***MÓDULO 4: JavaScript Avanzado: Expresiones Regulares***

1. ***INTRODUCCIÓN:***

***1. Presentación del curso***

Algunas tareas de codificación no suelen ser fáciles de llevar a cabo. Pues la generación del algoritmo que pueden solucionar el problema que queremos atacar no suele ser siempre el óptima para el procesamiento del equipo o dispositivo donde se ejecuta nuestro programa. Puede llevar a tener demasiados pasos, o en su defecto, pueden ser muy completos tanto que puede ser complicado mantener el control sobre el flujo de dicho algoritmo. Es por esto que existen las expresiones regulares, las cuales son mecanismos que nos permitirán evaluar cadenas de textos en base a patrones, es decir, haciendo uso de caracteres especiales como llaves, paréntesis, corchetes, asteriscos y algunos otros símbolos más. Podremos establecer patrones para identificar ciertos modelos o estructuras que estén definidas en el código o textos que estemos manipulando.

1. ***CONCEPTOS BÁSICOS DE LAS EXPRESIONES REGULARES***

***2.1. Establecer el entorno del trabajo para el curso***

Para comenzar a utilizar las expresiones regulares, vamos a necesitar establecer nuestro entorno de trabajo. Para lo cual, es importante conocer que herramientas vamos a utilizar.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**console.log**(“Hola”);

***2.2. Conoce y crea tu primera expresión regular***

Las expresiones regulares son un mecanismo para encontrar patrones en cadenas de textos. Este tipo de mecanismo me permite encontrar de manera muy simple los textos que yo necesito, ya que estos textos requieren de cierta sofisticación para poder encontrarlos. Esa sofisticación que un algoritmo común y corriente no permite hacerlo o, a veces, para llegar a ese resultado, necesitamos de un algoritmo muy complejo; todo esto lo resuelven las expresiones regulares.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = **new RegExp**(‘abc’);

**var** expresion2 = /abc/;

**var** parrafo = /text1/ \**/n** text2/;

**NOTAS:**

1. **var** expresion1 = **new RegExp**(‘abc’); //Es un objeto para crear expresiones regulares y recibe como parámetro la evaluación que nosotros queremos hacer. Se utiliza cuando el código es bastante extenso.

2. Las dos variables definidas posteriormente, también son formas de crear expresiones regulares. La primera, es más conveniente que la segunda dada su facilidad de lectura y se emplea en códigos no tan extensos.

***2.3. Prueba y evalúa tus primeras expresiones regulares***

Vamos a comenzar a probar nuestras expresiones regulares. Para poder evaluar las expresiones regulares, primero hay que invocar al método que me va a ayudar a probarlas. Este método se llama **test**.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = **new RegExp**(‘abc’);

**console.log**(expresion1**.test**(‘abcdef’));

**var** expresion2 = /Hola Mundo/;

**console.log**(expresion2**.test**(‘El primer programa es Hola Mundo));

**NOTAS:**

1. **console.log**(expresion1**.test**(‘abcdef’)); //Evalúa la expresión regular. Si encuentra el patrón que le paso como argumento, imprime un valor “true” en consola, caso contrario, un “false”.

2. También se puede invocar una expresión regular desde consola y evaluar la prueba.

***2.4. Evaluar un conjunto de caracteres con expresiones regulares***

Si tengo un conjunto de caracteres, esto me permite evaluar si, al menos, tengo una coincidencia, aunque no tiene que ser exacta pero debe existir alguno de esos elementos.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

//**var** expresion1 = /[0123456789]/;

**var** expresion1 = /[0-9]/;

**console.log**(expresion1**.test**(‘Sucedió en 1996’));

**NOTAS:**

1. **var** expresion1 = /[0123456789]/; //Permite evaluar si, en la expresión que deseo probar, existe algún número dentro del paréntesis.

2. **var** expresion1 = /[0-9]/; //Misma función que la anterior pero trabaja con rangos. Permite evaluar si, en la expresión que deseo probar, existe algún número del 0 al 9.

3. La evaluación siempre arroja un valor **booleano**, true o false.

***2.5. Uso de atajos para grupos de caracteres***

Las expresiones regulares cuentan con algunos atajos o *shortcuts* para agrupar caracteres.

Grupos de caracteres

===

Grupo | Descripción

------|-----

\d | Cualquier número

\w | Cualquier letra o número

\s | Cualquier espacio en blanco incluido tabs, salto de línea o similares

\D | Cualquier carácter que no sea un número

\W | Cualquier carácter que no sea alfanumérico

\S | Cualquier carácter que no sea un espacio en blanco

. | Cualquier carácter excepto nuevas líneas

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /\d\d-\d\d-\d\d\d\d/;

**console.log**(expresion1**.test**(’29-06-2007’));

**NOTA:** A través de este programa, puedo evaluar si existe una expresión como el formato que aparece en la variable “expresion1”. Si existe, el valor será true, caso contrario false.

***2.6. Coincidencias y negaciones de expresiones***

Otra manera de utilizar los atajos es cuando trabajamos con las repeticiones o un universo de estas.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript - Coincidencias:*

**var** expresion1 = /[\d]/;

**console.log**(expresion1**.test**(’29-06-2007’));

**NOTA:** El programa evalúa si existe al menos una coincidencia con el formato que le pasé.

• *JavaScript – Negación de expresiones:*

**var** expresion2 = /[^01]/;

**console.log**(expresion2**.test**(’011101000110010’));

**NOTA:** La negación de expresiones se denota a partir del símbolo ^.

***2.7. Trabajar con caracteres especiales para repetir patrones***

Con el símbolo + puedo repetir un patrón una o más veces. Con el símbolo \*, el patrón puede repetirse una o más veces pero también acepta que no esté presente.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /\d+/;

**var** expresion2 = /\d\*/;

**console.log**(expresion1**.test**(’1997’));

**console.log**(expresion2**.test**(’’));

**NOTA:** La evaluación resulta “true” para “expresión1” y “true” para “expresion2” dado que el \* admite ninguna repetición en el patrón.

***2.8. Patrones opcionales en las expresiones regulares***

Los patrones opcionales van a permitir identificar si alguna letra o algún elemento que quiero evaluar aparece o no aparece, es decir, si aparece resulta true pero si no aparece también resulta true.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /Hola mun**?**do/;

**console.log**(expresion1**.test**(’Hola mundo’));

**NOTA:** El patrón opcional viene denotado a partir del símbolo “?”. Si está presente, indica que dicho elemento puede o no estar, por lo tanto, se utiliza para elementos “pocos” importantes.

***2.9. Conteo de patrones de expresiones regulares***

Para contabilizar los patrones de las expresiones regulares, utilizaremos las llaves **{}**. Es decir, dentro de las llaves se expresan la cantidad mínima y máxima de elementos que pueden existir en la expresión regular.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /\d{1,2}-\d{1,2}\-d{4} \d{1,2}:\d{2}/;

**console.log**(expresion1**.test**(’27-06-2007 11:45’));

**NOTAS:**

1. \d{1,2}:\ //Denota puede existir 1 o hasta 2 dígitos en la expresión regular evaluada.

2. \d{1,}:\ //Denota puede existir 1 o hasta n dígitos en la expresión regular evaluada.

1. ***CONOCE Y AGRUPA LOS PATRONES SEGÚN SUS TIPOS Y FUNCIONES***

***3.1. Patrones no sensibles a mayúsculas y minúsculas***

Al estar estableciendo patrones a veces tenemos que enfrentarnos a trata con letras en mayúsculas o minúsculas, es decir, trabajar con cadenas de textos sensibles a las mayúsculas y minúsculas. Para ello existe un modificador para alterar las letras. Este operador se representa con la letra **“i”**.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /Hola Mundo/**i**;

**console.log**(expresion1**.test**(‘Hola mundo’));

***3.2. Aprende a agrupar patrones en subexpresiones***

Si tenemos un patrón que se está repitiendo constantemente, podemos simplemente definir la base de ese patrón y después anidarlo como una subexpresión.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /Woo+(hoo+)+/**i**;

//Woohoooooooo!

**console.log**(expresion1**.test**(‘Woooohooooohooooo’));

**NOTA:** /Woo+(hoo+)+/ //Es una expresión regular aninadada cuya repetición puede darse muchas veces.

***3.3. Coincidencias y grupos mediante exec y match***

El método **excec** nos brinda información en forma de arreglo de la posición donde se encontró el elemento que vamos a evaluar, input que es la cadena de texto que se evaluó y longitud del mismo a diferencia de los otros métodos que sólo arrojaban el valor de *true* o *false*.

El método **match**, por su parte, permite encontrar, a partir de una cadena de texto, el elemento que le pasemos como argumento. Nos brinda la misma información que el método anterior.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript – Método exec:*

**var** expresion1 = /\d+/**.exec**(‘Del año 1984’);

**console.log**(expresion1);

• *JavaScript – Método match:*

**console.log**(“Hasta 1994”**.match**(/\d+/));

1. ***ESTRUCTURA Y ORDENA EXPRESIONES REGULARES***

***4.1. Limitar palabras y cadenas de texto en una expresión regular***

Para forzar los límites de un patrón de expresión regular utilizaremos **\b** al incio y final de cada cadena de texto a evaluar. Esta expresión es llamada límites de cadenas de texto o de palabra.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /\bcat\b/;

**console.log**(expresion1**.test**(“cat”));

***4.2. Uso de patrones opcionales en tus expresiones regulares***

Puedo dejar determinados patrones y dependiendo de la evaluación que se vaya a hacer, elija uno u otra patrón. Estos son los patrones opcionales y se denotan a partir de la barra vertical **|**.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresion1 = /pollo**|**res /;

**console.log**(expresion1);

**console.log**(expresion1**.test**(“El cliente pidió res”));

**NOTA:** Obsérvese que el valor de *true* se puede producir a partir del uso del patrón opcional donde al elemento original “pollo” le agregamos para su evaluación otro elemento más, en este caso “res”. En cambio, si citaramos otro elemento, por ejemplo “tacos”, la consola imprimirá *false*.

***4.3. Patrones al inicio y al final de una expresión regular***

Es posible evaluar las palabras que hay al inicio o al final de una cadena de caracteres. Para esto vamos a utilizar algunos símbolos:

^ Permite definir todo lo que hay en el inicio de una cadena de texto.

$ Permite identificar todo lo hay en el final de una cadena de texto.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** expresionInicio = /**^**Any/;

**var** expresionFin = /JavaScript**$**/;

**var** expInicioFin = /**^**Any **|** JavaScript**$**/;

**console.log**(expresionInicio**.test**(“Any app that can be imagined can be made in JavaScript”));

**console.log**(expresionFin**.test**(“Any app that can be imagined can be made in JavaScript”));

**console.log**(expInicioFin**.test**(“Any app that can be imagined can be made in JavaScript”));

**NOTA:** Para ahorrar código, se puede utilizar SÓLO la variable “expInicioFin” dado que utiliza un patrón opcional donde engloba las evaluaciones al inicio y final de la cadena de caracteres.

1. ***SACA EL MAYOR PROVECHO A TUS EXPRESIONES REGULARES***

***5.1. Conoce y usa el método Replace en las cadenas de texto***

El método **replace** sirve para reemplazar ciertos bloques o ciertas cadenas o ciertos caracteres dentro de una cadena de caracteres más grande.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** mensaje = ‘El cliente pidió sushi y comió su sushi’;

**console.log**(mensaje**.replace**(/sushi/**g**, ‘tacos’));

**NOTA:** **replace**(/sushi/**g**, ‘tacos’). //El método permite reemplazar, en este caso, la palabra “sushi” por la palabra “tacos”. La letra **g** permite hacer una búsqueda y cambio de la palabra a nivel global de la cadena de caracteres y reemplazarla por “tacos”.

***5.2. Conoce y usa el método Search en las cadenas de texto***

El método **search** permite hacer búsquedas en determinado patrón en una cadena de texto.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** mensaje = ‘Esta historia continuará…;

**console.log**(mensaje**.search**(/\w/));

**console.log**(mensaje**.search**(/\s/));

**NOTAS:** 1. **search**(/\w/). //Permite buscar un carácter que no sea alfanumérico (\w).

2. **search**(/\s/). //Permite buscar cualquier carácter que no sea espacio en blanco (\s).

3. La consola, en cualquiera de los casos anteriores, imprimirá el valor de la posición de la primera incidencia que encuentre. En el caso de \s, si no encuentra incidencias, el valor que imprimirá el valor de **-1**.

***5.3. La propiedad LastIndex dentro de las expresiones regulares***

La propiedad **LastIndex** va a servir para indicar el último índice donde fue encontrada una expresión.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript – Ejemplo 1:*

**var** expresion1 = /y/**g**;

**console.log**(expresion1**.lastIndex**);

expresion1**.lastIndex** = 6;

**console.log**(expresion1**.lastIndex**);

**var** coincidencia = expresion1**.exec**(‘sibr yfx sibr yfx’);

**console.log**(coincidencia);

**console.log**(expresion1**.lastIndex**);

• *JavaScript – Ejemplo 2:*

**var** expresion2 = /\d/**g**;

**console.log**(expresion2**.exec**(‘Pedí 4 tacos y 1 agua));

**console.log**(expresion2**.exec**(‘Pedí 4 tacos y 1 agua));

expresion2**.lastIndex** = 0;

**console.log**(expresion2**.exec**(‘Pedí 4 tacos y 1 agua));

**NOTA:** expresion1**.lastIndex** = 6; //Le otorgo, forzadamente, una posición 6 del último índice.

***5.4. Ciclos y evaluaciones en las expresiones regulares***

Hacer **looping** dentro de una cadena de texto para buscar iteraciones y tener exactamente el nivel de cada una.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *JavaScript:*

**var** mensaje = “Llegaban por montones, primero 10, después 20, después 35 y se duplicaban hasta llegar a 1000”;

**var** patron = /\b(\d+)\b/g;

**var** match;

**while** (match = patron**.exec**(mensaje)) {

**console.log**(match);

**console.log**(“Se encontró “, match[1], “ en la posición: “, match**.index**);

}

1. ***CASOS DE USO MÁS COMUNES EN LAS EXPRESIONES REGULARES***

***6.1. Cómo identificar patrones para evaluar un correo electrónico***

**tools.ietf.org /html/rfc5322**

Este RFC como tal es como el conjunto de buenas prácticas de especificaciones técnicas que van a servir para detallar el funcionamiento de algo en internet, en este caso, como se va a componery funcionar un correo electrónico.

**regular-expressions.info/email.html**

Me permite ver todas las variantes que existen para poder evaluar un correo electrónico.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *HTML:*

<!DOCTYPE html>

<html lang="en">

<head>

    <title></title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="

        crossorigin="anonymous"></script>

    <!-- Latest compiled and minified CSS -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"

        crossorigin="anonymous">

    <!-- Optional theme -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"

        crossorigin="anonymous">

    <!-- Latest compiled and minified JavaScript -->

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"

        crossorigin="anonymous"></script>

    <script src="app.js"></script>

    <link rel="stylesheet" type="text/css" href="base.css">

</head>

<body>

    <form id="contactForm">

        <div class="form-group">

            <label for="email">Email:</label>

            <input type="email" class="form-control" id="email">

        </div>

        <div class="form-group">

            <label for="pwd">Password:</label>

            <input type="password" class="form-control" id="pwd">

        </div>

        <button type="submit" class="btn btn-default">Validar</button>

    </form>

</body>

</html>

• *JavaScript:*

$(function () {

    var contactForm = document.getElementById('contactForm');

    contactForm.onsubmit = function (e) {

        e.preventDefault();

        console.log("El correo es válido?",isValidEmail);

    }

    var isValidEmail = function() {

        var email = document.getElementById('email');

        return /^\w+([\.\+\-]?\w+)\*@([\.-]?\w+)\*(\.\w{2,4})+$/.test(email.value);

    }

})

• *CSS3:*

form {

    width: 300px;

    margin: 50px auto;

    background-color: #eee;

    padding: 20px;

    border-radius: 10px;

}

***6.2. Validación de usuario de Twiter mediante su cuenta***

• *HTML:*

<!DOCTYPE html>

<html lang="en">

<head>

    <title></title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="

        crossorigin="anonymous"></script>

    <!-- Latest compiled and minified CSS -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"

        crossorigin="anonymous">

    <!-- Optional theme -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"

        crossorigin="anonymous">

    <!-- Latest compiled and minified JavaScript -->

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"

        crossorigin="anonymous"></script>

    <script src="app.js"></script>

    <link rel="stylesheet" type="text/css" href="base.css">

</head>

<body>

    <form id="contactForm">

        <div class="form-group">

            <label for="twitter">Twitter:</label>

            <input type="text" class="form-control" id="twitter">

        </div>

        <div class="form-group">

            <label for="email">Email:</label>

            <input type="email" class="form-control" id="email">

        </div>

        <div class="form-group">

            <label for="pwd">Password:</label>

            <input type="password" class="form-control" id="pwd">

        </div>

        <button type="submit" class="btn btn-default">Validar</button>

    </form>

</body>

</html>

• *JavaScript:*

$(function () {

    var contactForm = document.getElementById('contactForm');

    contactForm.onsubmit = function (e) {

        e.preventDefault();

        console.log('¿El Twitter es válido?', isValidTwitter());

    }

    var isValidTwitter = function () {

        var twitter = document.getElementById('twitter');

        return /^@(\w+)$/.test(twitter.value)

    }

});

• *CSS:*

form {

    width: 300px;

    margin: 50px auto;

    background-color: #eee;

    padding: 20px;

    border-radius: 10px;

}

***6.3. Otros casos de uso de las expresiones regulares***

Algunas expresiones regulares más utilizadas se muestran a continuación.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

• *HTML:*

<!DOCTYPE html>

<html lang="en">

<head>

    <title></title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <script src="https://code.jquery.com/jquery-2.2.4.min.js" integrity="sha256-BbhdlvQf/xTY9gja0Dq3HiwQF8LaCRTXxZKRutelT44="

        crossorigin="anonymous"></script>

    <!-- Latest compiled and minified CSS -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u"

        crossorigin="anonymous">

    <!-- Optional theme -->

    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp"

        crossorigin="anonymous">

    <!-- Latest compiled and minified JavaScript -->

    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa"

        crossorigin="anonymous"></script>

    <script src="app.js"></script>

    <link rel="stylesheet" type="text/css" href="base.css">

</head>

<body>

    <form id="contactForm">

        <div class="form-group">

            <label for="twitter">Twitter:</label>

            <input type="text" class="form-control" id="twitter">

        </div>

        <div class="form-group">

            <label for="email">Email:</label>

            <input type="email" class="form-control" id="email">

        </div>

        <div class="form-group">

            <label for="pwd">Password:</label>

            <input type="password" class="form-control" id="pwd">

        </div>

        <button type="submit" class="btn btn-default">Validar</button>

    </form>

</body>

</html>

• *JavaScript:*

$(function () {

    var contactForm = document.getElementById('contactForm');

    contactForm.onsubmit = function (e) {

        e.preventDefault();

        console.log('¿La URL es válida?', isValidUrl());

    }

    /\*\*

     \* URL/Dirección web válida

     \*/

    var isValidUrl = function () {

        var url = document.getElementById('url');

        return /^(https?:\/\/)?([\da-z\.-]+)\.([a-z\.]{2,6})([\/\w \?=.-]\*)\*\/?$/.test(url.value)

    }

    /\*\*

     \* Contraseña segura

     \* Contraseñas con al menos una letra mayúscula.

     \* Contraseñas con al menos una letra minúscula.

     \* Contraseñas con al menos un número o caracter especial.

     \* Contraseñas con mínimo 8 caracteres.

     \*\*/

    var isValidPass = function () {

        var pass = document.getElementById('pass');

        return /(?=^.{8,}$)((?=.\*\d)|(?=.\*\W+))(?![.\n])(?=.\*[A-Z])(?=.\*[a-z]).\*$/.test(pass.value)

    }

    /\*\*

     \* Teléfonos en el mundo

     \* https://en.wikipedia.org/wiki/List\_of\_country\_calling\_codes

     \*/

    var isValidPhone = function () {

        var phone = document.getElementById('phone');

        return /(?=^.{8,}$)((?=.\*\d)|(?=.\*\W+))(?![.\n])(?=.\*[A-Z])(?=.\*[a-z]).\*$/.test(phone.value)

    }

    /\*\*

     \* Tarjetas de credito

     \*/

    var isValidCreditCard = function () {

        var phone = document.getElementById('phone');

        return /^((67\d{2})|(4\d{3})|(5[1-5]\d{2})|(6011))(-?\s?\d{4}){3}|(3[4,7])\ d{2}-?\s?\d{6}-?\s?\d{5}$/.test(phone.value)

    }

    /\*\*

     \* Código postal

     \*/

    var isValidPostalCode = function () {

        var phone = document.getElementById('phone');

        return /^([1-9]{2}|[0-9][1-9]|[1-9][0-9])[0-9]{3}$/.test(phone.value)

    }

    /\*\*

     \* Nombres de usuario

     \* Mínimo 6 caracteres

     \* Máximo 30 caracteres

     \*/

    var isValidUserName = function () {

        var phone = document.getElementById('phone');

        return /^[a-z\d\_]{6,30}$/i.test(phone.value)

    }

});

• *CSS3:*

form {

    width: 300px;

    margin: 50px auto;

    background-color: #eee;

    padding: 20px;

    border-radius: 10px;

}