



studio shodwe



EXPRESSIONES REGULARES

Start

OBJETIVOS

- Comprender qué son las expresiones regulares y para qué se utilizan.
- Conocer la sintaxis básica de expresiones regulares.
- Aprender a utilizar expresiones regulares en JavaScript.
- Profundizar en técnicas avanzadas para trabajar con expresiones regulares.

INTRODUCCIÓN A EXPRESIONES REGULARES

Las expresiones regulares (RegEx) son patrones que se utilizan para buscar y manipular texto. Nos permiten validar, buscar y reemplazar cadenas de manera eficiente.

Uso en JavaScript: Las expresiones regulares en JavaScript se definen entre dos barras diagonales / o usando el constructor new RegExp().

```
let regex = /hola/;  
let texto = "hola mundo";  
console.log(regex.test(texto));
```



CREACIÓN DE EXPRESIONES REGULARES



Existen dos formas de crear expresiones regulares en JavaScript:

Literal de expresión regular: let regex = /patron/;

Constructor RegExp: let regex = new RegExp("patron");

Modificadores:

Los modificadores alteran el comportamiento de la búsqueda.

- i: Insensible a mayúsculas.
- g: Búsqueda global (no se detiene después del primer match).
- m: Búsqueda multilínea.

```
let regex = /hola/i;  
console.log(regex.test("Hola Mundo")); // true
```



METACARACTERES BÁSICOS

Los metacaracteres son caracteres especiales que permiten definir patrones más complejos:

- .: Representa cualquier carácter.
- \d: Cualquier dígito (0-9).
- \w: Cualquier carácter alfanumérico (letras, números o guiones bajos).
- \s: Cualquier espacio en blanco.
- \b: Límite de palabra.

```
let regex = /\d\d/;  
console.log(regex.test("Tengo 42 años")); // true
```

Este patrón busca dos dígitos consecutivos.



CUANTIFICADORES



Los cuantificadores definen cuántas veces debe aparecer un carácter o grupo.

- *: 0 o más veces.
- +: 1 o más veces.
- ?: 0 o 1 vez.
- {n}: Exactamente n veces.
- {n,}: Al menos n veces.
- {n,m}: Entre n y m veces.

```
let regex = /ho+la/;  
console.log(regex.test("holaaaa")); // true  
console.log(regex.test("hla")); // false
```

Aquí, el patrón /ho+la/ requiere que la letra "o" aparezca al menos una vez.



AGRUPACIONES Y ALTERNANCIAS



Agrupación ():

Permite agrupar parte de la expresión para aplicar cuantificadores o capturar coincidencias.

```
let regex = /(ab)+/;  
console.log(regex.test("ababab")); // true
```

Alternancia |:

Permite definir varias opciones.

```
let regex = /perro|gato/;  
console.log(regex.test("Tengo un perro")); // true
```



ANCLAS Y FRONTERAS



Las anclas permiten definir posiciones específicas en el texto.

- ^: Inicio de línea.
- \$: Final de línea

```
let regex = /^hola/;  
console.log(regex.test("hola mundo")); // true  
console.log(regex.test("mundo hola")); // false
```



ESCAPANDO CARACTERES ESPECIALES



Algunos caracteres tienen significados especiales en RegEx. Para buscarlos literalmente, debes escaparlos con \.

```
let regex = /\. /; // Busca un punto literal
console.log(regex.test("1.2")); // true
```



EJERCICIOS PARA PRACTICAR



Validar un correo electrónico.

Validar un número de teléfono con formato (123) 456-7890



MÉTODOS EN JAVASCRIPT PARA TRABAJAR CON REGEX

- `test()`: Verifica si una cadena cumple con el patrón.
- `match()`: Devuelve las coincidencias encontradas.
- `replace()`: Reemplaza partes del texto que coinciden con la RegEx.
- `split()`: Divide una cadena usando una RegEx como delimitador.

```
let texto = "Me gusta la pizza";
let regex = /pizza/;
let nuevoTexto = texto.replace(regex, "hamburguesa");
console.log(nuevoTexto); // "Me gusta la hamburguesa"
```





studio shodwe



THANK YOU!

page 10



www.reallygreatsite.com