**Lab: Javascript in the browser**

The purpose of this lab is to study the concepts discussed during [INF203 on Web technologies](http://perso.enst.fr/dufourd/cours/inf203/) and in particular JavaScript. The objective of this lab is to program in JavaScript in the browser.

This session is automatically graded and you can test the grading. See at the end of this document. It is important to respect strictly the file names mentioned here.

Unlike previous labs, this one will not work if you stay “local”, i.e. if you leave all your files in a folder and you access them directly by “Open File” in the browser. You need to install a web server and use a folder in its server root, for example.

Telecom Paris offers a web container accessible via the school server. The documentation for this service is [there](https://eole.telecom-paris.fr/vos-services/services-numeriques/hebergement-web-persoprojetblog/guide-dutilisation-dun-container). To activate this personal pages service, you have to go to <Http://moncompte.telecom-paris.fr>.

**Assignment**

For this lab, you will need to upload the different JavaScript files you created and all the files necessary for their execution (HTML, resources). Use the “strict” mode of JavaScript, indentation and comments.

***Thank you for using zip (and not tar and gzip or bzip2). Zip all files directly into a single zip, do not zip the folder.***

***No spaces or accented characters or special characters in the name of the zip file.***

***This work is to be done individually.***

Please respect the file names and IDs given, otherwise automatic grading will not work and you will not get the correct grade for your work.

**Additional useful information**

Additional useful information can be found in: [Labs Overview](https://perso.telecom-paristech.fr/dufourd/cours/inf203/labs-overview.html)

**Exercise 1 - Get a file with AJAX and add its contents in the current HTML page**

Save the results of this exercise to a file named exercise1.js

Create a text.txt file in your personal pages, which contains [this](https://labgrader.r2.enst.fr/files/js2/m.txt)

Create an html page named ex1.html that uses exercise1.js, and place it in your server space.

Question 1a: Write a function named loadDoc that loads the file text.txt and includes it in the page when you click a button. The id of this button shall be bouton1. The imported text is put in a textarea element placed below the button. The id of the textarea node shall be textarea.

Question 1b: Write a function named loadDoc2 that inserts the text no longer in a textarea, but with each line in a p element with a style attribute that assigns different colors to each line of inserted text. The id of the button shall be bouton2 and the id of the div containing your p elements shall be textarea2.

Automatic correction relies on your respect of filenames and ids.

**Exercise 2 - Single Chat**

Create the chat.php file in your personal pages from these lines:

<?php

$chaine = gethostbyname($\_SERVER['REMOTE\_ADDR']) ;

$chaine .= " - " . $\_GET['phrase'] . "\n";

$fp = fopen("chatlog.txt","a");

if ($fp == false) {

echo "Permission error on chatlog.txt: do 'chmod a+w chatlog.txt'";

} else {

fwrite($fp, $chaine);

fclose($fp);

echo "Success";

}

?>

If you access http://perso.telecom-paris.fr/~YOU/tpinf203js2/chat.php?phrase=tatetitotu then the above file will add “tatetitotu” to the file chatlog.txt present in the same folder. You only have to reload chatlog.txt to see the new line of the chat appear. YOU is your login name.

The HTTP method you need to use is GET. When using the GET method, the parameters are sent as text in the URL after the ?. In the URL displayed above, the parameters are: ?phrase=tatetitotu

Question 2.1: Create files ex2.html and exercise2.js that contain:

* A text field to enter the new sentence, with the id textedit.
* A send button to send the new sentence with chat.php and delete the text field, with the id sendbutton.
* A div for the content of the chat with the id textarea; NOTE: do not put the chat text directly into the div with br tags but structure this chat into paragraphs using p elements. There should be no extra empty p elements or text nodes in that div.
* A loop that reloads the contents of the chatlog.txt file (= the chat itself). Display the text in p elements. Make sure you do not not have empty p elements at the beginning or end. Redisplay the text every second, no more.

You will be able to use the AJAX data retrieval function of Exercise 1 and the setTimeout function for the display loop.

If you still get the same text by Ajax while the file has changed, it may be the cache of the HTTP server, configure the .htaccess for no caching. Place this file at the folder on your web server.

# DISABLE CACHING: to include in .htaccess

<IfModule mod\_headers.c>

Header set Cache-Control "no-cache, no-store, must-revalidate"

Header set Pragma "no-cache"

Header set Expires 0

</IfModule>

Edit these files ex2.html and exercise2.js to include

* a display of the last 10 messages only
* the last line sent is displayed at the top

WARNING: do not put URLs with http://perso.telecom-paris.fr/~YOU/tpinf203js2/ in ex2.html and exercise2.js. When deployed and used in a different configuration, these URLs will stop working. In the files, use relative URLs, e.g. chat.php rather than http://perso.telecom-paris.fr/~YOU/tpinf203js2/chat.php

The labgrader will deploy your code on another machine, so your grade depends on heeding this warning.

WARNING2: do not change the content of the file ‘chat.php’, the grader assumes that specific content and will not grade your work correctly if you change it.

**Exercise 3 - Slides in JSON**

Put the slides.json file on your personal space from the content below. This file is a specification of a 12s slideshow. Each slide consists of one start time in seconds and a URL to present

{

"slides": [

{

"time": 0,

"url": "https://perso.telecom-paris.fr/dufourd/cours/inf203/"

},

{

"time": 2,

"url": "https://perso.telecom-paris.fr/dufourd/cours/inf203/js.html#/loops"

},

{

"time": 4,

"url": "https://perso.telecom-paris.fr/dufourd/cours/git/git-advanced.htm#/git-in-practice"

},

{

"time": 6,

"url": "https://perso.telecom-paris.fr/dufourd/cours/inf203/svg.html#/svg-arcs"

},

{

"time": 8,

"url": "https://inf103.telecom-paris.fr/inf103/JavaClassesObjets.html#slide24"

},

{

"time": 10,

"url": ""

}

]

}

Question 3.1: Create files ex3.html and exercise3.js that contain:

* An empty div with id="MAIN" to receive the slides
* A script to load exercise3.js
* A function that loads the slides.json file with AJAX and renders the object described in the file
* A function that plays the slideshow: at the time indicated by time, empty the div with id="MAIN" and add in this div an iframe pointing to the given URL. The id of the play button shall be pl.

You can use the setTimeout function.

Question 3.2: create an ex4.html file and exercise4.js that contain the previous features plus:

* A “pause/continue” button with id="pauseButton"
* A “next slide” button with id="suivant" and a “previous slide” button with id "but\_prev", which interrupts the playing of the slide show. These buttons should work even if the slideshow was not played.

In this version, you have to change the algorithm substantially from the simple version that worked for the previous question.

Note: the grader assumes that

* after a fresh load, the slideshow is empty
* after a fresh load, the first click on next shows the first slide (not the second)
* the grader needs next to work in order to test previous, so if next fails, previous will fail automatically