

1. ¿Qué es una Expresión Regular (Regex)?

Una Expresión Regular es la **lista de reglas** que das para decidir quién entra y quién no.

No validan si el dato es *cierto* (no saben si tu correo existe), pero validan si el dato tiene la **forma correcta**.

Definición simple: Una secuencia de caracteres que forma un "patrón de búsqueda".

La Sintaxis Básica (El "Diccionario")

Antes de ver los ejemplos, el alumno debe conocer estos 5 símbolos básicos:

1. **^** (Caret): Indica el **inicio** de la cadena.
2. **\$** (Dólar): Indica el **final** de la cadena.
3. **\d** : Representa cualquier **dígito** (0-9).
4. **{n}** : Indica cantidad. **{8}** significa "el anterior se repite 8 veces".
5. **[]** (Corchetes): Rango u opciones. **[A-Z]** es cualquier mayúscula.

inicio de expresión regular

Flags

expresión regular

final de expresión regular

- Los slash **/** (**azul**) son los **delimitadores** de las partes de una expresión regular.
- La **definición de la expresión regular** (**verde**) es un texto con símbolos especiales que indica que textos va a incluir.
- Los **flags** (**amarillo**), son una serie de caracteres que indican como funcionará la expresión regular.

2. Ejemplos Prácticos para tu Formulario

A. Teléfono Móvil (España)

Regla: Debe empezar por 6 o 7, y tener 9 números en total.

El Patrón:

Fragmento de código

```
^[67]\d{8}$
```

Explicación:

- `^`: Empieza aquí.
- `[67]`: El primer número **debe** ser un 6 O un 7.
- `\d{8}`: Después, queremos exactamente 8 dígitos más (del 0 al 9).
- `$`: Y aquí termina (evita que alguien escriba 10 o 15 números).

B. DNI (NIF Español)

Regla: 8 números seguidos de una letra (mayúscula o minúscula).

El Patrón:

Fragmento de código

```
^\d{8}[A-Za-z]$
```

Explicación:

- `^\d{8}`: Desde el inicio, quiero 8 dígitos obligatorios.
- `[A-Za-z]`: El último carácter debe ser una letra. Aceptamos mayúsculas (A-Z) o minúsculas (a-z).
- `$`: Final de la cadena.

Nota: Regex solo valida el **formato**. No valida matemáticamente si la letra corresponde a los números (eso requiere código JavaScript con el algoritmo del módulo 23).

C. Correo Electrónico (Email estándar)

Regla: Texto + Arroba + Dominio + Punto + Extensión (de 2 a 4 letras).

El Patrón:

Fragmento de código

```
^[w-\.]+@([w-]+\.)+[w-]{2,4}$
```

Explicación:

- `^[\w- \.]+`: El nombre de usuario puede tener letras, números (`\w`), guiones o puntos.
 - `@`: Obligatorio el símbolo arroba.
 - `([\w-]+ \.)+`: El dominio (ej: *gmail.* o *alumnos.repaso.*). Permite subdominios.
 - `[\w-]{2,4}$`: La extensión final (.com, .es, .net) suele tener entre 2 y 4 letras.
-

3. ¿Cómo implementarlo? (HTML vs JS)

Se puede usar en dos sitios.

Opción 1: En HTML (Atributo `pattern`)

Es la forma más rápida. El navegador impide enviar el formulario si no cumple la regla.

HTML

```
<form>
  <label>Móvil:</label>
  <input type="tel" pattern="^[67]\d{8}$" title="Debe empezar por 6 o 7 y tener 9 dígitos"
  required>

  <label>DNI:</label>
  <input type="text" pattern="^\d{8}[A-Za-z]$" title="8 números y una letra" required>

  <button type="submit">Enviar</button>
</form>
```

Opción 2: En JavaScript (Método `.test()`)

Para tu validación avanzada (la que activa/desactiva botones).

JavaScript

```
const patronMovil = /^[67]\d{8}$/; // Fíjate que en JS va entre barras /.../
const input = document.getElementById('movil').value;

if (patronMovil.test(input)) {
  console.log("Móvil válido ");
} else {
  console.error("Formato incorrecto ");
}
```

Recurso recomendado

RegExr.com.

1. Pones el patrón arriba.
2. Pones textos de prueba abajo.
3. Se iluminan en azul los que coinciden en tiempo real.

ESCENARIOS LABORALES

Escenario 1 : El jefe te dice: *"El código de producto debe ser: 3 letras mayúsculas (Categoría), un guion, y 4 números (ID)".* Ejemplo: **PAN-0045**.

El Patrón:

Fragmento de código

```
^[A-Z]{3}-\d{4}$
```

Desglose:

- **[A-Z]{3}**: Exactamente 3 letras mayúsculas. Si ponen "pan" (minúsculas), falla.
- **-**: El guion es literal. Debe estar ahí.
- **\d{4}**: Exactamente 4 números.

El usuario español escribe **31/12/2023**, pero la base de datos MySQL/MariaDB necesita **2023-12-31**.

Escenario2 : Validar que el usuario ha introducido la fecha en el formato internacional AAAA-MM-DD.

El Patrón:

Fragmento de código

```
^\d{4}-(0[1-9]|1[0-2])-(0[1-9]|[12]\d|3[01])$
```

-

- **Desglose (Aquí subimos el nivel):**

- **\d{4}-**: El año (4 cifras y guion).
- **(0[1-9]|1[0-2])**: **El Mes**. Aquí usamos el operador **|** (O lógico).
 - Puede ser **0** seguido de **1-9** (01, 02... 09).
 - **0** (**|**) puede ser **1** seguido de **0-2** (10, 11, 12).
 - *Esto evita que alguien ponga el mes 19 o el mes 00.*
- **(0[1-9]|[12]\d|3[01])**: **El Día**.
 - Del 01 al 09.
 - O del 10 al 29 (**[12]\d**).
 - O el 30/31 (**3[01]**)

Escenario 3 : *"La contraseña debe tener al menos 8 caracteres, incluir una mayúscula y un número".*

El Concepto Teórico: Aquí el regex lineal no sirve bien porque no sabemos el orden (el número puede estar al principio o al final). Usamos **Lookaheads** (Mirar hacia adelante).

El Patrón (Para copiar y pegar):

Fragmento de código

```
^(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,}$
```

Desglose :

- `(?=.*[A-Z])`: "Oye, asegúrate de que en algún sitio (`.` `*`) hay una Mayúscula (`[A-Z]`). Si la hay, sigue leyendo".
- `(?=.*\d)`: "Asegúrate también de que en algún sitio hay un dígito".
- `[A-Za-z\d]{8,}`: Si se cumplen las dos condiciones anteriores, ahora sí, valida que sean letras o números y mínimo 8 caracteres.