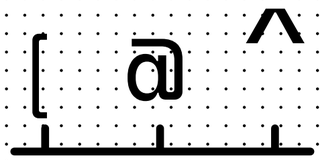
REGEX

Expresiones Regulares

```
/(\(\d{2}\)\s?)?\d{4,5}-\d{4}/g
```



Qué son las Expresiones Regulares?

- Secuencias de caracteres que forman un patrón de búsqueda.
- Puede ser un solo carácter o un patrón más complicado.

- Manipulación avanzada de cadenas, permitiendo la **búsqueda**, **validación** y **extracción** eficiente de patrones complejos.
- Compatible con la mayoría de los lenguajes de programación

Sintaxis

// cadenas compuestas por delimitadores, un patrón y modificadores opcionales.

```
$x = "/femcoders/i";
```

/ = delimitador

femcoders = patrón

i = modificador

Patrones y Coincidencias

//Los corchetes se utilizan para encontrar un rango de caracteres

[abc]	Encuentra un solo carácter de las opciones entre corchetes.
[^abc]	Encuentra cualquier carácter que NO esté entre corchetes.
[0-9]	Encuentra un solo carácter en el rango de 0 a 9.

Modificadores

//pueden cambiar la forma en que se realiza una búsqueda

- **i** Búsqueda que no distingue entre mayúsculas y minúsculas
- **m** Búsqueda multilineal (los patrones que buscan el comienzo o el final de una cadena coincidirán con el comienzo o el final de cada línea)
- **u** asegura que la expresión regular maneje la cadena como UTF-8.
- **s** Permite que el punto (.) coincida con cualquier carácter, incluyendo saltos de línea.

```php
$texto = "Hola Mundo,\nEsto es un ejemplo de una cadena grande con\n varias líneas en español. Hola universo.'

$m = '/Esto/m';  // Esto
$g = '/Hola/g';  // Hola, Hola
$i = '/mundo/i';  //Mundo
```

Metacaracteres *//caracteres con un significado especial*

|	Encuentra una coincidencia para cualquiera de los patrones separados por | como en: cat|dog|fish
.	Encuentra una única instancia de cualquier carácter
^	Encuentra una coincidencia al principio de una cadena como en: ^Hola
$	Encuentra una coincidencia al final de la cadena como en: Mundo$
\d	Encuentra un dígito
\s	Encuentra un carácter de espacio en blanco \S - NO SPACE
\b	Encuentra una coincidencia al principio de una palabra como esto: \bPALABRA, o al final de una palabra como esto: PALABRA\b
\uxxx	Encuentra el carácter Unicode especificado por el número hexadecimal **xxxx**

```
const texto = "Fem Coders Norte 2023. Factoría F5.";

const numeros = /\d+/g;                  //[ '2023', '5' ]
const letras = /[a-zA-Z\s]+/g;           //[ 'Fem Coders Norte ', ' Factoría F' ]
const letrasMinusculas = /^Fem.*/gm;     //[ 'Fem Coders Norte 2023. Factoría F5.' ]
```

Cuantificadores

//definen cantidades

n+	Coincide con cualquier cadena que contenga al menos una "n".
n*	Coincide con cualquier cadena que contenga cero o más ocurrencias de "n".
n?	Coincide con cualquier cadena que contenga cero o una ocurrencia de "n".
n{x}	Coincide con cualquier cadena que contenga una secuencia de X "n".
n{x,y}	Coincide con cualquier cadena que contenga una secuencia de X a Y "n".
n{x,}	Coincide con cualquier cadena que contenga una secuencia de al menos X "n".

// Puedes usar paréntesis ( ) para aplicar cuantificadores a patrones completos. También se pueden usar para seleccionar partes del patrón que se utilizarán como coincidencia.

```php
<?php
$str = "Apples and bananas.";
$pattern = "/ba(na){2}/i";
echo preg_match($pattern, $str); // Outputs 1
?>
```

Funciones

preg_match_all()

```php
$str = "The rain in SPAIN falls mainly on the plains.";
$pattern = "/ain/i";
echo preg_match_all($pattern, $str); // Outputs 4
```

preg_match()

```php
$str = "Visit W3Schools";
$pattern = "/w3schools/i";
echo preg_match($pattern, $str);
```

preg_replace()

```php
$str = "Visit Microsoft!";
$pattern = "/microsoft/i";
echo preg_replace($pattern, "W3Schools", $str); // Outputs "Visit W3Schools!'
```

preg_match()	Retorna 1 si el patrón fue encontrado en la cadena y 0 si no.
preg_match_all()	Retorna el número de veces que se encontró el patrón en la cadena, que también puede ser 0.
preg_replace()	Retorna una nueva cadena donde los patrones coincidentes han sido reemplazados con otra cadena.

preg_filter()	Retorna una cadena o un array con las coincidencias del patrón reemplazadas, pero solo si se encontraron coincidencias.
preg_grep()	Retorna un array compuesto únicamente por elementos del array de entrada que coincidieron con el patrón.
preg_last_error()	Retorna un código de error que indica la razón por la cual la llamada más reciente a la expresión regular falló.
preg_replace_callback()	Dada una expresión y una función de retorno de llamada, retorna una cadena donde todas las coincidencias de la expresión son reemplazadas con la subcadena retornada por la función de retorno de llamada.
preg_replace_callback_array()	Dado un array que asocia expresiones con funciones de retorno de llamada, retorna una cadena donde todas las coincidencias de cada expresión son reemplazadas con la subcadena retornada por la función de retorno de llamada.
preg_split()	Divide una cadena en un array utilizando coincidencias de una expresión regular como separadores.
preg_quote()	Escapa los caracteres que tienen un significado especial en expresiones regulares colocando una barra invertida delante de ellos.

Mejores prácticas

Simplicidad: Evitar expresiones regulares complejas cuando soluciones más simples son viables. Cadenas largas pueden degradar el rendimiento.

Funciones específicas

Especificidad!

Greedy X Lazy

.* .*?

No son adecuadas para todo

Vamos a practicar!

Desarrolla una función llamada validarCorreo para un sistema de registro de usuarios que asegure que los correos electrónicos sigan un formato adecuado. La función debe retornar verdadero si el correo es válido y falso si no lo es.

Requisitos:

- El correo electrónico debe comenzar con una secuencia de letras o números.
- Debe contener el símbolo "@" después de la secuencia inicial.
- Después de "@" debe haber un dominio válido, que consiste en letras y puede incluir uno o más subdominios separados por puntos.
- El dominio debe terminar con una secuencia de dos o tres letras (por ejemplo, ".com", ".org").

Recursos

Para probar

Regex101

RegExr

Documentación

W3Schools

PHP.net

Comunidades

Gracias!