Rapport de la semaine de 2/7/2018 :

L’efficacité et le temps d’exécution :

Afin de pouvoir mieux comparer le temps d’exécution des trois méthodes, on procède la comparaison de ce dernier en deux phases. Premièrement on compare le temps moyen de compilation des fichier .ml en comparant les temps système, utilisateur et réel puis on comparera le temps d’exécution des fichiers .js .

Ocaml :
J’ai écrit un script shell ainsi qu’un programme caml qui lance la commande time sur un programme caml (un programme qui affiche une simple fenêtre alert avec un message dans le navigateur) pour les 3 méthodes et pour un nombre d’exécution passé en paramètre , ce script renvoie donc le temps moyens d’exécution réel , système et utilisateur pour les trois méthodes sur un nombre x d’exécution passé en paramètre.
Les résultats sont disponibles sur les graphes de en pièce jointes.

JavaScript :
Comment on mesure l’efficacité temps en JavaScript ?
Cette efficacité n’est mesurée que par le temps réel , ainsi on choisi un point de depart au début du script où on initialise une date , un point d’arrivée à la fin du script où on arrête l’écoulement du temps depuis le point de départ , puis on affiche le résultat de la soustraction du deuxième point par le premier point .Pour se faire il existe plusieurs méthodes en JavaScript :
La meilleure méthode de mesure de l’efficacité d’un code en JavaScript jusqu’à présent est la fonction performance :
ainsi on pourra mesurer l’efficacité du code suivant en milliseconde :

```
var t0 = performance.now() ;
doSomething() ;
var t1 = performance.now() ;
console.log(t1-t2 + millisecondes )
```

la deuxième méthode utilise l’objet date :

```
var t0 = date.now() ;
doSomething() ;
var t1 = date.now() ;
console.log(t1-t2 + millisecondes )
```

Js_of_ocaml :

La méthode performance n'est pas encore implémentée en js_of_ocaml on peut toujours utiliser la méthode un peu plus ancienne et moins efficace avec les objets date.

Obrowser :

Cette bibliothèque n'a aucune des méthodes time , date ou performance par conséquent on n'a qu'une seule façon de mesurer le temps et il s'agit de la fonction time de la bibliothèque standard de caml pour le temps réel.

BuckleScripts :

Toutes les méthodes telles que performance , date et time sont disponible en BuckleScript.

Les programmes et les comparaisons :

Afin de réaliser la comparaison pour les fichier javaScript j'ai procéder de façon suivante:
J'ai rajouté aux trois programmes JavaScript que j'avais déjà écrit une deuxième alert affichant le temps écoulée entre le début et la fin du script en millisecondes.Puis dans le but de réaliser des fichier .txt contenant les temps sur plusieurs exécution j'ai fait une deuxième version des programmes qui en plus d'effectuer l'affichage ouvre un fichier et rajoute les temps (écriture avec la bibliothèque Caml) mais plusieurs problème ont été rencontré :
en BuckleSctipt j'ai rencontré l'erreur suivante :

```
Uncaught Error: caml_sys_open not implemented by BuckleScript yet
```

donc cette possibilité n'a pas été encore implémentée en BuckleScript.
En Js_of_ocaml le problème vient au niveau de la compilation.
Le travail à realiser : On pourra essayer d'utiliser des fonction de lecture/écriture en fichier de JavaScript ultérieurement.