INSTITUTO UNIVERSITARIO AERONAUTICO
FACULTAD DE INGENIERIA
Procesamiento de Strings
Ingeniería Web II – Ing. Carlos Simes

En esta clase, vamos a tratar un tema bastante sencillo, pero no por eso importante que es la manipulación de **Strings**. Como se abran dado cuenta con lo realizado hasta ahora, todo en la web es básicamente un String.
La información que se intercambia entre un cliente web y el servidor, codificada de diversas formas, es básicamente un conjunto de caracteres.
Los **String** están presentes en todas partes en la web.

Un documento HTML no deja de ser una estructura sofisticada de **Strings** que en algún momento debemos capturar y luego procesar. Por esto, es importante conocer las características y propiedades de este objeto, pues lo usaremos constantemente cuando queramos efectuar operaciones en nuestra web.

El Objeto String

El objeto **String** se usa para manipular cadenas de caracteres. En **JavaScript** todo texto encerrado entre comillas, dobles o simples, se interpreta como una cadena, así '45' no es el número cuarenta y cinco sino la cadena formada por los caracteres 4 y 5.

El objeto **String** permite realizar operaciones con cadenas como concatenar o dividir cadenas, buscar texto, extraer parte de un texto, etc.. La operación de crear una variable de este tipo se lleva a cabo como es habitual con el operador **new** o de manera implícita, pudiéndole pasar un argumento para inicializar la variable. Al usar un método o referirnos a una propiedad podemos usar el nombre de la variable o una constante de cadena así como en el ejemplo:

```
var mitexto = "Esta es una cadena";
var pos = mitexto.indexOf("una");
```

puede también escribirse en la siguiente forma:

```
var pos = "Esta es una cadena". indexOf("una");
```

Propiedades de String

length: devuelve la longitud de la cadena.

Métodos de String

a) anchor(atrcad)
Este método crea, a partir de un objeto String, una cadena conteniendo un elemento HTML ANCHOR, con el atributo NAME igual a la cadena que se le pase en **atrcad**.

```
var refer = "Referencia num. 1" ;
var ancla = refer.anchor("Refer1");
```

El valor de la variable **ancla** será:

<A NAME="Refer1">Referencia num. 1

La sintaxis de este método permite usar una constante String en lugar del nombre de un objeto String. El ejemplo anterior podría haber escrito como:

```
var ancla = "Referencia num. 1".anchor("Refer1");
```

b) Métodos de String: big()

Este método devuelve una cadena consistente en el objeto String rodeado con las etiquetas <BIG> </BIG> del lenguaje HTML. Por ejemplo:

```
var mitexto = "Este es el texto";
mitexto = mitexto.big();
```

Tras la última sentencia la variable **mitext**
contendrá <big>Este es el texto</big>
Se puede usar una constante de cadena en lugar de un nombre de variable, así el ejemplo podría haberse escrito:

```
var mitexto = "Este es el texto".big();
```

c) Métodos de String: blink()

Este método devuelve una cadena consistente en el String rodeado con las etiquetas <blink></blink> del lenguaje HTML. Por ejemplo:

```
var mitexto = "Texto para intermitente";
mitexto = mitexto.blink();
```

Tras la última sentencia la variable mi texto contendrá el valor:
<blink>Texto para intermitente</blink>

d) Métodos de String: bold()

Este método devuelve una cadena consistente en el String rodeado con las etiquetas <B> </B>, negrita, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Texto para negrita";
mitexto = mitexto.bold();
```
Tras la última sentencia la variable mi texto contendrá el valor:
```
<B>Texto para negrita</B>
```

e) Métodos de String: charAt(atrent)

Este método aplicado a una cadena devuelve el carácter que se encuentra en la posición dada por el atributo **atrent**, teniendo en cuenta que el índice del primer carácter a la izquierda de la cadena es 0 y el último es una unidad menor que longitud de la cadena.
Si el valor del atributo no es válido (igual o mayor que la longitud de la cadena o negativo) el método devuelve el valor **undefined**. Por ejemplo el siguiente código devuelve la posición del tercer carácter de la cadena **nombre**:

```
var nombre = "abcdefghij";
var car3 = nombre.charAt(2);
```

Devolverá "c", que es el tercer carácter por la izquierda (índice
igual a 2). _ **f) Métodos de String: charAt(atrent)**
Este método aplicado a una cadena devuelve el código Unicode del carácter que se encuentra en la posición dada por el atributo **atrent,** teniendo en cuenta que el índice del primer carácter a la izquierda de la cadena es 0 y el último es una unidad menor que longitud de la cadena. Si el valor del atributo no es válido (igual o mayor que la longitud de la cadena o negativo) el método devuelve el valor **NAN**. Por ejemplo el siguiente código devuelve el Unicode del tercer carácter de la cadena **nombre**:

```
var nombre = "abcdefghij";
var car3 = nombre.charAt(2);
```

Devolverá 99, que es el código de la letra 'c', el tercer carácter por la izquierda (índice igual a 2).

g) Métodos de String: concat(atrcad)

Este método aplicado a una cadena le agrega la cadena pasada como atributo, **atrcad,**
que será una variable o constante literal, cualquier otro tipo es convertido a cadena. Por ejemplo el siguiente código concatena 'Buenos ' y 'días':

```
var saludo = "Buenos ";
var hola = saludo.concat("días");
```

La variable **hola** contendrá "Buenos días", es lo mismo que si se hubiera escrito:

```
var hola = saludo + "días"
```

h) Métodos de String: fixed()

Este método devuelve una cadena consistente en el objeto String rodeado con las etiquetas <TT> </TT>, espaciado fijo o teletype, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Este es el texto";
mitexto = mitexto.fixed();
```

Tras la última sentencia la variable mitext
contendrá <TT>Este es el texto</TT>
Se puede usar una constante de cadena en lugar de un nombre de variable, así el ejemplo podría haberse escrito:

```
var mitexto = "Este es el texto".fixed();
```

i) Métodos de String: fontcolor(atrcad)

Este método crea, a partir de un objeto String, una cadena conteniendo un elemento FONT del lenguaje HTML con el atributo COLOR igual a la cadena que se le pase en **atrcad**.

```
var mitexto = "Texto en color" ;
mitexto = mitexto.fontcolor("#FFAC3E");
```

El valor de la variable **ancla** será:

```
<FONT COLOR="#FFAC3E">Texto en color</FONT>
```

La sintaxis de este método permite usar una constante String en lugar del nombre de un objeto
String. El ejemplo anterior podría haber escrito como:

```
var mitexto = "Texto en
color".fontcolor("#FFAC3E");
```

j) Métodos de String: fontsize(atrnum)

Este método crea, a partir de un objeto String, una cadena conteniendo un elemento FONT del lenguaje HTML con el atributo SIZE igual al valor entero que se le pase en **atrnum**.

```
var mitexto = "Texto de prueba" ;
mitexto = mitexto.fontsize(-1);
```

El valor de la variable **mitexto** será:

```
<FONT SIZE="-1">Texto de prueba</FONT>
```

La sintaxis de este método permite usar una constante String en lugar del nombre de un objeto
String. El ejemplo anterior podría haber escrito como:

```
var mitexto = "Texto de prueba".fontsize(-1);
```

k) Métodos de String: fromCharCode(cod1, cod2, ...)_

Este es un método global del objeto String que crea una cadena a partir de los códigos Unicode que se le pasen como parámetros. Por ejemplo:

```
var cadena = String.fromCharCode(65,66,67);
```
La variable cadena contendrá "ABC", que son los caracteres con los códigos 65, 66 y 67

l) Métodos de String: indexOf(atrcad, desde)
Este método devuelve la primera posición dentro del objeto String donde comienza la subcadena pasada como argumento en **atrcad**. Admite un segundo argumento opcional que indica desde que posición debe realizar la búsqueda, si se omite comienza a buscar por el primer carácter de la izquierda. Valores del segundo argumento negativos o mayores que la longitud de la cadena se consideran 0. Si la subcadena no se encuentra el valor devuelto es -1. Por ejemplo:

```
var cadena = "mi.correo@mail.com";
var arroba = cadena.indexOf('@');
var punto = cadena.indexOf('.',arroba);
```

Este ejemplo devuelve en **arroba** la posición 9 mientras que **punto** contiene la 14 pues la búsqueda se hizo desde la posición donde está el carácter arroba y encuentra el segundo punto. Recuerda que las posiciones en las cadenas se cuentan desde 0.

m) Métodos de String: italics()

Este método devuelve una cadena consistente en el String rodeado con las etiquetas <I> </I>, cursivas, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Texto en cursiva";
mitexto = mitexto.italics();
```

Tras la última sentencia la variable mi texto contendrá el valor:
```
<I>Texto en cursiva</I>
```

n) Métodos de String: lastIndexOf(atrcad, desde)

Este método devuelve la primera posición dentro del objeto String donde comienza la subcadena pasada como argumento en **atrcad**, pero realizando la búsqueda de derecha a izquierda. Admite un segundo argumento opcional que indica desde que posición debe realizar la búsqueda, si se omite comienza a buscar por el primer carácter de la derecha, valores negativos o mayores que la longitud de la cadena se consideran 0. Si la subcadena no se encuentra el valor devuelto es -1. Por ejemplo:

```
var cadena = "mi.correo@mail.com";
var arroba = cadena.lastIndexOf('@');
var punto = cadena.lastIndexOf('.',arroba);
```

Este ejemplo devuelve en **arroba** la posición 9 mientras que **punto** contiene la 2 pues la búsqueda se hizo desde la posición donde está el carácter arroba hacia el principio de la cadena encontrando el primer punto. Recuerda que las posiciones en las cadenas se cuentan desde 0.

o) Métodos de String: link(atrcad)
Este método crea, a partir de un objeto String, una cadena conteniendo un elemento ANCHOR del lenguaje HTML, con el atributo HREF igual a la cadena que se le pase en **atrcad**.

```
var enlace = "Dirección" ;
enlace = enlace.link("http://www.ciudadfutura.com');
```
El valor de la variable **enlace** será:
```
<A HREF="http://www.ciudadfutura.com">Dirección</a>
```

La sintaxis de este método permite usar una constante String en lugar del nombre de un objeto
String. El ejemplo anterior podría haber escrito como:
```
var enlace = "Dirección".anchor("Refer1");
```

p) Métodos de String: match(expreg)

Este es uno de los más potentes métodos para buscar subcadenas y realizar sustituciones dentro de cadenas de texto. Permite usar patrones de búsqueda construidos con comodines y texto, lo que se denominan expresiones regulares. Este método usa como argumento una expresión regular y va buscando en el objeto alguna subcadena que concuerde con esa expresión. Esta subcadena la devuelve en un **array**. Si no encuentra ninguna devuelve **null**. Además actualiza el valor de una variable global **RegExp** que

almacena en sus propiedades diversa información acerca de la búsqueda realizada. Por ejemplo:

```
var frase = new String();
frase="Busco palabras con menos de cinco letras";
var result=new Array();
result=frase.match(/\b\w{1,4}\b/g);
document.write("Hallados: '+result+'<br>');
document.write("En la frase: " + RegExp.input);
```

Si pruebas el ejemplo obtendrás el siguiente listado

```
Hallados: con,de
En la frase: Busco palabras con menos de cinco letras
```

El patrón de búsqueda está encerrado entre dos barras / , y busca caracteres alfanuméricos (\ **w**) comprendidos entre límites de palabras (\ **b**) además hace una búsqueda global (indicado por la **g** fuera de las barras).

q) Métodos de String: replace (expreg, nueva)

A vueltas con las expresiones regulares, difíciles pero potentes. Con este método todas las cadenas que concuerden con la **expreg** del primer argumento son reemplazadas por la cadena especificada en el segundo argumento, **nueva**, que puede contener elementos del patrón mediante los símbolos $1 a $9. El resultado devuelto es la cadena con las sustituciones realizadas. Por ejemplo vamos a cambiar **palabra** por **frase** en la frase "Cada palabra dicha es una palabra falsa"

```
var linea = new String();
linea="Cada palabra dicha es una
palabra falsa"; linea =
linea.replace(/palabra/g, "frase");
document.write(linea);
```

Si pruebas el ejemplo obtendrás lo siguiente:

```
Cada frase dicha es una frase falsa
```

En esta ocasión se ha usado un patrón con el modificador **g** de global por lo que cambia todas las coincidencias, si se omite sólo se cambia la primera. En la cadena nueva pueden usarse elementos del patrón, por ejemplo cambiemos las negritas a cursivas en la frase:

```
var patron = /(<b>)([^<]+)(<\/b>)/g;
var frase = "Cada <b>negrita</b> pasa a
<b>itálica</b>"; document.write(frase+"<br>");
newstr = str.replace(patron, "<i>$2</i>");
```

```
document.write(frase);
```

veras la frase antes y después del cambio:
Cada **negrita** pasa a **itálica**
Cada *negrita* pasa a *itálica*
El $2 es un índice referido a los paréntesis del patrón, así $1 indica lo contenido en el primer paréntesis (<b>) mientras que $3 es <\b>, el tercer paréntesis.

r) Métodos de String: search (expreg)
Es un método similar al método match pero más rápido. Este método realiza una búsqueda en la cadena y devuelve el índice donde se produce la primera concordancia con el patrón o -1 si no existe ninguna. Por ejemplo buscamos las cadenas 'lunes' o 'martes' en la frase cita, la letra i del patrón indica que se debe ignorar el tipo mayúsculas o minúsculas en la búsqueda:

```
var patron = /sábado|miércoles/i;
var cita = "La reunión será el lunes y el
miércoles";
document.write(cita.search(patron)+"<br
>");
```
Si pruebas el ejemplo obtendrás un 30, la posición de la palabra 'lunes'.

s) Métodos de String: slice (inicio, ultimo)
Este método devuelve la porción de cadena comprendida entre las posiciones dadas por los argumentos inicio y ultimo, o el final de la cadena si se omite este segundo argumento. Si ultimo es negativo, se interpreta como número de posiciones contadas desde el final de la cadena. Si los argumentos no son números enteros, por ejemplo cadenas, se convierten a números enteros como haría el método **Number.parseInt().**

```
var frase = "Autor: Luis Sepúlveda";
var nombre = frase.slice(7);
```

La variable **nombre** contendrá "Luis Sepúlveda". En este otro ejemplo usamos un segundo argumento:

```
var frase = "Autor: Luis Septiembre";
var nombre = frase.slice(7, -10);
```

nombre contendrá "Gonzalo"', es decir desde la posición 7 hasta 10 posiciones antes del final.

t) Métodos de String: small()

Este método devuelve una cadena consistente en el objeto String rodeado con las etiquetas <SMALL> </SMALL>, reducir tamaño, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Este es el texto";
mitexto = mitexto.small();
```

Tras la última sentencia la variable mitext
contendrá <SMALL>Este es el
texto</SMALL>

Se puede usar una constante de cadena en lugar de un nombre de variable, así el ejemplo podría haberse escrito:
var mitexto = "Este es el texto".small();

u) Métodos de String: split (separ)

Devuelve un array conteniendo las porciones en que queda separada la cadena por el separador indicado mediante el argumento **separ**, que será una expresión regular o una cadena literal. Si este separador es una cadena vacía el texto queda desglosado en todos sus caracteres. Si se omite el separador el resultado es un array de un elemento con la cadena completa.

```
var linea=new String("Título: El portero");
var lista = linea.split(/:\s*/);
```

La variable lista es un **array** con dos elementos "Título" y "El portero". También podríamos haberlo escrito como var linea=new String("Título: El portero");

```
lista = linea.split(":");
document.write(lista);
```

en este caso el primer elemento de lista es "Título" y el segundo " El portero" con un espacio por delante. Por último si el separador es una cadena vacía:

```
var linea=new String("Título: El portero");
lista = linea.split("");
document.write(lista);
```

la variable lista contendrá T,í,t,u,l,o,:, ,E,l, ,p,o,r,t,e,r,o.

v) Métodos de String: strike()

Este método devuelve una cadena consistente en el String rodeado con las etiquetas <STRIKE> </STRIKE>, tachado, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Texto para ver tachado";
mitexto = mitexto.strike();
```

Tras la última sentencia la variable mi texto contendrá el valor:
<STRIKE>Texto para ver tachado</STRIKE>
w) Métodos de String: sub()
Este método devuelve una cadena consistente en el objeto String rodeado con las etiquetas , subíndice, del lenguaje HTML. Por ejemplo:

```
var mitexto = "Este es el texto";
mitexto = mitexto.sub();
```

Tras la última sentencia la variable mitext
contendrá _{Este es el texto}
Se puede usar una constante de cadena en lugar de un nombre de variable, así el ejemplo podría haberse escrito:

```
var mitexto = "Este es el texto".sub();
```

x) Métodos de String: substr(inicio, largo)

Devuelve una subcadena extraída del objeto string comenzando por la posición dada por el primer argumento, **inicio**, y con un número de caracteres dado por el segundo argumento, **largo**. Si se omite este último argumento la subcadena extraída va desde **inicio** hasta el final de la cadena.

```
var linea=new String("Mi página es ideal);
var lista = linea.substr(3);
```

La variable **lista** contendrá "página es ideal".

```
var linea=new String("Mi página es ideal);
var lista = linea.substr(3, 6);
```

ahora la variable **lista** contendrá "página".

y) Métodos de String: substring(ind1,ind2)

Devuelve una subcadena del objeto string que comienza en la posición dada por el menor de los argumentos y finaliza en la posición dada por el otro argumento. Si se omite este último argumento la subcadena extraída va desde la posición indicada por el único argumento hasta el final de la cadena. Si los argumentos son literales se convierten a enteros como un **parseInt()**.

```
var linea=new String("Mi página es ideal);
var lista = linea.substr(3);
```

La variable lista contendrá "página es ideal".

```
var linea=new String("Mi página es ideal);
var lista = linea.substr(3, 9);
```

ahora la variable lista contendrá "página", al
igual que en

```
var linea=new String("Mi página es ideal);
var lista = linea.substr(9, 3);
```

z) Métodos de String: sup()

Este método devuelve una cadena consistente en el objeto String
rodeado con las etiquetas , superíndice, del
lenguaje HTML. Por ejemplo:

```
var mitexto = "Este es el texto";
mitexto = mitexto.sup();
```

Tras la última sentencia la variable **mitexto**
contendrá <big>Este es el texto</big>
Se puede usar una constante de cadena en lugar de un nombre de variable, así el ejemplo podría haberse escrito:

```
var mitexto = "Este es el texto".sup();
```

aa) Métodos de String: toLowerCase()

Devuelve una cadena igual a la original pero con todos los caracteres en minúsculas. No afecta como es lógico a caracteres no alfabéticos, o sea, a los números, letras acentuadas y caracteres especiales como la Ñ

```
var linea=new String("´Hoy es Domingo");
linea = linea.toLowerCasesubstr();
```

La variable lista contendrá "hoy es domingo".

bb) Métodos de String: toUpperCase()

Devuelve una cadena igual a la original pero con todos los caracteres en mayúsculas. No afecta como es lógico a caracteres no alfabéticos, o sea, a los números, letras acentuadas y caracteres especiales como la Ñ. Es muy útil a la hora de comparar cadenas para asegurarse que dos cadenas no difieran sólo por que algún carácter esté en mayúscula o minúscula.

```
var linea=new String("´Hoy es Domingo");
linea = linea.toUpperCase();
```

La variable lista contendrá "HOY ES DOMINGO".

Resumen de Funciones de String y su aplicación

Método	Descripción
charAt()	Devuelve el carácter ubicado en la posición pasada como parámetro
charCodeAt()	Devuelve el UNICODE del carácter ubicado en la posición pasada como parámetro
concat()	Une dos o más String y devuelve el resultado de la unión.
endsWith()	Controla si la String finaliza con el carácter pasado como parámetro.
fromCharCode()	Convierte en carácter el código UNICODE
includes()	Controla si la String contiene el carácter / String pasada como parámetro
indexOf()	Regresa la primera ocurrencia dentro de la String del carácter pasado como parámetro
lastIndexOf()	Devuelve la posición de la última aparición encontrada de un valor especificado en una String.
localeCompare()	Compara dos Strings en la configuración regional actual
match()	Busca una String para una coincidencia con una expresión regular y devuelve las coincidencias

repeat()	Devuelve una nueva String con un número especificado de copias de una String existente
replace()	Busca una String para un valor especificado, o una expresión regular, y devuelve una nueva String donde se reemplazan los valores especificados
search()	Busca en una String un valor especificado, o expresión regular, y devuelve la posición de la coincidencia
slice()	Extrae una parte de una String y devuelve una nueva String
split()	Divide una String en una matriz de substrings
startsWith()	Comprueba si una String comienza con caracteres especificados
substr()	Extrae los caracteres de una String, comenzando en una posición inicial especificada, y hasta la cantidad especificada de caracteres.
substring()	Extrae los caracteres de una String, entre dos índices especificados
toLocaleLowerCase()	Convierte una String en letras minúsculas, según la configuración regional del host
toLocaleUpperCase()	Convierte una String en letras mayúsculas, según la configuración regional del host

toLowerCase()	Convierte una String en letras minúsculas
toString()	Devuelve el valor de un objeto en String
toUpperCase()	Convierte una String en letras mayúsculas
trim()	Elimina el espacio en blanco de ambos extremos de una String
valueOf()	Devuelve el valor primitivo de un objeto String.

ACTIVIDADES PRÁCTICAS:

Actividad Nº1:

Realice un formulario que posea un input text donde se introducirá una Frase y un botón procesar. Cuando se oprima este, se deberá devolver la cantidad de palabras que tiene la Frase.

Actividad Nº2:

Modifique la actividad anterior de manera que se devuelva una tabla con la lista de palabras que forman la frase.

Actividad Nº3:

Realice un formulario donde se ingresen el apellido, nombre y dirección de correo electrónico del usuario. Realice una función denominada Controlar que verifique que la dirección de correo contenga el carácter @ y el carácter '.'
Convierta el apellido y el nombre a mayúsculas.

Actividad Nº4:

Realice un formulario que permita ingresar una frase y una palabra o letra que se desea buscar en la frase. Desarrolle una función **Buscar** que encuentre la palabra dentro de la frase y cuente la cantidad de veces que la misma aparece en ella.

Actividad Nº5:

Modifique la actividad anterior de manera que cuando se oprima el botón Buscar, cambie a color rojo todos los lugares en la frase donde aparece la palabra o letra buscada.

Actividad Nº6:

Realice un formulario donde se introduzca una frase y el botón Analizar.
Al oprimir este se deberá descomponer la frase en todas palabras que la forman y devolver una tabla conteniendo dichas palabras sin repetir y con el numero de veces que aparecen las mismas en la frase. Ejemplo:
Si se introduce la Frase: " Las Montañas de Mendoza son más altas que las de la Rioja" se deberá devolver:

Palabra	Cantidad
Las	2
Montañas	1
De	2
Mendoza	1
Son	1
Más	1
Que	1
Altas	1
La	1
Rioja	1

Actividad N° 7:

Crea un documento HTML (página web) que contenga un formulario. Se pedirá al usuario que introduzca nombres, apellidos,DNI y correo electrónico.
Defina dentro de la etiqueta form que cuando se produzca el **evento onsubmit** (pulsación del botón de envío del formulario) se ejecute una función a la que denominaremos **Validar**.

El formulario deberá contener los input para ingresar los datos y un input type="submit" que será el botón enviar.

En la etiqueta del Formulario colocaremos el evento onsubmit="Validar()"

El formulario debería quedar más o menos así:

```
    <form action="" onsubmit="Validar()">
      Apellido:  <input type="text" name="apellido" value="" id="apellido" placeholder="Ingresa tu
 apellido"> <br><br><hr>
      Nombres:   <input type="text" name="nombres" value="" id="nombres" placeholder="Ingresa tus
nombres"> <br><br><hr>
      DNI:   <input type="text" name="dni" value="" id="dni" placeholder="Ingresa tu DNI"> <br><br
><hr>
      Email:     <input type="text" name="email" value="" id="email" placeholder="Ingresa tu email
"><br><br><hr>
                <input type="submit"  value="Enviar Datos">
    </form>
```

La función validar debe realizar estas tareas y comprobaciones:

a) Comprobar que el nombre contiene **al menos tres letras**. Si no es así, debe aparecer un mensaje por pantalla indicando que el texto no cumple tener al menos tres letras. Por ejemplo si se trata de enviar Ka como nombre debe aparecer el mensaje: "El nombre deb tener al menos tres letras".

b) Comprobar que el correo electrónico contiene el **carácter @** (arroba) y el carácter . (punto). De no ser así, deberá aparecer un mensaje indicando que al correo electrónico le falta uno o ambos caracteres. Por ejemplo si se trata de enviar pacogmail.com deberá aparecer el mensaje: "Falta el símbolo @ en el correo electrónico".

c) Comprobar que el DNI contiene al menos 7 caracteres numéricos y los puntos separadores de miles. Una estructura correcta para DNI debe tener el formato 0.000.000

d)Antes de enviarse los datos del formulario a la página de destino, todas las letras del correo electrónico deben **transformarse a minúsculas**. Por ejemplo si se ha escrito PACO@GMAIL.COM debe enviarse paco@gmail.com