Expresiones regulares en Javascript

Expresiones regulares

Una expresión regular es un patrón utilizado para buscar, validar o manipular texto.

Se utilizan en tareas comunes como validación de formularios, búsqueda de texto, reemplazos, etc.

Ejemplo cotidiano:

- Buscar un número de teléfono en un texto.
- Validar un correo electrónico.

Ventajas de usar RegEx:

- Potencia y flexibilidad para manejar texto.
- Reducción de código repetitivo.

Sintaxis básica:

Podemos crear una expresión regular de cualquiera de estas dos formas:

```
let regex = /patrón/; // Notación literal
let regexObj = new RegExp("patrón"); // Constructor
```

Sintaxis

Caracteres básicos

- `.`: Coincide con cualquier carácter excepto salto de línea.
- `\d`: Coincide con cualquier dígito (0-9).
- `\w`: Coincide con cualquier carácter alfanumérico.

- `\s`: Coincide con espacios en blanco.
- `\b`: Coincide con los límites de una palabra.

Modificadores

- `g`: Búsqueda global.
- `i`: Ignorar mayúsculas y minúsculas.
- `m`: Búsqueda multilínea.

Cuantificadores

- `*`: 0 o más ocurrencias.
- `+`: 1 o más ocurrencias.
- `?`: 0 o 1 ocurrencia.
- `{n}`: Exactamente `n` ocurrencias.
- `{n,}`: `n` o más ocurrencias.
- `{n,m}`: Entre `n` y `m` ocurrencias.

Grupos y rangos

- `[abc]`: Coincide con cualquier carácter dentro de los corchetes.
- `[^abc]`: Coincide con cualquier carácter **excepto** los dentro de los corchetes.
- `(abc)`: Grupo para agrupar caracteres o patrones.

Anclas

- `^`: Inicio de una línea.
- `$`: Fin de una línea.

Ejemplo práctico

Validar un número de teléfono con RegEx:

```
let regex = /^\d{3}-\d{3}-\d{4}$/;
```

```
console.log(regex.test("123-456-7890")); // true
```

Uso de Expresiones Regulares en JavaScript

Métodos principales

- `test()`: Devuelve `true` o `false` si el texto coincide con el patrón.

```
let regex = /hello/i;
console.log(regex.test("Hello world")); // true
```

- `match()`: Devuelve las coincidencias en un array.

```
let str = "The rain in Spain";
console.log(str.match(/ain/g)); // ["ain", "ain"]
```

- `replace()`: Reemplaza texto coincidente con un nuevo texto.

```
let str = "Hello, World!";
console.log(str.replace(/World/, "RegEx")); // "Hello, R
egEx!"
```

- `search()`: Encuentra la posición del primer match.

```
let str = "JavaScript is amazing";
console.log(str.search(/is/)); // 11
```

Ejemplo

Crear una RegEx para validar direcciones de correo electrónico:

```
let emailRegex = /^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\.[a-zA
-Z]{2,}$/;
console.log(emailRegex.test("test@example.com")); // tru
```

```
e
console.log(emailRegex.test("invalid-email")); // false
```

Validar contraseñas:

- Deben tener al menos 8 caracteres.
- Al menos una letra mayúscula, una minúscula y un número.

```
let passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[A-Za-z\d]{8,}$/;
console.log(passwordRegex.test("Password123")); // true
console.log(passwordRegex.test("pass123")); // false
```

Ejercicios prácticos

1. **Encuentra todas las palabras en un texto que comiencen con mayúscula**

```
let text = "Hola Mundo, Esto es JavaScript.";
console.log(text.match(/\b[A-Z][a-z]*\b/g)); // ["Hola", "Mundo", "Esto", "JavaScript"]
```

2. **Reemplaza todos los números por "[número]"**

```
let text = "Mi número es 12345 y mi código postal es 67890.";
console.log(text.replace(/\d+/g, "[número]"));
```

3. **Valida URLs básicas**

```
let urlRegex = /^(https?:\/\/)?(www\.)?[a-z0-9]+(\.[a-z]+)+$/i;
console.log(urlRegex.test("https://www.example.com"));
// true
console.log(urlRegex.test("example")); // false
```