Fonctions FormPublisher

FormPublisher est pourvu d'une bibliothèques de fonctions pour opérer sur des dates, des textes, des calculs, Array, Technique etc.

1. Dates

Fonctions dates.

1.1. Fonction getCurrentDate

La fonction getCurrentDate permet de retourner la date courante sur deux formes :

1. La fonction getCurrentDate() sans paramètre : elle retourne la date courante sous forme numérique entier
   Syntaxe : Integer getCurentDate()
2. La fonction getCurrentDate('dateFormat') avec paramètre : elle retourne une chaîne de caractères représentant la date courante formatée selon le format spécifié.
   Syntaxe : String getCurrentDate('dateFormat')

Exemple 1 : utilisation de getCurrentDate()

Voici la date d'aujourd'hui représentée en valeur numérique entier :

Un coup d'œil sur comment le reproduire.

Exemple 2 : utilisation de getCurrentDate('dd/MM/yyyy HH:mm')

Voici la date d'aujourd'hui représentée en chaine de caractères selon le format spécifié :

Un coup d'œil sur l'implémentation.

1.2. Fonction getDate

La fonction getDate est l'inverse de formatDate. Elle évalue une valeur de type chaîne de caractères et retourne la valeur numérique entier correspondante en fonction du format date spécifié.

Syntaxe : Integer getDate(date_en_String, dateFormat)

Exemple : getDate('06/06/2005', 'dd/MM/yyyy')

: Est la valeur numérique entier de la date **06/06/2005.** Le format spécifié pour le calcule est : dd/MM/yyyy

1.3. Fonction formatDate

La fonction formatDate est l'inverse de getDate. Elle évalue une valeur date de type numérique entier et retourne la valeur correspondante en chaine de caractères.

Syntaxe : String formatDate(Integer date1, String dateFormat)

Exemple : formatDate(getCurrentDate(), 'dd/MM/yyyy HH:mm')

: Est la date courant **au** format : dd/MM/yyyy HH:mm

1.4. Fonction isDateCorrect

Elle teste si une valeur textuelle (representant une date), correspond au format.

Les caractères significatifs du format sont 'd', 'm' et 'y'.

Syntaxe : Boolean isDateCorrect(String date1, String dateFormat)

Paramètre 1: une valeur (date sous forme de texte) e

Paramètre 2: le format à vérifier

Résultat: Renvoie true si la valeur correspond au format attendu, false sinon.

Exemple:

Saisissez une date:		format: dd-mm-yy

Format date correct.

Format date erroné.

1.5. Fonction isTimeCorrect

Elle teste si une valeur textuelle (representant un horaire), correspond au format.

Les caractères significatifs du format sont 'h' ou 'H', 'm' ou 'M' et 's' ou 'S'.

Syntaxe : Boolean isTimeCorrect(String heure1, String timeFormat)

Paramètre1: une valeur (heure sous forme de texte)

Paramètre2: le format à vérifier

Résultat: Renvoie true si la valeur correspond au format attendu, false sinon.

Exemple:

Heure:		format: hh:mm:ss

Format heure correct.

Format heure erroné.

1.6. Fonction dateDiffInDays

Cette fonction retourne une valeur entière représentant la différence en jours entre deux dates. Ces deux dates sont des valeurs de type numérique entier (obtenus par exemple avec la fonction getDate()).

Syntaxe : dateDiffInDays(Integer1 , Integer2)

Le résultat est négatif si la première date est postérieure à la seconde (date1 > date2).

Pour les différences non multiples de périodes de 24 heures, voir la fonction dateDiffInDaysRound().

Exemple :

Le formulaire ci-dessous permet à l'utilisateur de saisir deux dates et calcule la différence en jours entre les deux dates saisies. La deuxième date doit être postérieure à la première. Pour vérifier cette contrainte, des contrôles (startDate, endDate) sont appliqués aux champs de saisie (via un groupe de contrôles). La valeur par défaut du premier champ est la date du jour.

Date de début :	/ /
Date de fin :	/ /
Le nombre de jours entre ces deux dates est : jours	

L'implémentation de cet exemple est disponible ici

1.7. Fonction dateDiffInDaysRound

Cette fonction retourne une valeur de type entier représentant la différence **approximative** de jours entre deux dates. C'est à dire, le nombre entier le plus proche du nombre de tranches de 24 heures. Le résultat est signé et négatif si la première date est postérieure à la seconde (date1 > date2). Les deux dates en entrée sont des valeurs de type numérique entier.

Syntaxe : dateDiffInDaysRound(Integer1, Integer2)

Exemple :

A partir de l'exemple précédent la fonction dateDiffInDaysRound calcule la différence de jours entre les deux dates précédemment saisies par l'utilisateur.

> Le nombre de jours entre les deux dates calculé avec la fonction dateDiffInDaysRound est :
> jours

L'implémentation de cet exemple est disponible ici

1.8. Fonction dateDiffInHours

Cette fonction retourne une valeur de type entier représentant la différence **d'heures** entre deux dates en entrée. Le résultat est signé et négatif si la première date est postérieure à la seconde (date1 > date2). Les deux dates en entrée sont des valeurs de type numérique entier.

Syntaxe : dateDiffInHours(Integer1, Integer2)

Exemple :

Calcul du nombre d'heures entre les dates saisies de l'exemple précédant.

Le nombre de jours entre ces deux dates est : heures

L'implémentation de cet exemple est disponible ici

1.9. Fonction compareDate

Cette fonction compare deux dates et retourne un entier numérique indiquant si elles sont égales ou différentes.
La valeur retournée est soit :

- - 1 : la première date est antérieure à la deuxième
- 0 : les deux dates sont égales
- 1 : La première date est postérieure à la deuxième

Syntaxe :

La fonction compareDate peut prendre deux ou trois paramètres.

- Integer compareDate (Integer date1, Integer date2)
- Integer compareDate (String date1, String date2, String format)

Exemple:

Dans cet exemple, l'utilisateur doit saisir trois champs : un format date, et deux champs de saisie date sans contrôle.

Format des dates à saisir :			
Date effective		Date souhaitée	

Le résultat de la comparaison en utilisant compareDate à deux paramètres :
Le résultat de la comparaison en utilisant compareDate à trois paramètres :

Veuillez cliquer ici pour l'aperçu de l'implémentation.

1.10. Fonction compareDateDiffInDays

Cette fonction détermine si la différence en jours entre deux dates en paramètre correspond ou non au nombre de jours indiqué en paramètre.

La fonction compareDateDiffInDays calcule la différence en nombre de jours entre deux dates (identique à getNbDays) ; puis, compare la valeur obtenue au nombre de jours passé en paramètre. La comparaison se fait avec le symbole logique passé en paramètre exprimé en chaîne de caractères.

Le résultat étant boolean, la valeur de retour sera :

- True si l'expression est vérifiée
- False si l'expression n'est pas vérifiée

La fonction compareDateDiffInDays prend quatre paramètres si les dates en paramètres sont exprimées en numérique entier sinon, elle prend cinq pour les dates exprimées en chaîne de caractères.

Syntaxe :

Boolean compareDateDiffInDays(Integer1, Integer2, Integer D, String)

Boolean compareDateDiffInDays(String1, String2, dateFormat, Integer D, String)

- Paramètres 1 et 2 : dates exprimées soit en numérique entier soit en chaine de caractères
- Paramètre 3 : nombre de jours avec lequel sera comparée la différence entre les 2 dates
- Paramètre 4 : symbole de comparaison exprimé en chaine de caractères
- Paramètre 5 : format dans lequel les dates ont étés exprimées

Exemple:

En s'inspirant de l'exemple précédant, nous déterminons s'il y a plus ou moins de 10 jours entre la date effective et la date souhaitée précédemment saisies (exemple compareDate).

compareDateDiffInDays à quatre paramètres : compareDateDiffInDays($(date_effec1), $(date_souh2), 10, '<=')

La différence de jours entre les deux dates est-elle inférieur ou égale à 10 jours ?

compareDateDiffInDays à cinq paramètres : compareDateDiffInDays($(date_effective), $(date_souhaite), $(format_date), 10, '>')

Y a t-il plus de 10 jours entre les deux dates ?

1.11. Fonction addMonths

Cette fonction ajoute ou soustrait un nombre de mois à une date donnée et retourne une valeur de type numérique entier représentant la date résultante.

Syntaxe :

Integer addMonths(String, Integer, dateFormat)

- Paramètre 1 : date exprimée en chaine de caractères
- Paramètre 2 : nombre de mois à ajouter ou à soustraire
- Paramètre 3 : format dans lequel la date a été exprimée (dd/MM/yyyy)

Exemple :

Date :	/ /
Nombre de mois à ajouter/soustraire :	
Résultat :	

1.12. Fonction addDays

Cette fonction ajoute ou soustrait un nombre de jour(s) à une date donnée et retourne une valeur de type numérique entier représentant la date résultante.

Syntaxe :

Integer addDays(String, Integer, dateFormat)

- Paramètre 1 : date exprimée en chaine de caractères
- Paramètre 2 : nombre de jours à ajouter ou à soustraire
- Paramètre 3 : format dans lequel la date a été exprimée

Exemple :

Date :	/ /
Nombre de jour(s) à ajouter/soustraire :	
Résultat :	

1.13. Fonction addYears

Cette fonction ajoute ou soustrait un nombre d'année(s) à une date donnée et retourne une valeur de type numérique entier représentant la date résultante.

Syntaxe :

Integer addYears(String, Integer, dateFormat)

- Paramètre 1 : date exprimée en chaine de caractères
- Paramètre 2 : nombre d'année(s) à ajouter ou à soustraire
- Paramètre 3 : format dans lequel la date a été exprimée

Exemple :

Date :	/ /
Nombre d'année(s) à ajouter/soustraire :	
Résultat :	

2. Maths

Fonctions mathématiques.

2.1. Fonction modulo

Cette fonction retourne le reste de la division euclidienne du dividende par le diviseur.

Syntaxe :

Integer modulo (Integer[1], Integer[2])

- Paramètre 1 : le dividende
- Paramètre 2 : le diviseur

Exemple :

Vous souhaitez divisez :	
Par :	
Le reste de cette division est :	

2.2. Fonction asNumber

Cette fonction retourne une valeur numérique correspondant à la valeur exprimée en chaîne de caractères.

Syntaxe **:**

valeur asNumber(String)

- Paramètre : chaîne de caractères contenant des chiffres.

Exemple :

'1980' étant une chaîne de caractères vu qu'elle contient des guillemets, par usage de la fonction asNumber, nous récupérons cette chaîne comme valeur numérique. **asNumber('1980')**nous retourne:

2.3. Fonction formatNumber

Cette fonction retourne un texte correspondant à la valeur numérique d'origine (exprimée en String) avec le formatage demandé.

Le nombre de paramètres après le point correspond au nombre de chiffres souhaité après la virgule.

Syntaxe :

String formatNumber(valeur, format)

- Paramètre 1 : valeur à formater
- Paramètre 2 : format à appliquer

Exemple : formatNumber('200,1056','.##') donnera 200,10

Valeur à formater :	
Résultat :	

Souhaitez vous changer le format ? ☐

Nouveau format :	
Résultat :	

Souhaitez vous utiliser le formatString ? ☐

Exemple: formatString('352542223', '#.###.###,##')

Numérique à formater :
Format :
La fonction formatString retourne :

2.4. Fonction isNumber(*value*)

Elle teste si la valeur pourrait être interprétée comme une valeur numérique.

Syntaxe: Boolean isNumber(*value*)

Paramètre: une valeur à tester.

Résultat: Renvoie true s'il est possible de construire un nombre à partir de la valeur.

2.5. Fonction formatCurrency

Cette fonction convertit un nombre représentant un montant en une chaîne de caractères formatée avec la devise pour en faciliter la lecture.

Syntaxe : String formatCurrency(valeur, format)

- Paramètre 1 : valeur à formater
- Paramètre 2 : format à appliquer

Exemple : soit total = 3 580 246,79

formatCurrency($(total), '*€') donne 3.580.246,79€

formatCurrency($(total), '$*') donne $3.580.246,79

Note : le format utilise une étoile, à la différence de celui de formatNumber qui utilise un dièse.

3. Textes

Fonctions sur les textes.

3.1. Fonction contains

Cette fonction permet de déterminer si une chaîne de caractère contient ou pas une autre chaîne de caractère.
Les valeurs retournées sont :

- true : si la première chaîne contient la chaîne recherchée.
- false : si la première chaîne ne contient pas la chaîne recherchée.

Syntaxe :

Boolean contains(String [1], String [2])

- Paramètre 1 : chaîne de caractères dans laquelle la recherche est faite
- Paramètre 2 : chaîne de caractères recherchée

Exemple :

Veuillez s'il vous plait saisir dans le deuxième champ la chaîne de caractère à rechercher et dans le premier celle dans laquelle la recherche sera faite.

Saisir un texte :	
Je recherche :	

YES ! la chaîne recherchée est bien dans le texte.
Désolé ! le texte ne contient pas la chaîne recherchée

3.2. Fonction matchesRegexp

Cette fonction permet de déterminer si une chaine de caractère contient ou pas une autre chaine de caractère représentée par une expression régulières.

Les valeurs retournées sont :

- True : si la première chaine contient une chaine de caractère correspondant à l'expression régulière.
- False : si la première chaine ne contient pas une chaine correspondant à l'expression régulière recherchée

Syntaxe :

Boolean matchesRegexp(String [1], String [2])

- Paramètre 1 : chaine de caractères dans laquelle la recherche est faite
- Paramètre 2 : chaine de caractères représentant l'expression régulière à rechercher

Exemple : matchesRegexp('123B', '\\d+\\w*')

Veuillez s'il vous plait saisir dans le deuxième champ la chaine de caractère à rechercher et dans le premier celle dans laquelle la recherche sera faite.

Saisir un texte :	
Expression régulière à rechercher :	
La chaine recherchée est bien dans le texte. **Désolé !** le texte ne contient pas la chaine recherchée	

3.3. Fonction replace

Cette fonction retourne une chaîne de caractères correspondant à celle d'origine dont toutes les occurrences de la chaîne de recherche ont été remplacées par la chaîne de substitution.

Syntaxe : String replace(String [1], String [2], String [3])

- Paramètre 1 : chaîne dans laquelle la chaîne à remplacer sera recherchée
- Paramètre 2 : chaîne de caractère recherchée
- Paramètre 3 : chaîne de caractère de remplacement

Exemple :

Saisir un texte :	
Je recherche :	
Je remplace par :	

Après remplacement :

Désolé ! le texte ne contient pas la chaîne recherchée. Le remplacement ne peut donc être fait.

3.4. Fonction replaceRegexp

Cette fonction retourne une chaîne de caractères correspondant à celle d'origine dont toutes **les occurrences répondant à l'expression régulière** recherchée ont été remplacées par la chaîne de substitution.

Syntaxe: String replaceRegexp(String [1], String [2], String [3])

- Paramètre 1 : chaîne dans laquelle la chaîne à remplacer sera recherchée
- Paramètre 2 : chaîne de caractère représentant l'expression régulière de recherche
- Paramètre 3 : chaîne de caractère de remplacement

Exemple : replaceRegexp('Hello World', 'W.+d','Everybody')

Saisir un texte :	
Expression regulière de recherche :	
Je remplace par :	

Après remplacement :

Désolé ! le texte ne contient pas la chaîne recherchée. Le remplacement ne peut donc être fait.

3.5. Fonction getTokens

Cette fonction retourne une liste de chaînes de caractères extraite de la chaîne d'origine en isolent l'élément délimiteur.

Syntaxe : String getTokens(String t^1, String t^2)

- Paramètre 1 : chaîne à découper
- Paramètre 2 : caractère représentant l'élément délimiteur

Exemple :

Chaîne à découper :	
Elément délimiteur :	
Token :	

Désolé ! aucune liste à retourner.

3.6. Fonction extractsDigits

Cette fonction retourne une chaîne de caractères ne contenant que les chiffres (digits) présents dans la chaîne d'origine.

Syntaxe : String extractsDigits(String value)

- Paramètre : chaîne dans laquelle la recherche sera faite

Exemple :

Saisir un texte contenant des chiffres :

: sont les chiffres présents dans le texte saisi.

Désolé ! le texte ne contient pas de chiffres.

3.7. Fonction parseDynString

Cette fonction retourne une chaîne de caractères dans laquelle les variables sont replacées par leur valeur.(les constructions du type $(var) sont remplacées par la valeur de la variable "var").

Syntaxe : String parseDynString(String value)

- Paramètre : chaîne contenant la variable à remplacer

Exemple : name = 'Marie Dupong' parseDynString('Bonjour $(name)')

Votre nom :
La fonction parseDynString retourne:

3.8. Fonction formatString

Permet de formater une valeur textuelle suivant un format donné.

Syntaxe : String formatString(Integer [1], String [2]))

- Paramètre 1 : valeur numérique à formater
- Paramètre 2 : format à appliquer

Exemple : formatString('352542223', 'Tel: + ### ##-##-##')

Numérique à formater :
Format :
La fonction formatString retourne :

3.9. Fonction isFormatCorrect

Elle teste si une valeur textuelle correspond au format.
Les caractères significatifs du format sont '*' (lettre ou chiffre), '#' (chiffre), 'x' ou 'X' (lettre)

Syntaxe : Boolean isFormatCorrect(String value, String Format)

Paramètre 1: une valeur

Paramètre 2: le format à vérifier

Résultat: Renvoie true si la valeur correspond au format attendu, false sinon.

Exemple :

Code secrét:		format: ###xx#####x
Code valide.		
Code erroné.		

4. Array / list

Fonctions liste liée aux listes ou tableaux

4.1. Fonction sortList

Cette fonction retourne une liste triée par ordre alphabétique.

- La fonction prend deux paramètres: le tri est fait en fonction d'un élément.
  Syntaxe: String [] sortList(String[] tab[1], String s)
- Elle prend un paramètre.
  Syntaxe: String [] sortList(String[] tab[1])

- Paramètre [1]: tableau de chaîne de caractère à trier
- Paramètre [2]: chaîne de caractère indiquant l'élément sur lequel se fera le tri

Exemple:

Suite de l'exemple de la fonction getTokens. Nous avons trié la liste par ordre alphabétique. sortList($(tokens_list))
éléments dans la liste

,

Exemple : cet exemple illustre le tri en fonction d'un élément.
Soit la liste semaine[0]|jour='Dimanche';
semaine[1]|jour='Lundi';
semaine[2]|jour='Mardi';
semaine[3]|jour='Mercredi';
semaine[4]|jour='Jeudi';

semaine[5]|jour='Vendredi';

semaine[6]|jour='Samedi';

Le tri de la liste en fonction de l'élément "jour" est exprimé ainsi: **liste_triee = sortList($(semaine),'jour')**. Les jours seront triées par ordre alphabétique.

:

4.2. Fonction getSize

Retourne le nombre d'éléments présent dans un tableau à une dimension.

Exemple:

Soit la liste 'annee' .

list_annee [0]='janvier'

list_annee [1]='fevrier'

list_annee [2]='mars'

Cette liste contient: éléments.

,

5. Technique

Fonctions technique

5.1. Fonction exists

Elle teste l'existence d'une donnée.

Syntaxe: Boolean exists(String *value*)

Paramètre: une valeur

Résultat: renvoie false si la valeur est "null", true sinon.

A noter: la valeur peut être vide.

Exemple :

TextBox DataType string exists =

TextBox DataType string exists =

TextBox DataType string exists =

5.2. Fonction existAndNotEmpty

Cette fonction identifie si une variable existe et possède un contenu.

Syntaxe **: Boolean** existAndNotEmpty($(variable_name))

Paramétre : le nom de la variable

Exemple :

Comment savoir si, il existe dans ce document une variable '$(dateDefault)' ayant un contenu?

: Est la réponse booléenne de : " existsAndNotEmpty($(dateDefault))" nous permettant de savoir si cette variable existe.

5.3. Fonction getProperty

Cette fonction retourne la valeur d'une entrée contenue dans le fichier publication.properties.

Syntaxe **: Boolean** existAndNotEmpty(String property_name))

Paramétre : le nom de la propriété

Exemple :

Nous allons afficher la valeur de la propriété "jway.publication.name" de cette application.

getProperty('jway.publication.name') nous permets d'avoir :

6. Web

Fonctions web.

6.1. Fonction isEmailCorrect

Elle teste si la valeur est une adresse mail correcte.

Syntaxe : Boolean isEmailCorrect(*String value*)

Paramètre 1: une valeur à tester

Résultat: Renvoie true si la valeur correspond à une adresse mail valide , false sinon.

Exemple :

Email:	
Email valide.	
Email erroné.	

6.2. Fonction isHttpCorrect

Elle teste si la valeur est une URL correcte.

Syntaxe : Boolean isHttpCorrect(*String http*)

Paramètre 1: une valeur à tester

Résultat: Renvoie true si la valeur correspond à une url valide , false sinon.

Exemple :

Url:	
Url valide.	
Url erroné.	