


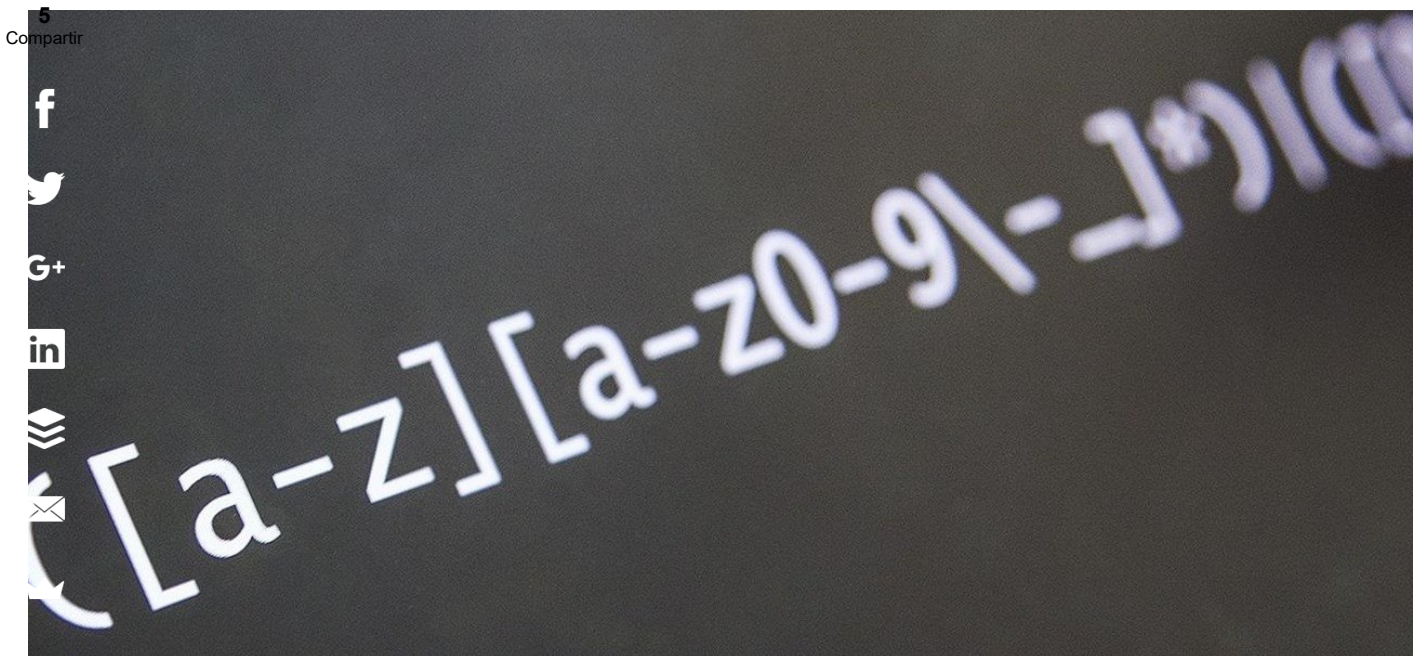


(/)

 [Conéctate \(/login\)](/login) o [Regístrate \(/registro\)](/registro)

[Inicio \(/\)](#) > [Artículos \(/articulos\)](#) > [25 expresiones regulares que todo programador web debería conocer \(/articulo/25_expresiones_regulares_que_todo_programador_web_deberia_conocer_1213\)](#)

25 expresiones regulares que todo programador web debería conocer



Las expresiones regulares son una poderosa herramienta que debe estar en el cinturón de herramientas de todos los desarrolladores. Estas expresiones tratan de coincidir contra una cadena de caracteres para extraer la información que nosotros necesitemos y así, ahorrarnos un montón de tiempo a la hora de desarrollar sitios web dinámicos.

Los desarrolladores web se enfrentan a tareas distintas comparado con los desarrolladores de software, pero muchos de los mismos fundamentos del código permanecen. Las expresiones regulares (o regex) tienen una curva de aprendizaje inicial empinada, pero pueden ser tremendamente útiles cuando se usan correctamente.

La parte más difícil de aprender es la sintaxis y sobre todo el hecho de empezar a crear tus propias expresiones regulares partiendo de cero. Para ahorrar tiempo he compilado 25 fragmentos de código regex diferentes que pueden ser incorporados en tus propios proyectos. Y como las expresiones regulares no se limitan a un solo lenguaje, puede aplicar estos snippets en cualquier lenguaje web, desde JavaScript, PHP o Python.

1. Contraseñas válidas

```
^(?=.*[A-Z].*[A-Z])(?=.*[!@#$%*])(?=.*[0-9].*[0-9])(?=.*[a-z].*[a-z].*[a-z]).{8}$
```

Código muy útil para saber si una contraseña es lo bastante segura. Con este código te ahorrarás el escribir tu propio corrector de contraseñas desde cero.

2. Color Hexadecimal

```
#([a-fA-F]|[0-9]){3, 6}
```



Ya sabéis que para establecer colores en el desarrollo web, es necesario que estén formateados en hexadecimal. Si le estamos solicitando a un usuario que ingrese un color en hexadecimal, tendremos que comprobar si lo ha hecho correctamente. Y qué mejor para ello que hacerlo mediante este código.

3. Validar dirección de email

```
/[A-Z0-9._%+-]+@[A-Z0-9-]+.+. [A-Z]{2,4}/igm
```

Una de las tareas más comunes para un desarrollador es comprobar si una cadena está formateada con el estilo de una dirección de correo electrónico. Hay muchas maneras distintas para llevar a cabo esta tarea, pero esta creemos que es la más ligera de todas las que he conocido.

4. Dirección IPv4

5b(?:(?:25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?).){3}(?:25[0-5]|2[0-4][0-9]| [01]?[0-9][0-9]?)b/
Compartir

Esta expresión regular comprobará una cadena para ver si se sigue la sintaxis de direcciones IPv4.

5. Dirección IPv6

$$(([-9a-fA-F]\{1,4\}:)\{7,7\}([-9a-fA-F]\{1,4\}|([-9a-fA-F]\{1,4\}:)\{1,7\}:|([-9a-fA-F]\{1,4\}:)\{1,6\}:[-9a-fA-F]\{1,4\})|([-9a-fA-F]$$

Esta expresión regular comprobará una cadena para ver si se sigue la sintaxis de direcciones IPv6.

b. Separador de miles

$$\cdot \{1, 3\} (? = (d\{3\}) + (? ! d)) / g$$

Los sistemas de numeración tradicionales requieren una coma, un punto, o algún otro símbolo en cada tres dígitos. Este código regex funciona con cualquier número y aplicará cualquier marca que elijas para cada tres dígitos separando entre miles, millones, etc.

7. Anteponer HTTP a enlace

```
if (!s.match(/^[\a-zA-Z]+:\/\//))
{
    s = 'http://' + s;
}
```

Independientemente del lenguaje en que trabajes (JavaScript, Ruby o PHP), esta expresión regular puede resultar muy útil. Comprobará cualquier cadena URL para ver si tiene un prefijo HTTP / HTTPS, y si no, lo antepone en consecuencia.

8. Obtener nombre de dominio

```
/https?:/(?:[-w]+.)?([-w]+).w+(?:.w+)?/?.*?/i
```

Un dominio puede contener el protocolo inicial (HTTP o HTTPS) aparte de un subdominio, más la ruta adicional de la página. Puedes utilizar este fragmento para eliminar todo eso y quedarte sólo con el nombre del dominio sin las demás florituras.

9. Ordenar palabras clave por número de palabras

```

^[^s]*$           matches exactly 1-word keyword
^[^s]*s[^s]*$    matches exactly 2-word keyword
^[^s]*s[^s]*$    matches keywords of at least 2 words (2 and more)
^([^s]*s){2}[^s]*$ matches exactly 3-word keyword
^([^s]*s){4}[^s]*$ matches 5-words-and-more keywords (longtail)

```

Los usuarios de Google Analytics y Webmaster Tools van a disfrutar con esta expresión regular. Puedes ordenar y organizar las palabras clave, basándote en el número de palabras que se utilizan en una búsqueda. Esto puede ser numéricamente específico (es decir, sólo 5 palabras) o puede coincidir con una serie de palabras (es decir, 2 o más palabras). Cuando se utiliza para ordenar los datos de análisis, se convierte en una poderosa expresión regular.

10. Encontrar una cadena Base64 en PHP

```
?php[ t]eval(base64_decode('([A-Za-z0-9+/]{4})*([A-Za-z0-9+/]{3}=[A-Za-z0-9+/]{2}==)?{1}'));
```

Si eres desarrollador de PHP, en algún momento puede que tengas que parsear el código en busca de objetos binarios codificados en Base64. Este fragmento se puede aplicar a todo el código PHP y comprueba que no existan cadenas Base64.

11. Quitar espacios

```
^[ s]+|[ s]+$
```

Un método muy útil de formatear los inputs para guardar en base de datos, hacer consultas o insertarlos dentro de un documento.

5
Compartir

12. Extraer ruta de la imagen

```
\s*\[img][^>]*[src] *= *["']{0,1}(<[^" ' >]*)
```

Si por alguna razón necesitas extraer el src de una imagen directamente desde HTML, este fragmento de código es la solución perfecta.

13. Validar fecha en formato dd/mm/YYYY

```
^/?:(?:31(/|-|\.)?(?:0?[13578]|1[02]))1|(?:(?:29|30)(/|-|\.)?(?:0?[1,3-9]|1[0-2])2))(?:?(?:1[6-9]||[2-9]d)?d{2})$|^?(?:29(/|-|\.
```

Las fechas son datos difíciles, ya que pueden aparecer como texto + números, o simplemente como números con diferentes formatos. PHP tiene una función de fecha fantástica, pero no siempre es la mejor opción. Considera utilizar esta expresión regular desarrollada para esta sintaxis de fecha específica.

14. Extraer ID de vídeo de YouTube

```
/http://(?:youtu.be/|(?:[a-z]{2,3})?youtube.com/watch(?:?|#!)v=)([w-]{11}).*/gi
```

YouTube ha mantenido la misma estructura de URL durante años porque simplemente funciona. Es también el sitio más popular para compartir videos en la web, por lo que los vídeos de YouTube tienden a conducir más tráfico. Si necesita extraer el ID de un vídeo de YouTube desde una URL, este código regex es perfecto y debería funcionar perfectamente para todas las variantes de estructuras URL de YouTube.

15. Validar ISBN

```
/b(?:ISBN(?:\s?| ))?((?:97[89])?d{9}[dx])b/i
```

Los libros siguen un sistema numérico conocido como ISBN. A través de este regex puedes validar si un input de un usuario es válido como ISBN o no.

16. Comprobar Código Postal

```
^d{5}(?:[-s]d{4})?$
```

Pues creo que no hay nada más que explicar. Esta expresión regular comprueba si una cadena puede ser considerada como un código postal de USA.

17. Validar nombre de usuario de Twitter

```
/@([A-Za-z0-9_]{1,15})/
```

Imaginemos que solicitamos a través de un formulario a un usuario que nos ingrese su nombre de usuario en Twitter. Si queremos comprobar el dato dado es correcto como nombre de usuario en Twitter, podemos utilizar esta expresión regular.

18. Encontrar atributos CSS

```
^s*[a-zA-Z-]+s*[:]{1}s[a-zA-Z0-9s.#]+[:]{1}
```

Es raro ejecutar expresiones regulares sobre CSS, pero tampoco es una situación muy extraña. Este fragmento de código se puede utilizar para extraer todas las propiedades y valores CSS de selectores individuales. Se puede utilizar para un sinfín de razones, posiblemente para ver fragmentos de CSS o eliminar propiedades duplicadas, por ejemplo.

19. Comprobar tarjeta de crédito

```
^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13}|3(?:0[0-5]||[68][0-9])[0-9]{11}|
```

La validación de un número de tarjeta de crédito, a menudo requiere de una plataforma segura alojada en otros servidores. Pero las expresiones regulares también se pueden utilizar para validar los requisitos mínimos de un número típico de tarjeta de crédito.

20. Url de perfil de Facebook

```
/^?:http://)?(?:www.)?facebook.com/(?:(:w)*#!/)?(?:pages/)?(?:[w-]*/*)([w-]*)/
```

Facebook es muy popular y ha pasado por muchos esquemas de URL diferentes. Este fragmento comprueba si una URL de usuario dada es correcta o no, en el momento actual que estamos, claro...

21. Comprobar la versión de Internet Explorer

```
^.*MSIE [5-8](?:.[0-9]+)?(?:!.*Trident/[5-9].0).*$
```

Este regex puede utilizarse en JavaScript para comprobar qué versión de Internet Explorer (5-11) está siendo utilizado.

22. Extraer precio

```
/($[0-9,]+(?:.[0-9]{2})?)/
```

Los precios vienen en una variedad de formatos que pueden contener decimales, comas y símbolos de moneda. Esta expresión regular puede comprobar todos estos diferentes formatos para sacar el precio de cualquier cadena.

23. Parsear cabeceras de email

```
/b[A-Z0-9._%+-]+@(?:[A-Z0-9-]+.[A-Z]{2,6})b/i
```

Con esta sola línea de código puedes analizar a través de un encabezado de correo electrónico el campo "a:" de la información de la cabecera.

24. Encontrar una extensión específica

```
/^(.*(?:!(htm|html|class|js)$))?(^).*$/i
```

Cuando trabajas con diferentes formatos de archivo como .xml, .html y .js, puedes comprobar los archivos tanto a nivel local como los subidos por los usuarios. Este fragmento extrae la extensión de un archivo para comprobar si es válida a partir de una serie de extensiones válidas que puedes cambiar según sea necesario.

25. Añadir rel="nofollow" a enlaces

```
(<as*(!.*brel=)[^>]*)(href="https?:/)((?!(:?(?:www.)?).implode('|(?:www.)?', $follow_list).'))[^\"]+)((?!.*brel=)[^>]*
```

Este regex puede comprobar todos los enlaces de un bloque de HTML y añadir el atributo rel = "nofollow" a cada elemento.

[COMPARTE ESTE ARTÍCULO](#)[✉ ENVIAR A UN AMIGO](#)[f COMPARTIR EN FACEBOOK](#)[🐦 COMPARTIR EN TWITTER](#)[g+ COMPARTIR EN GOOGLE +](#)[ARTÍCULO ANTERIOR](#)[SIGUIENTE ARTÍCULO](#)

5
Top 6 características del nuevo IDE de Visual Studio 2015
/articulo/top_6_caracteristicas_del_nuevo_ide_de_visual_studio_2015_1212

Hacer una validación del lado del cliente y del servidor con MVC Razor 4.0
/articulo/hacer_una_validacion_del_lado_del_cliente_y_del_servidor_con_mvc_r

¡SÉ EL PRIMERO EN COMENTAR!

Conéctate (/login) o Regístrate (/registro) para dejar tu comentario.

Secciones

[Artículos \(/articulos\)](/articulos)
[Tutoriales y código fuente \(/codigos\)](/codigos)
[Foros \(/foros\)](/foros)
[Eventos \(/eventos\)](/eventos)
[Empleo \(/empleo\)](/empleo)

Lenguajes Destacados

[PHP \(/php\)](/php)
[Java \(/java\)](/java)
[ASP \(/asp\)](/asp)
[Bases de datos \(/bases-de-datos\)](/bases-de-datos)
[C \(/c\)](/c)

Información

[Datos Legales \(/datos_legales\)](/datos_legales)
[Política de privacidad \(/politica_de_privacidad\)](/politica_de_privacidad)
[Publicidad \(/publicidad\)](/publicidad)

Contacto

[Contacte con nosotros \(/contacto\)](/contacto)
[Publicidad \(/publicidad\)](/publicidad)

<https://www.facebook.com/programacionencastellano><https://twitter.com/noprog>

Diseño web y desarrollo web (<https://colorvivo.com>). Un proyecto de los hermanos Carrero (<https://carrero.es>).

Alojado en cloud privado [Stackscale \(https://www.stackscale.com/es/\)](https://www.stackscale.com/es/)

Más internet: [Password \(https://password.es\)](https://password.es) | [Crear tu Favicon \(https://getfavicon.com\)](https://getfavicon.com) | [QRcode maker \(https://face.co/qrcode/\)](https://face.co/qrcode/) | [lucha contra el virus \(https://luchacontravirus.com\)](https://luchacontravirus.com) | [decoración de interiores \(https://decoracion2.com\)](https://decoracion2.com)

