HubSpot

Les regex

Un gain de temps
pour la gestion de contenu

Table des matières

```
/^((?!\.)[\w-_.]*[^.])
(@\w+)(\.\w+(\.\w+)
?[^.\W])$/
```

```
^[a-z0-9-]+$
```

Dans le monde informatique, pouvoir manipuler les données à volonté est un avantage considérable. Que ce soit pour rechercher des informations spécifiques, ou bien remplacer des données, les **regex** (de l’anglais ***regular expressions***) permettent d’automatiser de nombreuses opérations. Elles ont de nombreux domaines d’application et représentent un formidable gain de productivité pour celles et ceux qui en maîtrisent l’usage.

```
$1 $2 style=”display: block;
margin : 0 auto;” $3
```

```
(\d{2}[-\/\s]?){5}
```

Chapitre 1

Qu’est-ce qu’une regex ?

Qu’est-ce qu’une regex ?

Les regex, ou expressions régulières, sont des modèles utilisés pour représenter des chaînes de caractères. Elles constituent un outil très puissant dans la manipulation du code et l’extraction d’information.

Si l’apprentissage de ces expressions régulières peut paraître difficile, il n’en est pas moins bénéfique. Les regex peuvent être utilisées aussi bien pour la validation de données ou le remplacement de chaînes de caractères, que pour le web scraping, c’est-à-dire l’extraction de contenu ou d’information d’un site web.

Une fois la syntaxe maîtrisée, les regex sont facilement compatibles avec de nombreux langages informatiques : JavaScript, PHP, Java, C#, Python ou Ruby. En effet, la logique de construction reste sensiblement la même.

Pourquoi utiliser les expressions régulières ?

Les expressions régulières sont associées à de nombreux cas d’application, ce qui en fait une compétence réutilisable et performante.

Dans le domaine du développement, le premier exemple généralement donné est celui de la sécurité. À l’aide d’une regex, il est possible de vérifier si le type de données reçues correspond au type de données attendues.

Dans le domaine du marketing, où leur connaissance fait souvent défaut, les regex permettent une utilisation optimale d’outils tels que la Search Console, Google Analytics, ou encore Looker Studio pour extraire des données à des fins de SEO.

Enfin, pour la gestion d’un blog, l’utilisation de regex permet la recherche rapide d’informations précises ou la modification générale du style d’un article, sans avoir à parcourir les lignes de codes une à une.

Reste à décortiquer les regex et voir comment les utiliser de façon optimale pour la recherche d’informations et pour la modification du contenu.

Chapitre 2

Syntaxe d'une regex

Syntaxe d’une regex

La regex est une chaîne de caractères représentant un modèle et qui sert à faire correspondre, gérer, et filtrer du texte. Exemple de construction d’une regex :

/\d{5}/ (une suite de 5 chiffres)

Ou encore :

/^((?!\.)[\w-_.]*[^.])(@\w+)(\.\w+(\.\w+)?[^.\W])$/ (une adresse e-mail)

L’une paraît, à première vue, plus complexe que l’autre, même si, pour les novices, aucune des deux n’est très compréhensible.

La structure de base d’une regex

Les ancres

Regex	Explications
^La	Correspond à toute chaîne commençant par **La**.
fin$	Correspond à toute chaîne terminant par **fin**.
^La fin$	Correspond à la chaîne exacte **La fin**, puisqu’elle commence et fini par **La fin**.
Bonjour	Correspond à toute chaîne contenant **Bonjour**. Évidemment, pour cet exemple, une simple recherche textuelle permet d’obtenir le même résultat.
\bBlog\b	Correspond à **Blog** en limite de mot, c’est-à-dire en tant que mot isolé (et non pas **blogging**, ni **monblog.com**). Cette expression correspond aussi à **Blog**, précédé ou suivi de ponctuation : **Blog! / -Blog / &Blog?** Remarque : La limite de mot ne fonctionne pas sur un alphabet non latin.

Les quantificateurs

Regex	Explications	Correspondance
abc*	Correspond à toute chaîne avec ab, suivie de zéro ou plusieurs c.	ab / abcc / abccccc / ...
abc+	Correspond à toute chaîne avec ab, suivie de un ou plusieurs c.	abc / abcc / abccccc / ...
abc?	Correspond à toute chaîne avec ab, suivie de zéro ou un seul c.	ab / abc
abc{2	Correspond à toute chaîne avec ab, suivie de deux (2) c.	abcc
abc{2,}	Correspond à toute chaîne avec ab, suivie de deux (2) c ou plus.	abcc / abccccc / abcccccccccc / ...
abc{2,5}	Correspond à toute chaîne avec ab, suivie de deux (2) à cinq (5) c.	abcc / abccc / abccccc / abccccc
a(bc)*	Correspond à toute chaîne avec a, suivie de zéro ou plusieurs fois la séquence bc.	a / abcbc / abcbcbcbc / ...
a(bc){2,5}	Correspond à toute chaîne avec a, suivie de deux (2) à cinq (5) fois la séquence bc.	abcbc / abcbcbcbc / ...

Les opérateurs

Regex	Explications	Correspondance
a(b|c)	Correspond à toute chaîne avec a, suivie de b ou de c et capturant b ou c.	ab / ac
a[bc]	Correspond à toute chaîne avec a, suivie de b ou c sans capturer b ou c.	a dans abrégé / a dans accès

Les classes de caractères

Les classes de caractères permettent de définir un modèle à l'aide d'une seule lettre, ce qui produit des regex plus compréhensibles et plus courtes.

Regex	Explications
\d	Correspond à n'importe quel chiffre.
\w	Correspond à n'importe quel mot (caractères alphanumériques).
\s	Correspond à n'importe quel espace blanc (tabulation et saut de ligne compris).
\b	Correspond au début ou à la fin d'un mot.
\r	Correspond à un retour chariot.
\n	Correspond à une nouvelle ligne.
abc(\r\n|\r|\n).*xyz	Correspond à abc.*xyz, soit abc suivi de xyz, avec la possibilité que abc et xyz soient espacés de n'importe quelle suite de caractères, y compris d'un retour à la ligne.
\D	Correspond à un caractère quelconque n'étant pas un chiffre. Remarque : Dans les classes de caractères, la majuscule d'une lettre correspond généralement à l'inverse de sa minuscule.
.	Le point correspond à tout caractère unique (lettre, chiffre ou symbole).

Les groupes

Les groupes sont très importants dans les regex utilisées pour des opérations de rechercher-remplacer, car ils permettent d'isoler des éléments de l'expression. On parle alors de groupes capturants.
Les groupes capturants ne peuvent pas faire l'objet d'un remplacement par regex.

En effet, **(<h2.*?>).*?(</h2>) et <h2.*?>.*?</h2>** obtiendront les mêmes résultats lors de la recherche, mais ils ne seront pas modifiables de la même manière. Dans le premier cas, seul le contenu de la balise h2 sera modifiable, tandis que dans le second cas, l'entièreté du résultat sera remplaçable.

Symbole	Explications
Parenthèse ()	Capture les caractères compris entre les parenthèses. Il est possible d'utiliser plusieurs groupes capturants : ils seront identifiés dans l'ordre de leur apparition.
Parenthèse négatives (?!foo)	Recherche les éléments ne possédant pas foo dans leur chaîne de caractères.
Crochet []	Correspond à un ensemble de caractères, indépendamment de leur ordre ou de leur place dans la chaîne.
Tiret -	Utilisé entre les crochets, il indique une plage de caractères comme 0-9 ou A-Z.

Fonctionnement des groupes lors d'un remplacement

Dans une regex, les groupes capturants sont compris entre parenthèses. Ils ne sont pas modifiables et sont identifiables par leur place dans la regex.
(ab)(cd) est une regex constituée de 2 groupes capturants.
(Bonjour) les amis (!) est une regex constituée de 2 groupes capturants.
((Hello), wyz [bc] (comment allez-vous ?)) est une regex constituée de 3 groupes capturants. Pour appeler ces groupes, il suffit de mettre le signe dollar $ suivi du chiffre correspondant à leur ordre dans la regex.
(ab)(cd) dans une regex de recherche correspondra à **$1$2** dans une regex de remplacement.

Astuce : Pour faciliter l'utilisation des groupes dans une regex, il est possible de copier une expression régulière sur le site **Regex101**, qui indiquera le nombre de groupes et leur correspondance.

Les opérations de rechercher-remplacer

Les regex permettent de rechercher dans le code toutes les informations souhaitées en vue de les remplacer.

Exemple simple

Pour rechercher et supprimer tous les attributs de style, et ainsi remettre à zéro le style du code, il est possible d'utiliser les regex pour rechercher l'attribut et la valeur conjointement et de tous les supprimer en même temps.
Cela évite d'effectuer une recherche simple qui renverra tous les mots correspondants à **style**, qu'il faudra ensuite parcourir un à un pour supprimer l'ensemble de l'attribut.
Dans un outil d'expressions régulières, il faudra rechercher tous les attributs de style dans le code : **style="([^"]*)"**
Puis dans la case **Remplacer par**, il suffit de laisser le champ vide puis de **Tout remplacer**. Voilà, le code est maintenant vierge de style.

Exemple plus complexe

Pour modifier tous les textes des titres de niveau 2 : **(<h2.*?>).*?(</h2>)**
Cette regex va rechercher dans le code toutes les balises h2 avec ou sans attributs ainsi que leur contenu. Elle pourrait donc renvoyer comme résultat :
<h2>Cours Regex 101</h2>
<h2 class="text-white subtitle" id="sous-titre2">Cours Regex en HTML</h2>

Cette regex de recherche possède deux groupes capturants, **(<h2.*?>)** et **(</h2>)**, respectivement **$1** et **$2**.
Ces groupes ne seront donc pas modifiables, mais il est possible, si nécessaire, de ne pas les appeler dans la regex de remplacement.
Pour remplacer le contenu textuel des sous-titres, il faut appeler la balise ouvrante puis insérer le texte désiré, et appeler la balise fermante.
Cela donne : **$1 Mon nouveau titre $2**

Remarque : Les espaces sont prises en compte dans la regex de remplacement.

Chapitre 3

Outils d'aide à la construction

Outils d’aide à la construction

De nombreux outils et sites web existent pour accompagner les utilisateurs, tous niveaux confondus, dans l’utilisation, le test et la validation des regex.
À noter que certains CMS ou éditeurs de texte ne permettent pas l’utilisation de regex sans l’installation des extensions nécessaires.

Pour utiliser les regex de recherche et de remplacement

- **Un CMS, comme celui de HubSpot**
- **Notepad++**
- **VSCode**

Certains CMS nécessitent le téléchargement d’un plug-in pour pouvoir gérer les regex, tandis que Notepad++ et VSCode intègrent cette fonctionnalité dès le départ.
Notepad++
La combinaison de touches **CTRL+H** ou le menu **Rechercher > Remplacer.**
VSCode
Le raccourci **CTRL+F** pour rechercher une donnée, puis un clic sur l’icône de cette zone de recherche pour transformer la recherche en regex et activer le remplacement.

Pour tester et apprendre

- L’application **Expresso** à télécharger.
- L’application web **Regex101** qui permet un accès rapide à un champ de test pour la regex.

Chapitre 4

Astuces et bonnes pratiques

Astuces et bonnes pratiques

1 - La combinaison point, étoile et point d’interrogation (.*?)

Cette partie de regex signifie que la recherche porte sur n’importe quelle donnée, du simple point aux balises les plus complexes.
<h1>.* ?</h1> Quelle que soit la structure du h1, complexe ou non, vide ou non, cette regex retrouvera tous les titres de premier niveau.

2 - Tester encore et encore

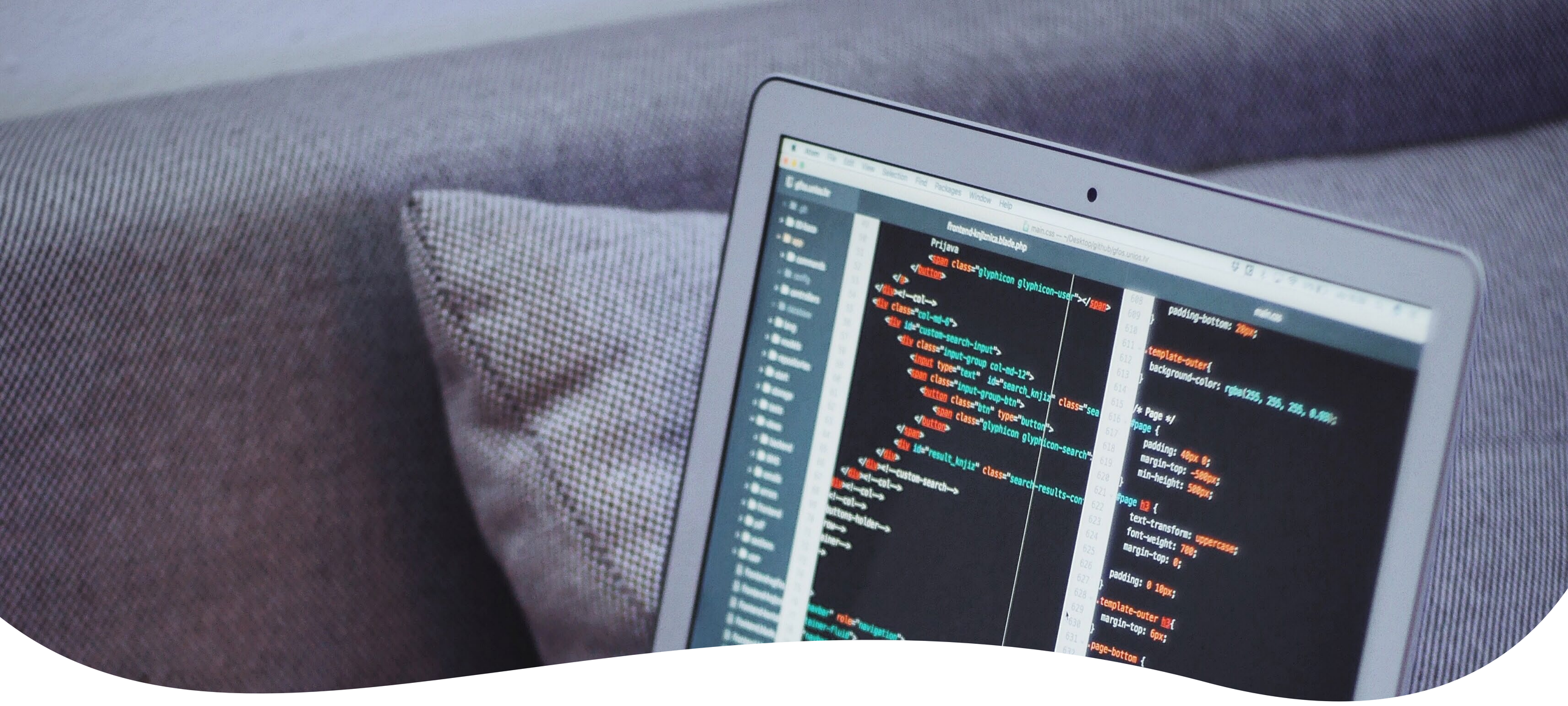
En fonction de la structure du code ou de la nature des données, il peut être intéressant de tester plusieurs fois une regex pour s’assurer d’avoir pris en compte toutes les possibilités.
Par exemple, dans le cadre de la gestion de données téléphoniques internationales : un numéro de téléphone français ne sera pas le même qu’un numéro de téléphone canadien. La regex ne sera donc pas la même, mais il est tout à fait possible de créer une regex prenant en compte tout type de numéro de téléphone.

3 - L’ordre de remplacement a son importance

Il est crucial d’anticiper l’ordre dans lequel appliquer des modifications par regex de remplacement. Comme pour l’astuce précédente, il est utile de tester la recherche avant de procéder au remplacement.

4 - Attention à la casse

Les regex sont sensibles à la casse. Ainsi, le résultat obtenu correspondra parfaitement à la regex recherchée, minuscules et majuscules comprises.



La casse des caractères est donc fondamentale lors de la rédaction d'une expression régulière. Il est cependant possible d'ajouter /i à la regex, ce qui permet d'ignorer la casse.

5 - Anticiper les exceptions

En cherchant un élément précis à supprimer, par exemple les doublons \b(\w+)\b(?=.*\1), il peut être intéressant de prendre en compte toutes les éventualités. Par exemple, « vous vous souvenez quand nous nous retrouvions... » sont des doublons intentionnels à conserver. Dans ce cas-là, il faudra sûrement parcourir chaque doublon ou alors intégrer dans la recherche des mots à traiter comme des exceptions, à l'aide de la regex négative (?!vous).

6 - Clarté et concision

[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9] [0-9]

[0-9]{10}

\d{10}

Ces trois regex donnent le même résultat : elles sont toutes fonctionnelles, mais la plus courte est préférable car elle est plus lisible.

7 - Copier les regex

Il est utile de garder à portée de main les regex les plus souvent utilisées, simplement dans un bloc-notes, ou à la suite du mémento HubSpot sur les expressions régulières les plus utiles.

Chapitre 5

Les regex ne fonctionnent pas partout

Les regex ne fonctionnent pas partout

La notion de saveur (flavor en anglais)

Il existe différentes saveurs, ou variantes de syntaxe, pour les expressions régulières. Selon les normes POSIX, deux formes sont possibles : les expressions régulières de base ou étendues. Et chaque forme possède ses propres spécificités.
C’est pourquoi, une fois maîtrisées, les regex sont facilement applicables à la majorité des **langages de programmation**, à quelques exceptions près.
Outre les éléments de base, chaque moteur de regex prend en charge un type de syntaxe différent.

Les caractères génériques (wildcards en anglais)

Il ne faut pas confondre les regex et les caractères génériques.
Une expression régulière est un modèle utilisé pour représenter une chaîne de caractères.
Un caractère générique est un composant, utilisable dans une expression régulière, qui représente tout caractère, un peu comme un joker.
En termes informatiques, ce joker peut simplement être un astérisque * qui peut correspondre à un ou plusieurs caractères, ou un point d’interrogation ? correspondant à n’importe quel caractère unique.
Même si les regex peuvent inclure des caractères génériques, elles restent un comparateur de modèles beaucoup plus puissant : elles donnent la possibilité de restreindre le type de caractères, mais aussi de rester flexible sur le nombre de caractères.

Conclusion

Les nombreux exemples cités plus haut montrent que le champ d'application d'une expression régulière est vaste. La maîtrise de cette compétence est bénéfique dans diverses professions : développeur web, analyste de données ou blogueur occasionnel.

Voici quelques exemples de cas d'utilisation des regex :

- **Validation des données**
- **Extraction de données (Data Scraping en anglais)**
- **Analyse de chaînes de caractères**
- **Remplacement de chaînes de caractères**
- **Modification syntaxique**
- **Conversion des données brutes *(Data Wrangling* en anglais)**

Une connaissance approfondie des regex n'est pas indispensable pour en retirer des bénéfices : même une maîtrise des bases peut améliorer grandement la productivité dans de nombreux domaines d'activité.

Mémento sur les expressions régulières

Rechercher une informations dans du code grâce à une regex

Info à chercher	Regex
Date	(0[1–9]|[12][0–9]|3[01])[- /.](0[1–9]|1[012])[- /.](19|20)\d\d
Heure (format 24H)	(0[0–9]|1[0–9]|2[0–3])[:Hh]([0–5][0–9])
Adresse email	[\w-\.]+@([\w-]+\.)+[\w-]{2,4}
Numéro de téléphone français	(?:(?:\+|00)33|0)\s*[1-9](?:[\s.-]*\d{2}){4}
Code Postal	\d{2}[ ]?\d{3}
URL	https?:\/\/(www\.)?[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z]{2,6}\b([-a-zA-Z0-9@:%_\+.~#()?&//=]*)
URL protocole optionnel	(https?:\/\/)?(www\.)?[-a-zA-Z0-9@:%._\+~#=]{2,256}\.[a-z] {2,6}\b([-a-zA-Z0-9@:%_\+.~#?&//=]*)
Code couleur - hexadécimal	^#?([a-fA-F0-9]{6}|[a-fA-F0-9]{3})$
Slug	^[a-z0-9-]+$
Chemin d'accès aux fichiers	(\/)?[a-z0-9 _@\-^!#$%&+={}.\/\\\[\]]+(\.[a-z]+)?$
Espace blanc/vide	^\s*$
Balises titres <h1> à <h6>	<h([1-6].*?)>(.*?)<\/h([1-6])>
Balises HTML	<[^>]*>

Info à chercher	Regex
Date de création d'un article	“date\.created” content=”\d{4}(-\d{2}){2}”
Date de modification (MàJ) d'un article	“date\.updated” content=”\d{4}(-\d{2}){2}”
Date de MàJ d'un article différente de 2023	“date\.updated” content=”(?!2023)\d{4}(-\d{2}){2}”
Date de modification (MàJ) du code HTML	“date\.modified” content=”\d{4}(-\d{2}){2}”
Recherche des images hébergées sur mon site exemple.com	(http(s?):\/\/)?(([^.]+)\.)?exemple\.com(\/)?(.*?)(?:jpg|gif|png)

Téléphone

Info à chercher	Regex
10 chiffres	\d{10}
10 chiffres commençant par 0	0\d{9}
5 couples de 2 chiffres avec espaces entre couples	(\d{2}\s?){5}
5 couples de 2 chiffres avec possible espaces, tirets, ou slash entre couples	(\d{2}[-\/\s]?){5}
5 couples de 2 chiffres avec possible espaces, tirets, ou slash entre couples, et commençant par 0	0\d (\d{2}[-\/\s]?){4}
Numéro gratuit français	080[0-5](\d{6})
Numéro gratuit français avec espace	08\s?0[0-5]\s?(\d{2}\s?){3}

Find & Replace :

Trouver l’information dans notre code grâce à une Regex et la remplacer

Objectif	Regex à chercher	Description	Regex pour remplacer	Description
Remplacer mes titres de niveau 1 par des titres de niveau 2	<h1.*?>(.*?)<\/h1>	Recherche de toutes les balises <h1>, y compris leur contenu	$2	Remplacement de la balise <h1> uniquement, sans modifier son contenu.
Modifier le texte de mes sous-titres	(<h2.*?>).*?(</h2>)	Recherche de toutes les balises , contenu compris	$1 Mon nouveau titre $2	Maintient du même niveau de titre h2, mais modification du contenu.
Chercher les images sans texte altérnatif	(<img(?!.*?alt=([“”]).*?\2)[^>]*)(>)	Recherche des balises images qui ne possèdent pas de texte altérnatif (alt=””)	$1 alt=”Photo de Chien”$3	Ajout des textes altérnatifs dans les balises images. Le texte à ajouter se place entre les guillemets
Supprimer les attributs de style	style=”([^”]*)”	Recherche des attributs style		Supprime tous les attributs styles du code
Modifier le style de mes balises	(style=”)([^”]*)(“)	Recherche des attributs style	$1 color: red ; $3	Modifie le contenu de l’attribut style, pour mettre la couleur de police en rouge
Centrer une image	(<img)([\w\W]+?)(>)	Recherche des balises images	$1 $2 style=”display: block; margin : 0 auto;” $3	Ajout d’un attribut style avec propriété de centrage au balise image
Chercher des lignes doublons succéssives	^(.*)(\r?\n\1)+$	Recherche des doublons dans le code ; ligne similaire les unes après les aautres.	$1	On ne garde que la première occurence de la chaîne de caractères répétées.
Modifier les couleurs de texte	#?([a-fA-F0-9]{6}|[a-fA-F0-9]{3})	Rechercher les styles de couleurs hexadecimale	#FF0000	Remplacement par la couleur blanche
Ajouter des couleurs au titre de niveau 2	(<h2)([\w\W]+?)(>)	Recher de toutes les balises <h2>	$1 style=”color : #FF0000;” $2 $3	Ajout de l’attribut style=”” aux balises <h2>
Modifier la couleur de police de mes h3 par #ff7a59	(<h3 (.*?) style=”\s?color\s?:)\s?#.*;(“(.*?)>(.*?))<\/h3>	Recherche des attributs style=”color:” dans les balises <h3> avec des valeurs différentes de la couleur #ff7a59	$1 #ff7a39 $3	Remplacement des couleurs de police dans les <h3> par la couleur #ff7a39
Modifier une adresse mail	[\w -\.]+(@)[\w-]+\.+([\w-]{2,4})	Recherche de toute chaîne de caractères correspondant à une adresse email	email$1email.$2	Remplacement de valeur dans l’adresse email tout en gardant @ et la terminaison (.com/.fr/. live, etc)
Modifier le sigle @ dans une adresse email	([\w -\.]+)@([\w-]+\.+([\w-]{2,4}))	Recherche de toute chaîne de caractères correspondant à une adresse email	$1at$2	Remplacement de @ par “at”.
Rechercher tous les h1 contenant une date qui n’est pas l’année actuelle et la modifier par l’année 2023 par exemplle.	(<h1.*?>)(.*?)(?!2023)\d{4}(.*?)(<\/h1>)	Recherche de tous les h1 contenant une date AAAA qui n’est pas 2023.	$1$2 2023 $3$4	Remplacement de la date trouvée par 2023.

Find & Replace :

Trouver l’information dans notre code grâce à une Regex et la remplacer (suite)

Objectif	Regex à chercher	Description	Regex pour remplacer	Description
Modifier la date de publication, mise à jour, d’un article	(datetime=”).*?(“)	Recherche de l’attribut datetime servant à mettre un timestamp sur la publication d’un article. Unique dans le code et n’est pas toujours présent.	$1 2022-12-24 $2	Modification de la date de publication au 24 décembre 2022. Attention au format datetime Année-Mois-Jour (AAAA-MM-JJ)
Trouver des articles dont l’année de publication n’est pas l’année actuelle	(datetime=”)\d{4}(?!2022)(-\d{2}-\d{2}”)	Recherche de date dans l’attribut datetime qui n’est pas l’année 2022	$12022$2	Remplacement de l’année uniquement.
Remplacer les espaces simple par devant la ponctuation double, le point d’intérrogation, et le point d’exclamation.	\s([:!?])	Recherche de tous les double-points, points d’intérrogation, point d’exclamation, avec un espace simple devant.	$1	Remplacement de l’espace blanc (pas toujours pris en compte dans certains formulaire, mail, etc) par $nbsp; , équivalent HTML de l’espace blanc (pris en compte partout).
Trouver tout type de balises entre les h2 et les supprimer	(<h2.*?>)<[^>]*>(.*?)<[^>]*>(<\/h2>)	Recherche de tous les h2 contenant UNE balise suivant le h2 et entourant le contenu textuel (...	$1 $2 $3	Suppression des balises contenues dans les h2
Remplacer les liens do-follow en no follow	(<a\s*(?!.*\brel=)[^>]*)(href=”[^”]+)”([^>]*)>	Recherche tous les liens n’ayant pas l’attribut rel=””	$1$2$3” rel=”nofollow”>	Ajout de l’attribut rel=”nofollow”
Récupération des titres avec numérotation	(<h([1-6].*?)>(\d\s?))[-\/\.](.*?)<\/h([1-6])>)	Recherche des titres des différents niveaux avec numérotation suivi d’un séparateur (1 - ou 1., etc.)	$1.$4	Remplacement du caractère de séparation par le caractère voulu, ici un point.

Recherche sur liens

Objectif	Regex à chercher	Description
Chercher tous les liens liés à mon domaine exemple.com	(http(s?):\/\/)?(([^.]+)\.)?exemple\.com	Recherche de tous les liens ayant pour domaine exemple.com, le groupe précédent le domaine étant le sous-domaine
Chercher les liens vers le domaine “exemple” qui ne contienent pas “utm_campaign=campagne”	(http(s?):\/\/)?(([^.]+)\.)?exemple\.com\/(?!\? utm_campaign=campagne)(.*?)	Récupération de tous les liens du domaine exemple.com , sous-domaine ou non, URL complexe ou non, qui ne contiennent pas utm_campaign=campagne
Extraire l’identifiant de campagne de l’URL	(?P<utm_campaign>(?<=utm_campaign=).*?(?=&|\\s|$))	Récupération du numéro ID de campagne UTM dans nos liens

HubSpot

Logiciel CMS

Le CMS gratuit de HubSpot offre toutes les fonctionnalités
pour développer un site web sécurisé
à l’image de votre marque.

Obtenir le CMS