**Commandes en JavaScript : Regex**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Symbole | Signification | Code | Exemple | Utilisation | |
| Commandes |  |  |  |  | |
| / / | définition d'une regex | var maRegex = /valeur/; |  | var maRegex = /début\/fin/; | \ pour échapper un / dans une regex |
| métacaractères | se sont les caractères utilisées dans la syntaxe des regex  si on souhaite les rechercher en tant que caractère, il faut les échapper avec \ | /valeur \/ \? / |  | ! ^ $ ( ) [ ] { } ? + \* . / \ |  [ ] - | liste des métacharactères en regex  liste des métacaractère dans les classes |
| RegExp | Objet JS dédié au regex  permet de construire une regex  avec une variable (seul avantage par rapport au type primitif) | var maRegex = new RegExp("valeur", "option");  var maRegex = new RegExp(maVariable, "option"); |  |  |  |
| Syntaxe |  |  |  |  |  |
| | | OU  définit plusieurs "mots" en 1 seule regex | var maRegex = /valeurA|valeurB/; |  |  |  |
|  |  |  |  |  |  |
| ^ | valeur devant se trouver en début de la chaine | /^valeur/ |  |  | la valeur recherchée devra se trouvée en début de chaine pour être validée |
| $ | valeur devant se trouver à la fin de la chaine | /valeur$/ |  | /tous$/.test("Bonjour à tous !"); | idem mais en fin de chaine  ATTENTION : retournera false car tous n'est pas en fin de chaine, c'est ! |
| . | n'importe quel caractère sauf un espace | /va.eur/ |  |  |  |
|  | classes de caractères |  |  |  |  |
| [ ] | classe de caractères : quand plusieurs caractères peuvent convenir indifféremment | /[abc]/ | /gr[ao]s/ => "gras" + "gros" |  |  |
| [ - ] | intervalle de caractère dans une classe | /[a-j]/  /[a-jA-J0-5]/ |  | [a-zâäàéèùêëîïôöçñ] | Remarque : a-z n'inclus pas les caractères accentués. Il faut les préciser tous |
| [^ ] | exclure des caractères ou des intervalles | /[^abc]/  /[^a-j]/ |  |  |  |
|  | quantificateurs |  |  |  |  |
| ? | le ca. peut apparaitre 0 ou 1 fois | /valeu?r/ | /raclett?e/ en cas de faute |  | ca. facultatif pouvant apparaitre au max 1 fois |
| + | le ca. peut apparaitre 1 ou plusieurs fois | /valeu+r/ |  |  | obligatoire au moins une fois |
| \* | le ca. peut apparaitre 0, 1 ou plusieurs fois | /valeu\*r/ |  |  | indifférent |
| {n}  {n,m}  {n,} | le ca. doit être répété n fois  le ca. doit être répété entre n et m fois  le ca. doit être répété au moins n fois | /valeu{n}r/  /val[eu]{n,}r/ |  | /Le [1-9][a-z]{2,3} septembre/  /Le [1-9][a-z]{2,3}[a-z]+/  appliquées sur : "Le 1er septembre" | => true  => false (espace non inclus dans [a-z] ) |
|  |  |  |  |  |  |
|  | types génériques |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| ( ) | permettent de récupérer la portion de chaine correspondante lors de l'analyse et de l'enregistrer dans les prop. de l'objet RegExp | /valeur(valeurAExtraire)valeur/ | var birth = 'Je suis né en mars';  /^Je suis né en (\S+)$/.test(birth); |  | portions de chaines sont enregistrées dans les prop. .$1, .$2, … .$9 de RegExp  donc max de 9 parenthèses par expression ! |
| (?: ) | parenthèses non capturantes  pour pouvoir se servir des parenthèse pour faire des séparations dans la regex sans qu'elle n'enregistre de portion de chaine lors d'une analyse | /valeur(?:valeur)valeur/ | Vérification du début d'une URL :  /(?:https|http|ftp|steam):\/\// |  |  |
| ( ? ) | pour effectuer une recherche non-greedy sur l'élément précédent le ?  (portion de chaine à extraire la plus courte correspondant à la condition) | /valeur(valeur?valeur)/ | extraction d'une URL dans une balise HTML qui contiendrai plusieurs paires de guillemets :  /<a href="(.+?)">/ |  |  |
|  |  |  |  |  |  |
| Options |  |  |  |  |  |
|  | les options portant sur les regex sont placées à la fin de celle-ci | var maRegex = /valeur/option(s); |  |  |  |
| /i | rend l'expression insensible à la casse | var maRegex = /valeur/i; |  |  |  |
| /g | permet de poursuivre la recherche sur la totalité de la chaine sans s'arréter au premier résultat positif | var maRegex = /valeur/g; |  |  | ne fonctionne pas avec .exec( )  indispensable pour .replace( , ) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
| Méthodes de RegEx |  |  |  |  |  |
| .test( ) | recherche la regex sur laquelle elle est appliquée dans la chaine placée en paramètre | maRegex.test(maChaineATester); | => true si trouvé  => false si non |  |  |
| .exec( ) | recherche la regex sur laquelle elle est appliquée dans la chaine placée en paramètre | maRegex.exec(maChaineATester); | => tableau dont 1re case contient premier résultat  => null sinon |  |  |
| .$1  …  .$9 | prop. de RegExp stockant les portion des chaines trouvées correspondantes aux parenthèses dans la regex  récupérer l'info. | analyse par l'une des 2 méthodes ci-dessus  RegExp.$1 | var birth = 'Je suis né en mars';  /^Je suis né en (\S+)$/.exec(birth);  alert(RegExp.$1); |  |  |
| Méthodes de String | pouvant être utilisées avec des regex |  |  |  |  |
| .search( ) | retourne la position de la 1re occurrence trouvée dans la chaine  -1 si aucune occurrence n'a été trouvée | maChaine.search(maRegex); |  |  |  |
| .match( ) | retourne un tableau contenant toutes les occurrences trouvées | maChaine.match(maRegex); |  |  |  |
| .split( ) | peut aussi utiliser une regex pour couper une chaine  elle retourne un tableau contenant tout les morceaux de la chaine | maChaine.split(maRegex); |  |  |  |
| .replace( g, ) | recherche et remplace dans une chaine  param1 : chaine ou regex à chercher  param2 : chaine à substituer | maLongueChaine.replace(maRegex, maChaineDeRemplacement); |  | var date = "05/26/2011";  date = date.replace(/^(\d{2})\/(\d{2})\/(\d{4})$/, "Le $2/$1/$3");  alert(date); // Le 26/05/2011 | ATTENTION : utiliser l'option /g sur la regex pour ne pas arrêter l'analyse au 1er résultat positif  convertir une date américaine en date française |
|  | idem avec fonction permettant un traitement des données et/ou un remplacement conditionnel | maChaine.replace(maRegex, maFonction);  function maFonction (str, p1, p2, …, offset, s) {  instruction(s);  } |  | maChaine.replace(maRegex, function (str, p1, …, offset, s) {  instruction(s);  }); | avec fonction anonyme   * str portion de texte trouvé par la regex * p1, … portions capturées par les ( ) * offset position de la portion de texte trouvé (facultatif) * s totalité de la chaine (facultatif) |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |