LABORATOIRE 4: À CHACUN SA FONCTION

À REMETTRE, AU PLUS TARD, LE 11 MAI À 12:00 via LÉA

1. Pour commencer

1. Considérant le dossier **L:\420-105** comme votre répertoire de travail, créez le dossier **L:\420-105\Labo4**.
2. Copiez les fichiers **Labo4.htm** et **libValidationBase.js** dans votre dossier **Labo4**.
3. Démarrer l'application ***Komodo Edit/Visual Studio Code***.
4. Ouvrez le fichier **L:\420-105\Labo4\Labo4A.htm.**
5. Ouvrez le fichier **L:\420-105\Labo4\libValidationBase.js.**
6. Au début du script **libValidationBase.js**, au niveau des auteurs, remplacez **VotreNom** par votre vrai nom.

2. Création d'une librairie de fonctions

Une librairie (ou une bibliothèque) de fonctions est un ensemble de fonctions centrées autour d'un thème particulier.

Par exemple, la librairie **libValidationBase.js** est une librairie de fonctions qui concernent des validations de base.

Cette session-ci, nous allons utiliser, à plusieurs reprises, la librairie de fonctions **libValidationBase.js** lorsque nous allons avoir des validations à effectuer. Cela va nous éviter d'avoir à réécrire du code puisque ces fonctions vont avoir déjà été programmées.

En cours de session, il est possible que d'autres fonctions soient ajoutées dans cette librairie au fur et à mesure des besoins.

Certaines fonctions ont déjà été programmées (par moi). Vous devez réaliser la programmation des autres fonctions.

En programmation, dans la mesure du possible, on ne réinvente pas la roue. Si la fonction a déjà été programmée et a bien été testée, il est inutile de la programmer de nouveau. On doit simplement la réutiliser (c'est-à-dire l'appeler).

```
// Retourne true si la donnée vaut null
// sinon retourne false
function estNull(donnee) {
    return (donnee == null);
}
```

Dans la fonction suivante, qui vérifie si une donnée est un nombre, on appelle la fonction **estNull()** qui a déjà été programmée.

```
// Retourne true si la donnée est un nombre (de type number)
// sinon retourne false
function estUnNombre(fltDonnee) {
    return ((!estNull(fltDonnee)) && (typeof fltDonnee == 'number'));
}
```

Lorsque vous vérifiez l'exécution d'une fonction, faites le maximum de tests.

Par exemple, pour la fonction **estUnNombre()**, vous devriez faire un minimum de trois (3) tests :

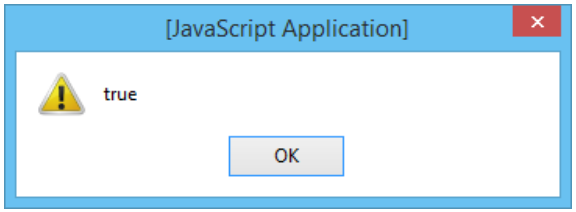
- lorsque la donnée vaut **null**,
- lorsque la donnée est un nombre (par exemple : **5, 10.6, -1,** etc.),
- lorsque la donnée n'est pas un nombre (par exemple : '**5**', '**allo**', **true**, etc.).

N'hésitez pas à tester INTENSIVEMENT chacune de vos fonctions. Souvent, on imagine, à tort, que la fonction a bien été programmée.

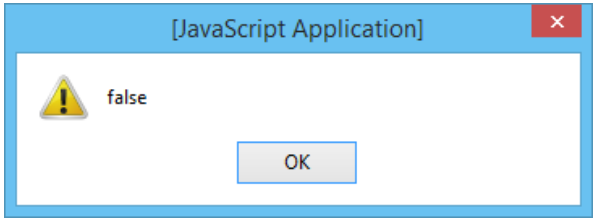
Vous pouvez, à l'intérieur de la fonction **main(),** utiliser la fonction **alert()** pour tester l'exécution de vos fonctions.

Par exemple, pour tester l'exécution de la fonction **estDansIntervalle()** :

```
function main() {
    alert(estDansIntervalle(6,4,8))
}
```

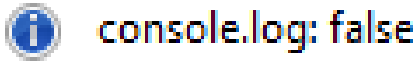



```
function main() {
    alert(estDansIntervalle(9,4,8))
}
```



Vous pouvez également (si vous le voulez) écrire le résultat de l'exécution directement sur la console.

```
function main() {
    console.log(estDansIntervalle(9,4,8))
}
```



3. A votre tour maintenant

function estDansIntervalle(donnee, valeur1, valeur2)

Cette fonction doit retourner **true** si la valeur de la donnée passée en paramètre est située entre la **valeur1** inclusivement et la **valeur2** inclusivement sinon elle doit retourner **false**.

Attention : Les trois paramètres doivent être de même type (utiliser la fonction **sontDeMemeType()** qui a déjà été programmée). S'ils ne sont pas de même type, la fonction doit retourner **false**.

Par exemples :
estDansIntervalle(6, 4, 8) doit retourner **true**
estDansIntervalle(4, 4, 8) doit retourner **true**
estDansIntervalle(8, 4, 8) doit retourner **true**
estDansIntervalle(9, 4,8) doit retourner **false**
estDansIntervalle(3, 4, 8) doit retourner **false**
estDansIntervalle(6, 8,4) doit retourner **false**
estDansIntervalle('6', 4, 8) doit retourner **false**
estDansIntervalle(6, '4', 8) doit retourner **false**
estDansIntervalle('d', 'a', 'z') doit retourner **true**
estDansIntervalle(('d', 'e', 'z')) doit retourner **false**
estDansIntervalle('D', 'A', 'Z') doit retourner **true**
estDansIntervalle('Jean', 'Janette', 'Julius') doit retourner **true**

Prendre le temps de bien vérifier

function contientSeulementUnNombre(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères contient seulement un nombre sinon elle doit retourner **false**.

Attention : La donnée doit être une donnée de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si la donnée n'est pas de type **string**, la fonction doit retourner **false**.

Par exemples :
contientSeulementUnNombre(60) doit retourner **false**
contientSeulementUnNombre(60.1) doit retourner **false**
contientSeulementUnNombre('') doit retourner **false**
contientSeulementUnNombre(' ') doit retourner **false**
contientSeulementUnNombre('60') doit retourner **true**
contientSeulementUnNombre('60s') doit retourner **false**
contientSeulementUnNombre('60.1') doit retourner **true**
contientSeulementUnNombre('Allo') doit retourner **false**
contientSeulementUnNombre('-60') doit retourner **true**
contientSeulementUnNombre('-60.1') doit retourner **true**
contientSeulementUnNombre(true) doit retourner **false**

Prendre le temps de bien vérifier

function contientSeulementUnNombreEntier(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères contient seulement un nombre entier sinon elle doit retourner **false**.

Attention : La donnée doit, tout d'abord, contenir seulement un nombre (utiliser la fonction **contientSeulementUnNombre()** que vous avez déjà programmée). Si la donnée ne contient pas seulement un nombre, la fonction doit retourner **false**.

Par exemples :
contientSeulementUnNombreEntier('60') doit retourner **true**
contientSeulementUnNombreEntier('60.1') doit retourner **false**
contientSeulementUnNombreEntier(60) doit retourner **false**
contientSeulementUnNombreEntier(60.1) doit retourner **false**
contientSeulementUnNombreEntier('Allo') doit retourner **false**
contientSeulementUnNombreEntier('60s') doit retourner **false**
contientSeulementUnNombreEntier('-60') doit retourner **true**
contientSeulementUnNombreEntier ('-60.1') doit retourner **false**
contientSeulementUnNombreEntier('') doit retourner **false**
contientSeulementUnNombreEntier(' ') doit retourner **false**
contientSeulementUnNombreEntier(true) doit retourner **false**

Prendre le temps de bien vérifier

function estUneChaineVide(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères est une chaîne qui ne contient aucun caractère (même pas un espace) sinon elle doit retourner **false**.

Attention : La donnée doit être une donnée de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si la donnée n'est pas de type **string**, la fonction doit retourner **false**.

estUneChaineVide(0) doit retourner **false**
estUneChaineVide(null) doit retourner **false**
estUneChaineVide('') doit retourner **true**
estUneChaineVide(' ') doit retourner **false**
estUneChaineVide('\n') doit retourner **false**
estUneChaineVide('Allo') doit retourner **false**

Prendre le temps de bien vérifier

function estUneChaineBlanche(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères est une chaîne vide ou une chaine qui ne contient que des caractères blancs (espaces, changements de ligne, tabulation).

Attention : La donnée doit être une donnée de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si la donnée n'est pas de type **string**, la fonction doit retourner **false**.

estUneChaineBlanche(0) doit retourner **false**
estUneChaineBlanche(' ') doit retourner **true**
estUneChaineBlanche('\n') doit retourner **true**
estUneChaineBlanche('\t') doit retourner **true**
estUneChaineBlanche('\t\n \n\t') doit retourner **true**
estUneChaineBlanche('\t\n0\n\t') doit retourner **false**
estUneChaineBlanche(' 0 ') doit retourner **false**
estUneChaineBlanche('Allo') doit retourner **false**
estUneChaineBlanche(' ') doit retourner **true**
estUneChaineBlanche(' Allo ') doit retourner **false**

Prendre le temps de bien vérifier

function estUnCaractereNumerique(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères contient un seul chiffre (un caractère numérique) sinon elle doit retourner **false**.

Attention : La donnée doit être une donnée de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si la donnée n'est pas de type **string**, la fonction doit retourner **false**.

Indice : Vous devez vous assurer qu'il n'y a qu'un et un seul caractère dans la chaîne et que ce caractère est situé dans l'intervalle **['0', '9']** (vous devez donc utiliser la fonction **estDansIntervalle()**).

Par exemples :
estUnCaractereNumerique(0) doit retourner **false**
estUnCaractereNumerique(null) doit retourner **false**
estUnCaractereNumerique(6) doit retourner **false**
estUnCaractereNumerique(61) doit retourner **false**
estUnCaractereNumerique('') doit retourner **false**
estUnCaractereNumerique ('6') doit retourner **true**
estUnCaractereNumerique('-6') doit retourner **false**
estUnCaractereNumerique ('61') doit retourner **false**
estUnCaractereNumerique('D') doit retourner **false**
estUnCaractereNumerique ('6D') doit retourner **false**

Prendre le temps de bien vérifier

function estUnCaractereAlpha(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères contient une seule lettre (un caractère alphabétique) sinon elle doit retourner **false**.

Attention : La donnée doit être une donnée de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si la donnée n'est pas de type **string**, la fonction doit retourner **false**.

Indice : Vous devez vous assurer qu'il n'y a qu'un et un seul caractère dans la chaîne et que ce caractère est situé dans l'intervalle **['A', 'Z']** ou dans l'intervalle **['a', 'z']** (vous devez donc utiliser la fonction **estDansIntervalle()**).

estUnCaractereAlpha(0) doit retourner **false**
estUnCaractereAlpha(null) doit retourner **false**
estUnCaractereAlpha(6) doit retourner **false**
estUnCaractereAlpha(true) doit retourner **false**
estUnCaractereAlpha('') doit retourner **false**
estUnCaractereAlpha('6') doit retourner **false**
estUnCaractereAlpha('d') doit retourner **true**
estUnCaractereAlpha('a') doit retourner **true**
estUnCaractereAlpha('D') doit retourner **true**
estUnCaractereAlpha('ZD') doit retourner **false**
estUnCaractereAlpha('Z') doit retourner **true**
estUnCaractereAlpha('z') doit retourner **true**
estUnCaractereAlpha(',') doit retourner **false**
estUnCaractereAlpha(' $') doit retourner **false**

Prendre le temps de bien vérifier

function estUnCaractereAlphaNumerique(strDonnee)

Cette fonction doit retourner **true** si une chaîne de caractères contient un seul chiffre ou une seule lettre (un caractère alphanumérique) sinon elle doit retourner **false**.

Indice : Utiliser les deux fonctions précédentes.

Prendre le temps de bien vérifier

function estUnCaractereValide(strCaractere, strChoixCaracteres)

Cette fonction doit retourner **true** si un caractère fait partie d'un choix de caractères sinon elle doit retourner **false**.

Par exemple, **estUnCaractereValide('Q', 'PQR')** devrait retourner **true** . Par contre, **estUnCaractereValide('S', 'PQR')** devrait retourner **false**.

Attention : Le caractère et les choix doivent être des données de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si un des deux n'est pas de type **string**, la fonction doit retourner **false**.

Attention : II ne doit y avoir qu'un et un seul caractère dans **strCaractere**. Si ce n'est pas le cas, la fonction doit retourner **false**.

Prendre le temps de bien vérifier

function estDansUnFormatValide(strDonnee, strFormat)

Cette fonction doit retourner **true** si une donnée est dans un format valide sinon elle doit retourner **false**.

Au niveau du format, le caractère # représente un chiffre, le caractère @ représente une lettre et tout autre caractère représente le caractère lui-même.

Par exemple, le format d'un code postal est '**@#@ #@#'** qui signifie une lettre suivie d'un chiffre suivie d'une lettre suivie d'un espace suivi d'un chiffre suivi d'une lettre suivi d'un chiffre.

Par conséquent, **estDansUnFormatValide('G9H 3N6', '@#@ #@#')** devrait retourner **true**. Par contre, **estDansUnFormatValide('G9H 3NB', '@#@ #@#')** devrait retourner **false**.

Attention : La donnée et le format doivent être des données de type **string** (utiliser la fonction **estUneChaine()** qui a déjà été programmée). Si un des deux n'est pas de type **string**, la fonction doit retourner **false**.

Indice : Vérifier tout d'abord si ce sont deux (2) chaînes et si elles ont le même nombre de caractères. Si c'est le cas, vous devez, dans une boucle **for**, allez chercher, caractère par caractère, le caractère #i de la donnée et le caractère #i du format. Si le format est le caractère '@', utiliser la fonction **estUnCaractereAlpha()**; si le format est le caractère '#', utiliser la fonction **estUnCaractereNumerique()**; si ce n'est ni un ni l'autre, vérifier l'égalité entre les 2 caractères.

Prendre le temps de bien vérifier

function estUnCodePostal(strDonnee)

Cette fonction doit retourner **true** si une donnée est un code postal sinon elle doit retourner **false**.

Le format d'un code postal est '**@#@ #@#**'

Attention : Utiliser la fonction **estDansUnFormatValide().**

function estUnNAS(strDonnee)

Cette fonction doit retourner **true** si une donnée est un numéro d'assurance sociale sinon elle doit retourner **false**.

Le format d'un NAS est '**### ### ###**' (par exemple : '**546 345 567**' est un numéro d'assurance sociale).

Attention : Utiliser la fonction **estDansUnFormatValide().**

function estUnMatricule(strDonnee)

Cette fonction doit retourner **true** si une donnée est un matricule sinon elle doit retourner **false**.

Le format d'un matricule est '**#######**' (par exemple : '**1234567**' est un matricule).

Attention : Utiliser la fonction **estDansUnFormatValide().**

function estUnCodePermanent(strDonnee)

Cette fonction doit retourner **true** si une donnée est un code permanent sinon elle doit retourner **false**.

Le format d'un matricule est ' **@@@@########'** (par exemple : '**MTRE15078912**' est un code permanent).

Attention : Utiliser la fonction **estDansUnFormatValide().**

function estUnNoDeTelephone(strDonnee)

Cette fonction doit retourner **true** si une donnée est un numéro de téléphone sinon elle doit retourner **false**.

Le format d'un numéro de téléphone est '**(###) ###-####'** (par exemple : '**(514) 564-5676**' est un numéro de téléphone)

Le format suivant aussi doit être accepté '**###-###-####'** (par exemple : '**514-564-5676**' est un numéro de téléphone)

Attention : Utiliser la fonction **estDansUnFormatValide().**

4. Remise

- ✓ Compressez votre dossier **L:\420-105\Labo4.**
- ✓ Transmettez le fichier **Labo4.zip** sur **LEA**.