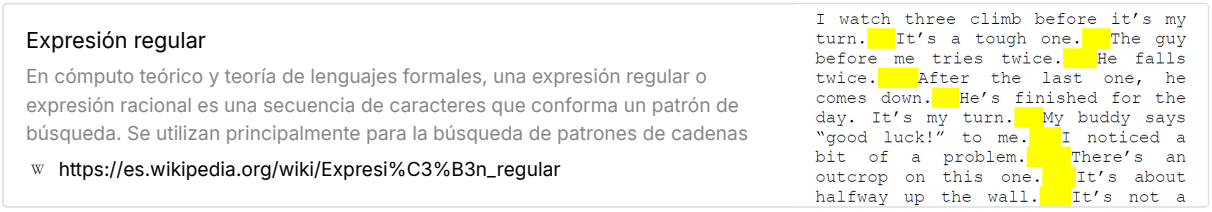
19. Expresiones regulares

Las **expresiones regulares** (RegExp) son patrones utilizados para buscar, validar o manipular texto. En JavaScript, la clase `RegExp` ofrece métodos y opciones (*flags*) para trabajar con estos patrones de manera eficiente.

---

1. La Clase `RegExp`

Recordemos brevemente la sintaxis de los patrones de las expresiones regulares:




```
// Forma 1: Notación literal
const regex = /patrón/flags;

// Forma 2: Constructor (útil para patrones dinámicos)
const regexDinamico = new RegExp("patrón", "flags");
```

---

2. Métodos de `RegExp`

a. `test()`

Verifica si un string **contiene** el patrón. Devuelve `true` o `false`.

Ejemplo:

```
const regex = /perro/i;
console.log(regex.test("Tengo un Perro")); // true
console.log(regex.test("Gato")); // false
```

b. `exec()`

Busca el patrón en un string y devuelve un **array con el resultado** (o `null` si no hay coincidencias). Incluye detalles como la posición de la coincidencia.

Ejemplo:

```
const regex = /(\w+)@(\w+)\.com/; // Sin duplicar las barras invertidas
const regex = new RegExp("(\\w+)@(\\w+)\\.com"); // Aquí SÍ se duplican las barras
const resultado = regex.exec("Correo: ana@example.com");
console.log(resultado[0]); // "ana@example.com"
console.log(resultado[1]); // "ana" (grupo 1)
console.log(resultado[2]); // "example" (grupo 2)
```

3. Métodos de String que Usan RegExp

Aunque no son métodos de `RegExp`, estos métodos de string trabajan con patrones:

a. `match()`

Similar a `exec()`, pero llamado desde el string. Devuelve un array con las coincidencias.

Ejemplo:

```
const texto = "El perro y el gato son mascotas";
const coincidencias = texto.match(/perro|gato/gi);
console.log(coincidencias); // ["perro", "gato"]
```

b. `replace()`

Reemplaza coincidencias del patrón con un nuevo valor.

Ejemplo:

```
const texto = "El animal es un PERRO";
const nuevoTexto = texto.replace(/perro/i, "gato");
console.log(nuevoTexto); // "El animal es un GATO"
```

c. `split()`

Divide un string en un array usando el patrón como separador.

Ejemplo:

```
const texto = "perro,gato;pez pájaro";
const animales = texto.split(/,|;| /);
console.log(animales); // ["perro", "gato", "pez", "pájaro"]
```

4. Flags (Banderas) de RegExp

Los **flags** modifican el comportamiento de las expresiones regulares:

Flag	Descripción
`g`	**Global**: Busca todas las coincidencias (no se detiene en la primera).
`i`	**Case-insensitive**: Ignora mayúsculas/minúsculas.
`m`	**Multiline**: Permite coincidencias en múltiples líneas (^ y $ afectan por línea).
`s`	**Dotall**: El punto (`.`) incluye saltos de línea.
`u`	**Unicode**: Permite trabajar con caracteres Unicode.
`y`	**Sticky**: Busca coincidencias consecutivas desde la última posición.

Ejemplos de Flags:

1. **Flag `g`**:

```
const regex = /mamífero/g;
const texto = "Mamífero, mamífero, MAMÍFERO";
console.log(texto.match(regex)); // ["Mamífero", "mamífero", "MAMÍFERO"]
```

2. **Flag `i`:**

```
const regex = /mamífero/i;
console.log(regex.test("MAMÍFERO")); // true
```

---

5. Ejemplos Prácticos

Ejemplo 1: Validar Nombres de Animales

```
const regexNombreValido = /^[a-záéíóúñ]+$/i;
const nombres = ["Lassie", "Garfield", "Snoopy", "123", "Tortuga Gigante"];

nombres.forEach(nombre => {
  if (regexNombreValido.test(nombre)) {
    console.log(`"${nombre}" es un nombre válido`);
  } else {
    console.log(`"${nombre}" no es válido`);
  }
});
```

Salida:

```
"Lassie" es un nombre válido
"Garfield" es un nombre válido
"Snoopy" es un nombre válido
"123" no es válido
"Tortuga Gigante" no es válido (contiene espacio)
```

Ejemplo 2: Extraer Información de un Texto

```
const texto = "Animal: Perro (4 patas), Gato (4 patas), Pájaro (2 patas)";
const regex = /(\w+)\s\((\d+)\s+patas\)/gi;

let coincidencia;
while ((coincidencia = regex.exec(texto)) !== null) {
  console.log(`Animal: ${coincidencia[1]}, Patas: ${coincidencia[2]}`);
}
```

Salida:

```
Animal: Perro, Patas: 4
Animal: Gato, Patas: 4
```

```
Animal: Pájaro, Patas: 2
```

6. Combinación de Flags

Los flags pueden combinarse para lograr comportamientos específicos:

```
// Buscar todas las coincidencias (g) sin importar mayúsculas/minúsculas (i)
const regex = /mamífero/gi;
const texto = "Mamífero, MAMÍFERO, mamífero";
console.log(texto.match(regex)); // ["Mamífero", "MAMÍFERO", "mamífero"]
```