Ejemplos Restricciones Javascript

const palabras = texto.value.trim().split(/\s+/);

1. **texto.value**:
    - Esto accede al valor de texto, que generalmente es un elemento de entrada, como un <input> o <textarea>. Aquí texto.value extrae el texto que el usuario ha ingresado.
2. **trim()**:
    - El método trim() elimina cualquier espacio en blanco al **principio** y al **final** del texto. Esto es útil para evitar contar espacios innecesarios que podrían rodear el texto.
    - Ejemplo:

        " hola mundo ".trim(); // Devuelve "hola mundo"
3. **split(/\s+/)**:
    - Este método **divide** el texto en fragmentos usando una expresión regular como delimitador.
    - La expresión regular /\s+/ representa cualquier secuencia de uno o más caracteres de **espacio en blanco** (\s incluye espacios, tabulaciones y saltos de línea). El + permite que el patrón coincida con uno o más espacios consecutivos.
    - Esto significa que la cadena se dividirá en cada secuencia de espacio en blanco, generando un array de palabras.
    - Ejemplo:

        "hola mundo de JavaScript".split(/\s+/);
        // Devuelve ["hola", "mundo", "de", "JavaScript"]

Resultado final
La constante palabras ahora contiene un array de las palabras en texto.value, sin los espacios en blanco adicionales.
Ejemplo completo

const texto = { value: " hola mundo de JavaScript " };
const palabras = texto.value.trim().split(/\s+/);

console.log(palabras); // Salida: ["hola", "mundo", "de", "JavaScript"]

Regex, o **expresiones regulares**, es una secuencia de caracteres que forma un patrón de búsqueda.
Este patrón se utiliza para realizar coincidencias y manipulaciones en cadenas de texto.
const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

1. **^**:
    - Este símbolo indica el **inicio** de la cadena. En una expresión regular, ^ asegura que el patrón coincida desde el principio del texto.
2. **[^\s@]+**:
    - [ ] delimitan un **grupo de caracteres**. ^ dentro del grupo significa "ninguno de estos caracteres".
    - \s representa cualquier tipo de **espacio en blanco** (como espacios, tabulaciones, etc.), y @ es el carácter **arroba**.
        1. Juntos, [^\s@] significa **cualquier carácter excepto espacios en blanco y @**.
    - El + indica que debe haber al menos uno o más caracteres consecutivos que no sean espacios ni @.
    - Esta primera parte garantiza que el correo electrónico tenga un conjunto de caracteres antes de la @.
3. **@**:
    - Es el símbolo de arroba @, que es **obligatorio** en cualquier dirección de correo electrónico.
4. **[^\s@]+**:
    - Esta segunda parte después de @ es similar a la primera: requiere uno o más caracteres que no sean espacios ni @. Representa el **nombre del dominio**.
5. **\.**:
    - El carácter \. representa un **punto literal**. Es necesario escapar el . con una barra invertida (\) para que se interprete como un punto y no como un comodín en la expresión regular.
6. **[^\s@]+**:
    - Finalmente, esta tercera parte garantiza que haya caracteres después del punto, representando la **extensión del dominio** (como .com, .org, .net, etc.).
7. **$**:
    - Este símbolo indica el **final** de la cadena. La expresión regular debe coincidir con toda la cadena hasta el final para que se considere válida.

const email = "example@example.com";
console.log(emailRegex.test(email)); // Devuelve `true` si es un correo válido