3.- Objetos nativos en Javascript.

Esto le va a ser muy útil para realizar su aplicación ya que tendrá que realizar diferentes tipos de conversiones de datos, trabajar intensivamente con cadenas y por supuesto con fechas y horas.

Aunque no hemos visto como crear objetos, sí que ya hemos dado unas pinceladas a lo que son los objetos, propiedades y métodos.

En esta sección vamos a echar una ojeada a objetos que son nativos en JavaScript, esto es, aquello que JavaScript nos da, listos para su utilización en nuestra aplicación.

Echaremos un vistazo a los objetos `String`, `Math`, `Number`, `Boolean` y `Date`.

"Si me hubieran hecho objeto sería objetivo, pero me hicieron sujeto."

BERGAMÍN, José.

3.1.- Objeto String.

Una cadena (`string`) consta de uno o más caracteres de texto, rodeados de comillas simples o dobles; da igual cuales usemos ya que se considerará una cadena de todas formas, pero en algunos casos resulta más cómodo el uso de unas u otras. Por ejemplo si queremos meter el siguiente texto dentro de una cadena de JavaScript:

```
<input type="checkbox" name="coche" />Audi A6
```

Podremos emplear las comillas dobles o simples:

```
var cadena = '<input  type="checkbox"  name="coche"  />Audi A6';
var cadena = "<input  type='checkbox'  name='coche'  />Audi A6";
```

Si queremos emplear comillas dobles al principio y fin de la cadena, y que en el contenido aparezcan también comillas dobles, tendríamos que escaparlas con `\"`, por ejemplo:

```
var cadena = "<input  type=\"checkbox\"  name=\"coche\"  />Audi A6";
```

Cuando estamos hablando de cadenas muy largas, podríamos concatenarlas con `+=`, por ejemplo:

```
var  nuevoDocumento  =  "";
nuevoDocumento  +=  "<!DOCTYPE  html>";
nuevoDocumento  +=  "<html>"  ;
```

```
nuevoDocumento  +=  "<head>";
nuevoDocumento  +=  '<meta  http-equiv="content-type"';
nuevoDocumento  +=  '  content="text/html;charset=utf-8">';
```

Si queremos concatenar el contenido de una variable dentro de una cadena de texto emplearemos el símbolo `+` :

```
nombreEquipo  =  prompt("Introduce el nombre de tu equipo favorito:","");
var  mensaje= "El  "  +  nombreEquipo +  "  ha sido el campeón de la Copa del Rey!";
alert(mensaje);
```

Caracteres especiales o caracteres de escape.

La forma en la que se crean las cadenas en JavaScript, hace que cuando tengamos que emplear ciertos caracteres especiales en una cadena de texto, tengamos que escaparlos, empleando el símbolo `\` seguido del carácter.

Vemos aquí un listado de los caracteres especiales o de escape en JavaScript:

Caracteres de escape y especiales en JavaScript	
Símbolos	**Explicación**
\"	Comillas dobles.
\'	Comilla simple.
\\	Barra inclinada.
\b	Retroceso.
\t	Tabulador.
\n	Nueva línea.
\r	Salto de línea.
\f	Avance de página.

El siguiente enlace amplía información sobre el objeto `String` y todas sus propiedades y métodos.

Más información y ejemplos sobre el objeto String.

3.1.1.- Propiedades y métodos del objeto String.

Para crear un objeto `String` lo podremos hacer de la siguiente forma:

```
var  miCadena = new  String("texto de la cadena");
```

O también se podría hacer:

```
var miCadena = "texto de la cadena";
```

Es decir, cada vez que tengamos una cadena de texto, en realidad es un objeto `String` que tiene propiedades y métodos:

```
cadena.propiedad;
cadena.metodo( [parámetros] );
```

Propiedades del objeto String	
Propiedad	**Descripción**
`length`	Contiene la longitud de una cadena.

Métodos del objeto String	
Métodos	**Descripción**
`charAt()`	Devuelve el carácter especificado por la posición que se indica entre paréntesis.
`charCodeAt()`	Devuelve el Unicode del carácter especificado por la posición que se indica entre paréntesis.
`concat()`	Une una o más cadenas y devuelve el resultado de esa unión.

`fromCharCode()`	Convierte valores Unicode a caracteres.
`indexOf()`	Devuelve la posición de la primera ocurrencia del carácter buscado en la cadena.
`lastIndexOf()`	Devuelve la posición de la última ocurrencia del carácter buscado en la cadena.
`match()`	Busca una coincidencia entre una expresión regular y una cadena y devuelve las coincidencias o null si no ha encontrado nada.
`replace()`	Busca una subcadena en la cadena y la reemplaza por la nueva cadena especificada.
`search()`	Busca una subcadena en la cadena y devuelve la posición dónde se encontró.
`slice()`	Extrae una parte de la cadena y devuelve una nueva cadena.
`split()`	Divide una cadena en un array de subcadenas.
`substr()`	Extrae los caracteres de una cadena, comenzando en una determinada posición y con el número de caracteres indicado.
`substring()`	Extrae los caracteres de una cadena entre dos índices especificados.
`toLowerCase()`	Convierte una cadena en minúsculas.
`toUpperCase()`	Convierte una cadena en mayúsculas.

Ejemplos de uso:

```
var cadena="El parapente es un deporte de riesgo medio";
document.write("La longitud de la cadena es: "+ cadena.length + "<br/>");
document.write(cadena.toLowerCase()+ "<br/>");
document.write(cadena.charAt(3)+ "<br/>");
document.write(cadena.indexOf('pente')+ "<br/>");
document.write(cadena.substring(3,16)+ "<br/>");
```