed by a semicolon. Instructions can be nested into a HTML tag or into the HTML document's <head> or <body> sections using the following tags: <script> </script>.

Instructions can also be written into one or more external files. In this last case, the HTML document must be linked to the external file(s).

Upon execution, instructions are executed one after another in their programming order, each of them producing results which combination produces the final result the program was created for.

```
<script>
        instruction 1;
        instruction 2;
        instruction 2;
</script>
```

Semicolon is not mandatory if each of the instructions are written on different lines. Although, semicolon are mandatory if a code line contains more than one instruction. The systematic use of semicolons is highly recommended.

Example:

```
instruction 1; instruction 2;
```

or

```
instruction 1
instruction 2
```

Indent and layout

If you take a look at any program's source code, you will notice programmers use tabs in order to create a hierarchy. in the program's layout. Even though this isn't mandatory and that it has no effect on the program itself, make sure to develop this essential good habit right from the beginning. Indents and line changes make the source code easier to read and to understand. Other programmer will judge you very negatively if you don't organize your code or if you are not commenting it adequately.

Comments

All code lines are considered as instructions to be executed. It is although possible to transform a code line into a comment line. These lines are ignored by the interpreter upon execution of the script. Single line comments must be preceded by two forward slashes (**//**), and multi-lines comments must start with **/*** and end with ***/**.

```
<script>
        instruction 1;      // Single line comment
        instruction 2;
        instruction 2;
        /* Single or multi-lines comment */
</script>
```

Script compression

In order to create lighter files for very big scripts, some scripts are sometimes run into softwares which task is to compress the file. To do so, spaces and changes of lines are deleted creating a compact block of characters.

This is one good reason to always use semicolons. In such a situation, only one missing semicolon and your program won't work.

Example of compressed script :

```
function displayClassElement(a,d){var c;nodes=new
Array();if(typeof(a)=="string"){c=document.getElementById(a)}
else{c=a}if(c.cells!=undefined){nodes=c.cells}else{if(c.rows!=undefined)
{nodes=c.rows}else{if(c.childNodes!=undefined){nodes=c.childNodes}}}
for(var b=0;b<nodes.length;b++){if(nodes[b].tagName!=undefined)
{displayClassElement(nodes[b],d);if(nodes[b].className==d){nodes[b].
style.display=getDisplayStyle(nodes[b])}}}};
```

Your first scripts

Work environment

It doesn't take much to a programmer to create its magic : a text-only editor and a web browser. You will eventually be using adapted coding applications that will make your programmer's life easier, but let concentrate on coding for now without having to learn to operate a new software.

You will first write your instructions in an HTML document which will be viewed in a web browser. We recommend Chrome or Firefox.

First steps

First, create a HTML5 document and save it. Then, test it in the web browser to make sure all is fine.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>My first JavaScript</title>
</head>
<body>

</body>
</html>
```

JavaScript instructions mult be placed within **<script></script>** in the **<head>** or **<body>** sections of the HTML document. We'll put within the **<body>** section for the time being.

Just like it is done for CSS, JavaScript instructions can be written in an external file. In this case, the JavaScript file (**.js** extension) will have to be linked to the HTML document using **<script> </script>** tags within the **<head>** section. :

```
<script src="myscript.js"></script>
```

Warning : JavaScript is case sensitive, which means the lowercase and uppercase of the same letter are considered two different characters.

For instance, the words « Hello » and « hello » will be considered as two different words.

Exercise 1 : Your first script

Code the following program and test it in the web browser.

The *alert* function (or *window.alert*) is used to create a pop-up window showing the parenthesis' content (in this case : text between quote marks).

Some programmers find it safer to use primes for text and quote marks for numbers. It really doesn't matter that much, but keep it to one way within a program or things could get quite confusing.

```
<script>
        alert("Hello world!");
</script>
```

Note : *It is a tradition since the very beginning of programming that the first program someone codes should say « Hello world ! ». Let's do it!*

Note : <script></script> tags are enough when coding in HTML5. But, when coding in HTML4 or XHTML, an attribute and a value are mandatory :

```
<script type="text/javascript">
<!--
        instruction;
//-->
</script>
```

Exercise 2 : multiple instructions

As earlier said, instructions are read one line after the other, each of them producing a result until the end of the program.

In this exercise, you are asked to code a sequence of instructions which will result in popping four consecutive windows.

```
<script>
        alert("Hello!");
        alert("This program contains several instructions");
        alert("Not to difficult, so far?");
        alert("Come back and visit us soon!");
</script>
```

Events

JavaScript functions, such as *alert(),* may be associated to events which will define when the function will be executed. Here are a few commonly used functions :

onload

Executes a function when a file has loaded.

<body onload="alert('Hello world')">

onclick

Executes a function when an element is clicked on.

My link

onmouseover

Executes a function when an element is rolled-over by the mouse pointer.

My link

onmouseout

Executes a function when an element is rolled-out by the mouse pointer.

My link

Functions

Functions are used to produced a specific action. Some are predefined in browsers (natives). Here are those you will be using for now :

alert()
Creates a message window.

prompt()
Creates a message widow followed by a text input field.

confirm()
Creates a message window along with buttons allowing the user whether to accept or refuse.

parseint()
Convert a character string into numerical value.

Types and values

A value is information, data, used in a program. Values can be of different *types*. The value's type determine its role and the operations that may be applied to it. Let's see two types : string and number.

String

A string consist into text or content made of various characters (letters, numbers, symbols) intended to be displayed. It is precisely what we have been using in the last exercises.

Strings are placed between quotes or primes, but again, stick to one way of doing things to avoid confusion.

Preceding them with a backslash, it is also possible to use special characters such as the followings :

Special characters :

Characters	**Results**
\n	Changing line
\' et \"	Prime / Quotes
\\	Backslash
\uXXXX	Unicode characters

String type doesn't allow mathematical operations. But the « + » operator allows joining to strings. That's called **concatenation**.

```
<script>
        alert("Hello!\nThis allows me\nto change line.");
</script>
```

```
<script>
        alert("Hel"+"lo");
</script>
```

Number

A number type value represents a numerical value, a quatity, a number allowing mathematical operations. They can be whole numbers (e.g. 3, 7, 12) or decimals (e.g. 3,1416).

All usual operators can be used (JavaScript also offers predefined formulas):

Operators	Results
+	Addition
-	Subtraction
*	Multiplication
/	Division

Mixing value types

Of course, it is possible to use different types of value within a program. For instance, a string and a number. In the following example, the string (between quotation marks) will be displayed followed by the result of the addition (between parenthesis), a number type value.

```
<script>
        alert(10+2);
</script>
```

```
<script>
        alert("10 + 2 = " + (10+2));
</script>
```

Note: Even though parenthesis aren't always mandatory for mathematical operations, it is here needed because the operator «+» is used to join the two values (the string and the addition's result).

Then, parenthesis aren't needed for subtraction, multiplication and division unless it is par of the mathematical operation or in a formula.

Assignment 1 :

In a HTML5 document, code the followings :

- Using an *onload* event, trigger a pop-up window showing the message « This was triggered by an event! » when the document has been loaded.
- Create a button and, using an *onclick* event, trigger a pop-up window showing the message « Wow! This works! ».
- Insert an image and then use an *onmouseover* event to trigger a pop-up window showing the message « This is a rollover ».
- Create a link to any web site and use an *onunload* event so that before leaving the page, a pop-up window pops up showing the message «Visit us again soon! ».