420KBG – Activité 00

Pour cette première activité de la session, nous allons « nous faire la main » et exposer un petit service Web de type `REST`.

Idée générale

L'idée est, en s'inspirant du petit service Web de type `REST` réalisé sans cadriciel à l'aide de Node.js[1], d'exposer par vous-même un service offrant des fonctions de nature mathématique, et de tester ce service. Les tests peuvent être réalisés à l'aide d'un fureteur Web ou d'un outil tel que Postman.

Fonctions à offrir

Votre service doit offrir au minimum les fonctions suivantes. Notez que pour cette activité, le seul verbe `http` dont nous aurons besoin sera `GET`.

Fonction	Exemple d'appel
Soit deux nombres x et y, calculer et retourner $x + y$	`api/math?op=+&x=50&y=25`
Soit deux nombres x et y, calculer et retourner $x - y$	`api/math?op=-&x=50&y=25`
Soit deux nombres x et y, calculer et retourner $x \times y$	`api/math?op=*&x=50&y=25`
Soit deux nombres x et y, calculer et retourner $\frac{x}{y}$	`api/math?op=/&x=50&y=25`
Soit deux nombres x et y, calculer et retourner $x\%y$	`api/math?op=%&x=50&y=7`
Soit un entier n, calculer et retourner $n!$	`api/math?op=!&n=5`
Étant donné un entier n, retourner `true` seulement si n est premier	`api/math?op=p&n=5`
Étant donné un entier n, retourner la valeur du $n$ième premier	`api/math?op=np&n=5`

Notez que chaque fonction est paramétrée par un code d'opération (p.ex. : `+` pour la somme, `!` pour la factorielle, `np` pour $n$ième premier, …) et par des paramètres nommés (`x`, `y`, `n`) comme le veut l'usage dans de tels services.

[1] Voir http://h-deb.clg.qc.ca/Sources/service-REST-JavaScript-sans-cadriciel.html pour des détails.

Décoder des paramètres

Nicolas Chourot, très gentil, vous offre ce petit brin d'information pour décoder les paramètres d'une requête `http` de type `GET` :

Avec le module `'query-string'` (pour l'installer : `npm install query-string` à travers la console de Node.js, ou simplement à travers la console de Windows), il est relativement facile de décomposer les paramètres d'une requête `GET`. Par exemple :

```
const queryString = require('query-string');

const parsed = queryString.parse("?op=/&x=123&y=5");

console.log(parsed);
```

… affichera :

```
[Object: null prototype] { op: '/', x: '123', y: '5' }
```

Tests à réaliser

Vous devez envisager au moins les problèmes suivants :

- Liste de paramètres incorrecte :
  - paramètres manquants,
  - paramètres de trop,
  - opération inexistante,
  - paramètres répétés, etc.
- Types incohérents. Notez que JavaScript n'est pas un langage fortement typé, mais il y a tout de même des enjeux dont il vous faudra tenir compte, par exemple le cas où on attendrait un nombre mais on recevrait quelque chose qui ne se convertit pas en nombre, ou encore le cas où on attendrait un entier mais où le nombre fourni ne serait pas convertible en entier sans perte

Notez qu'il existe une liste de code de succès ou d'erreur `http`[2], parmi lesquels `422` peut servir à signaler des paramètres incorrects.

[2] https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

Modalités de remise

Organisation :	Travail individuel ou par équipe de deux
Date de remise :	**Vendredi le 18 septembre 2020 à 23 h 59**
Remise par Colnet?	Oui, dans une archive zip nommée[3] comme suit : `Groupe-NomPrénom-Activité00.zip` (p. ex. : `05-TromblonGaetan-Activité00.zip`) ou encore, dans le cas d'une équipe de deux `Groupe-Nom0Prénom0-Nom1-Prénom1-Activité00.zip` Cette archive doit contenir : • Vos fichiers sources Node.js • Un document listant les `URL` utilisées pour fins de tests (à la fois dans le cas des succès et des échecs), de même que les résultats retournés dans chaque cas. Vous pouvez utiliser un format `.docx` ou `.pdf`, mais assurez-vous que ce soit clair et lisible.

Amusez-vous bien!

[3] http://h-deb.clg.qc.ca/CLG/Cours/demander-aide.html#remise_travaux