

# Servicios Telemáticos

---

*Expresiones Regulares (RegExp o RegEx)*

# OBJETIVOS

## *Expresiones Regulares (RegExp o RegEx)*

En este tema:

- Se estudiarán las Expresiones Regulares de forma genérica
- Posteriormente se particularizará para el lenguaje JavaScript

Bibliografía:

- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular\\_Expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions)
- <http://www.regular-expressions.info/>

# INTRODUCCIÓN

- Útiles en búsquedas de cadenas/valores que siguen **patrones**
- *(De otra forma...)* Útiles para extraer información dinámica a lo largo del tiempo que guarda el mismo formato (Patrón)
  - *Ejemplos:*
    - Una dirección de correo o una URL
    - Ficheros de log
    - Ver código HTML de <http://www.loteriasyapuestas.es/primitiva/>
    - Ver código HTML de <http://www.ecobolsa.com/>
- Las expresiones regulares se basan en utilizar operadores unión, concatenación y [clausura de Kleene](#).
- *Ejemplo de patrón en C: la función strstr (cadena1, cadena2) que busca cadena2 en cadena1 y retorna un puntero a su posición de cadena1. Cadena2 es el patrón.*

# CONSTRUCCIÓN DE REGEX (I)

- Repetición de caracteres / Cuantificadores:
  - Un cuantificador tras un carácter especifica la frecuencia con la que éste puede ocurrir.
  - Los cuantificadores más comunes son
    - “+” - El signo más indica que el carácter que le precede debe aparecer al menos una vez.
      - Ejemplo: El Patrón "ho+la" describe el conjunto infinito *hola, hoola, hooola, hooooa*, etcétera.
    - “?” - El signo de interrogación indica que el carácter que le precede puede aparecer como mucho una vez.
      - Ejemplo: El patrón "ob?scuro" se corresponde con *oscuro* y *obscuro*.
    - “\*” - El asterisco indica que el carácter que le precede puede aparecer cero, una, o más veces.
      - Ejemplo: El patrón "0\*42" se corresponde con *42, 042, 0042, 00042*, etcétera.
    - “{ }” – Las llaves determinan el número de veces que se puede repetir el carácter anterior
      - Ejemplo: El patrón “pol{1,2}o” se corresponde con *polo* o *pollo*

# CONSTRUCCIÓN DE REGEX (II)

- Opciones alternativas: Operador “|” (barra vertical)
  - Separa las alternativas.
    - Ejemplo: El patrón “Blanco | Negro” solo se tendrá una de los dos colores
- Agrupación: Operador paréntesis “( )”
  - Para definir el ámbito y precedencia de los demás operadores.
    - Ejemplo
      - El patrón “(p | m)adre” es lo mismo que “padre | madre”
      - El patrón “(des)?amor” se corresponde con *amor* y con *desamor*.
- Conjuntos: Operador corchetes “[ ]”
  - Sirven para determinar una clase/conjunto de caracteres
    - Ejemplo:
      - El patrón “[aeiou]” define un carácter que sea una vocal
      - El patrón “[1-9]” define un carácter que sea un dígito entre 1 y 9

# CARACTERES ESPECIALES (I)

- “.” — Representa cualquier carácter
- “\$” - Representa el final de la cadena de caracteres o el final de la línea, si se utiliza el modo multilinea.
  - No representa un carácter en especial sino una posición.
    - Ejemplo: Si se utiliza la expresión regular “\.\$” se buscaran los lugares donde un punto finalice la línea.
- “^” (acento circunflejo) Este carácter tiene una doble funcionalidad:
  - Utilizado individualmente el carácter “^” representa el inicio de la cadena (de la misma forma que el signo de dólar “\$” representa el final de la cadena).
  - Utilizado en conjunto con los corchetes de la siguiente forma “[^\\d ]” permite encontrar cualquier carácter que **NO** se encuentre dentro del grupo indicado
    - Ejemplo: El patrón “^\\d\\d/\\d\\d/\\d\\d\\d\\d\$” permite validar una fecha en formato corto, aunque no permite verificar si es una fecha válida.

# CARACTERES ESPECIALES (II)

- `\t` - Representa un tabulador.
- `\r` - Representa el "retorno de carro" o "regreso al inicio de línea"
- `\n` - Representa la "nueva línea" el carácter por medio del cual una línea da inicio.
- `\a` - Representa una "campana" o "beep" que se produce al imprimir este carácter.
- `\e` - Representa la tecla "Esc" o "Escape"
- `\f` - Representa un salto de página
- `\v` - Representa un tabulador vertical
- `\x` - Se utiliza para representar caracteres ASCII o ANSI si conoce su código. Ej.: © es `"\xA9"`
- `\u` - Se utiliza para representar caracteres Unicode si se conoce su código
- `\d` - Representa un dígito del 0 al 9.
- `\D` - Representa cualquier carácter que no sea un dígito del 0 al 9.
- `\w` - Representa cualquier carácter alfanumérico.
- `\W` - Representa cualquier carácter no alfanumérico.
- `\s` - Representa un espacio en blanco.
- `\S` - Representa cualquier carácter que no sea un espacio en blanco.
- `\A` - Representa el inicio de la cadena. No un carácter sino una posición.
- `\Z` - Representa el final de la cadena. No un carácter sino una posición.
- `\b` - Marca el inicio y el final de una palabra.
- `\B` - Marca la posición entre dos caracteres alfanuméricos o dos no-alfanuméricos.

# EJEMPLOS DE PATRONES

- Etiquetas HTML o XML: “<([a-zA-Z]\w\*?)>”
- Comienzo y fin de etiquetas tipo HTML: “<([a-zA-Z]\w\*?)>.\*?</\1>”
  - Ej.: **<b>Expresiones regulares</b>** y con **<li>\t — Representa un tabulador.</li>**
- Dirección IP: “\b\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}\b” (sin verificar que valores > 255)
- Fecha “válida” en formato dd/mm/aaaa con varios separadores  
 “^(0[1-9]|[12][0-9]|3[01])[ - /.](0[1-9]|1[012])[ - /.](19|20)\d\d\$”
- Dirección de correo electrónico: “^[A-Z0-9.\_%+-]+@[A-Z0-9.-]+\.(?:[A-Z]{2}|com|org|net|edu|gov|mil|biz|info|mobi|name|aero|asia|jobs|museum)\$”
- Patrón de número de tarjetas de crédito:
 





“^(?:4[0-9]{12} 5[1-5][0-9]{14} 3[47][0-9]{13} 3(?:0[0-5] 68[0-9])[0-9]{11} 6(?:011 5[0-9]{2})[0-9]{12} (?:2131 1800 35\d{3})\d{11})\$”	# Visa
	# MasterCard
	# American Express
	# Diners Club
	# Discover
	# JCB



# REGEX CON JAVASCRIPT (I)

- En JavaScript se trabaja con la dupla “/Patrón/Modificadores/”
  - Patrón es la expresión regular
  - Modificadores son una serie de caracteres que indican varias opciones
- Modificadores:
  - `/g` permite una búsqueda global, para encontrar todas las ocurrencias. Si no se indica solo se busca la primera ocurrencia.
  - `/i` hace la búsqueda en modo insensible. No distingue mayúsculas de minúsculas.
  - `/m` permite la búsqueda en modo multilinea.
  - Se pueden combinar distintos modificadores a la vez

# REGEX CON JAVASCRIPT (II)

- Método `compile(RegEx, modificador)`
  - Utilizada para recompilar una expresión regular durante la ejecución de un script 
  - Nota: Puede no ser de mucha utilidad. Probar el script del ejemplo quitando esta función
- Método `exec (RegEx)`
  - Busca ocurrencias del patrón en la cadena objeto 
  - Retorna NUL si no hay ocurrencias y el/los valores que concuerden
- Método `test (RegEx)`
  - Busca si existen ocurrencias del patrón en la cadena objeto 
  - Retorna TRUE si hay ocurrencias y FALSE en caso contrario
- Método `match (RegEx)`
  - Busca ocurrencias del patrón en la cadena objeto 
  - Retorna un objeto array con las ocurrencias.
  - Se recomienda usar este método frente a `exec()`

# REGEX CON JAVASCRIPT (III)

- Método **search** (RegEx)

- Busca ocurrencias del patrón en la cadena objeto
- Retorna el índice donde se encuentra el patrón encontrado y -1 si no encuentra nada



- Método **replace** (RegEx, *string*)

- Busca ocurrencias del patrón en la cadena objeto y las reemplaza por *string*
- Retorna la cadena modificada



- Método **split** (RegEx)

- Busca ocurrencias del patrón en la cadena objeto y la fragmenta en trozos
- Retorna un objeto array con los fragmentos



# RECURSOS EN LA WEB

- [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular\\_Expressions](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Regular_Expressions)
  - <http://www.cheatography.com/davechild/cheat-sheets/regular-expressions/> (CheatSheet)
  - <https://www.debuggex.com/> (Tester de Expresiones regulares para varios lenguajes)
  - <http://regexpal.com/> (Tester de Expresiones regulares y JavaScript)
  - <http://jsregex.com/> (Tester de Expresiones regulares y JavaScript)
  - <http://rubular.com/> (Editor de expresiones regulares para Ruby)
  - <http://www.regular-expressions.info/javascript.html>
  - <http://www.javascriptkit.com/javatutors/redev.shtml>
  - ...
- 
- Documentación de referencia de JavaScript
    - <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference?redirectlocale=en-US&redirectslug=JavaScript%2FReference>