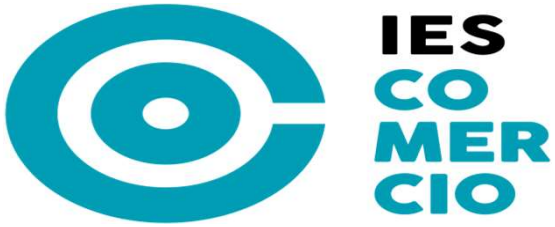



# Tema 4:
# Modelo de objetos predefinidos en JavaScript.

**Desarrollo Web en Entorno Cliente**

(Curso 2019 - 2020)

# Objetivos

- Conocer cuáles son los principales objetos predefinidos de JavaScript.
- Comprender las propiedades y métodos de los principales objetos de JavaScript.
- Aprender a generar código HTML desde sentencias JavaScript.
- Dominar el uso de los marcos de HTML.
- Manipular y gestionar la creación y apariencia de las ventanas del navegador.

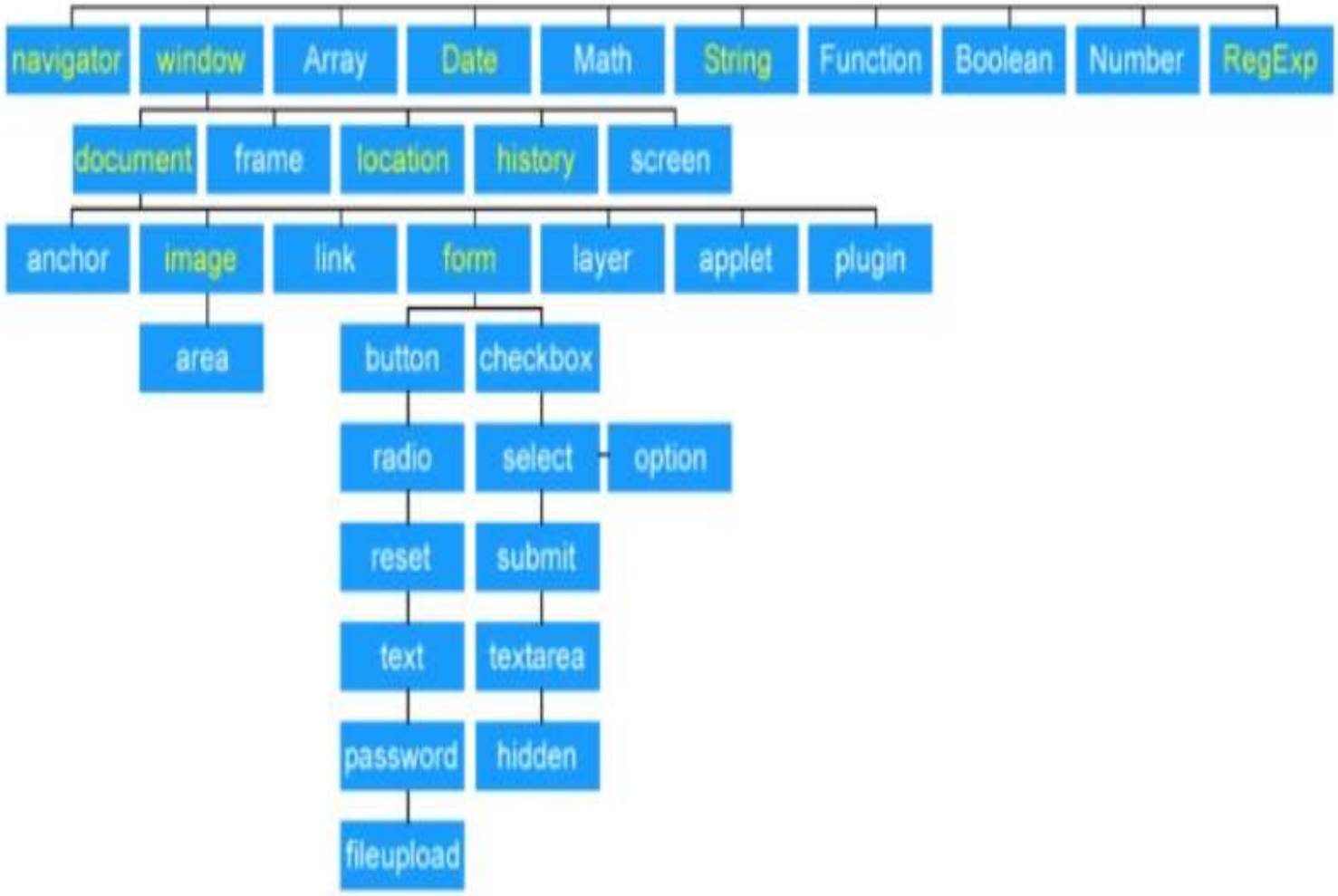
# Contenidos

1. Introducción.
2. Objetos nativos de JavaScript.
   - Date, Math, Number, String, …
3. Interacción de los objetos con el navegador.
4. Generación de elementos HTML desde código JavaScript.
5. Aplicaciones prácticas de los macros.
6. Gestión de las ventanas.

# 1.- Introducción.

- Un objeto es una estructura que posee propiedades y métodos.
  - Las ***propiedades*** son como los adjetivos en el lenguaje, es decir que expresan una cualidad del objeto. Otras propiedades pueden darnos información acerca del objeto.
  - Los ***métodos*** son funcionalidades, es decir funciones o modos de operar asociados a un objeto.
- Los objetos de JavaScript se ordenan de modo jerárquico.

# 1.- Introducción.



# 2.- Objetos nativos de JavaScript.

- JavaScript proporciona una serie de objetos definidos nativamente que no dependen del navegador.
- Para crear un objeto se utiliza la palabra clave **new**.
  - Ejemplo:

```
var mi_objeto= new Object();
```

- En JavaScript se accede a las propiedades y a los métodos de los objetos mediante el operador punto (“.”):

```
mi_objeto.nombre_propiedad;

mi_objeto.nombre_función([parámetros]);
```

# 2.- Objetos nativos de JavaScript.

- **El objeto Date:**
  - Permite realizar controles relacionados con el tiempo en las aplicaciones web.
  - Cuenta con una serie de métodos divididos en tres subconjuntos:
    - Métodos de lectura.
    - Métodos de escritura.
    - Métodos de conversión.

# 2.- Objetos nativos de JavaScript.

- El objeto Date –Métodos:

| Métodos | | | | |
|---|---|---|---|---|
| getDate() | getTime() | getUTCMonth() | setMonth() | setUTCMonth() |
| getDay() | getTimezoneOffset() | getUTCSeconds() | setSeconds() | setUTCSeconds() |
| getFullYear() | getUTCDate() | parse() | setTime() | toDateString() |
| getHours() | getUTCDay() | setDate() | setUTCDate() | toLocaleDateString() |
| getMilliseconds() | getUTCFullYear() | setFullYear() | setUTCFullYear() | toLocaleTimeString() |
| getMinutes() | getUTCHours() | setHours() | setUTCHours() | toLocaleString() |
| getMonth() | getUTCMilliseconds() | setMilliseconds() | setUTCMilliseconds() | toTimeString() |
| getSeconds() | getUTCMinutes() | setMinutes() | setUTCMinutes() | toUTCString() |

# 2.- Objetos nativos de JavaScript.

- **Ejemplo: Diferencia en minutos entre dos fechas.**

```
<script type="text/javascript">
  var fecha_uno = new Date(2012, 11, 21);
  var fecha_dos = new Date();
        var tiempo_restante = fecha_dos -
fecha_uno;
        alert("Hay " + tiempo_restante + "
milisegundos entre " + fecha_uno + " y " +
fecha_dos);
  //Dividimos 1000->Segundos 60->Minutos 60-
>Horas 24->Dias
  var tiempo_minutos = tiempo_restante/60000;
  alert("Hay " + tiempo_minutos + " minutos entre
" + fecha_uno + " y " + fecha_dos);
</script>
```

# 2.- Objetos nativos de JavaScript.

- **El objeto Math:**
  - No es un constructor, es decir, no podemos crear objetos de tipo Math, pero si podemos llamar a sus propiedades y métodos
  - Permite realizar operaciones matemáticas complejas en JavaScript.

| Propiedad | Descripción |
|---|---|
| E | Devuelve el número Euler (aproximadamente 2.718). |
| LN2 | Devuelve el logaritmo neperiano de 2 ( aproximadamente 0.693). |
| LN10 | Devuelve el logaritmo neperiano de 10 ( aproximadamente 2.302). |
| LOG2E | Devuelve el logaritmo base 2 de E ( aproximadamente 1.442). |
| LOG10E | Devuelve el logaritmo base 10 de E ( aproximadamente 0.434). |
| PI | Devuelve el número PI ( aproximadamente 3.14159). |
| SQRT2 | Devuelve la raíz cuadrada de 2 ( aproximadamente 1.414). |

# 2.- Objetos nativos de JavaScript.

| Método | Descripción |
|---|---|
| abs(x) | Devuelve el valor absoluto de x. |
| acos(x) | Devuelve el arcocoseno de x, en radianes. |
| asin(x) | Devuelve el arcoseno de x, en radianes. |
| atan(x) | Devuelve el arcotangente de x, en radianes con un valor entre -PI/2 y PI/2. |
| atan2(y,x) | Devuelve el arcotangente del cociente de sus argumentos. |
| ceil(x) | Devuelve el número x redondeado a la alta hacia el siguiente entero. |
| cos(x) | Devuelve el coseno de x (x está en radianes). |
| floor(x) | Devuelve el número x redondeado a la baja hacia el anterior entero. |
| log(x) | Devuelve el logaritmo neperiando (base E) de x. |

| Método | Descripción |
|---|---|
| max(x,y,z,...,n) | Devuelve el número más alto de los que se pasan como parámetros. |
| min(x,y,z,...,n) | Devuelve el número más bajo de los que se pasan como parámetros. |
| pow(x,y) | Devuelve el resultado de x elevado a y. |
| random() | Devuelve un número al azar entre 0 y 1. |
| round(x) | Redondea x al entero más próximo. |
| sin(x) | Devuelve el seno de x (x está en radianes). |
| sqrt(x) | Devuelve la raíz cuadrada de x. |
| tan(x) | Devuelve la tangente de un ángulo. |

# 2.- Objetos nativos de JavaScript.

- **Ejemplo: Calcular el área de un círculo ($A=PI*r^2$).**

```
<script type="text/javascript">
  var r = prompt("Introduce el radio
  cms de un circulo:");
  var area = Math.PI * Math.pow(r,
  2);
  alert("El área del circulo es de: "
  + area + " cms cuadrados");
 </script>
```

# 2.- Objetos nativos de JavaScript.

- **Ejemplo: Mostrar todas las propiedades del objeto, crea además una instancia del valor de PI y formatea con dos decimales.**

```
<script type="text/javascript">
    alert("Propiedad MAX_VALUE: " + Number.MAX_VALUE)
    alert("Propiedad MIN_VALUE: " + Number.MIN_VALUE)
    alert("Propiedad NaN: " + Number.NaN)
    alert("Propiedad NEGATIVE_INFINITY: " +
Number.NEGATIVE_INFINITY)
    alert("Propiedad POSITIVE_INFINITY: " +
Number.POSITIVE_INFINITY)
    var N1 = new Number(3.141592653589793);
    alert("Pi griego formateado: " + N1.toPrecision(3))
</script>
```

# 2.- Objetos nativos de JavaScript.

- **El objeto Number:**
  - Permite realizar tareas relacionadas con tipos de datos numéricos.

| Propiedad | Descripción |
|---|---|
| Constructor | Devuelve la función que creó el objeto Number. |
| MAX_VALUE | Devuelve el número más alto disponible en JavaScript. |
| MIN_VALUE | Devuelve el número más pequeño disponible en JavaScript |
| NEGATIVE_INFINITY | Representa a infinito negativo (se devuelve en caso de overflow) |
| POSITIVE_INFINITY | Representa a infinito positivo (se devuelve en caso de overflow) |
| Prototype | Permite añadir nuestras propias propiedades y métodos a un objeto. |

| Métodos | Descripción |
|---|---|
| toExponential() | Convierte el número en una notación exponencial. |
| toFixed() | Formatea el número con la cantidad de dígitos decimales que pasemos como parámetro. |
| toPrecision() | Formatea el número con la longitud que pasemos como parámetro. |
| toString() | Convierte un objeto Number en una cadena:<br>Si se pone 2 como parámetro se mostrará el número en binario.<br>Si se pone 8 como parámetro se mostrará el número en octal.<br>Si se pone 16 como parámetro se mostrará el número en hexadecimal. |
| valueOf() | Devuelve el valor primitivo de un objeto Number |

# 2.- Objetos nativos de JavaScript.

- **El objeto Boolean:**
  - Convierte un valor no booleano aun valor booelano (true o false).

| Propiedad | Descripción |
|---|---|
| Constructor | Devuelve la función que creó el objeto Boolean. |
| Prototype | Te permitirá añadir propiedades y métodos a un objeto. |

| Métodos | Descripción |
|---|---|
| toString() | Convierte un valor Boolean a una cadena: y devuelve el resultado. |
| valueOf() | Devuelve el valor primitivo de un objeto Boolean. |

Algunos ejemplos de uso:

```
var bool = new Boolean (1);
document.write(bool.toString());
document.write(bool.valueof());
```

Se inicializa a false cuando: no se pasa valor al constructor, se pasa la cadena vacía, la palabra false sin comillas o el número 0.

# 2.- Objetos nativos de JavaScript.

- **Ejemplos de uso del objeto Boolean.**

```
var  b1 = new Boolean()
document.write(b1 +
"</br>")
// muestra false
var  b2 = new Boolean("")
document.write(b2 +
"</br>")
// muestra false
var  b25 = new
Boolean(false)
document.write(b25 +
"</br>")
// muestra false
var  b3 = new Boolean(0)
document.write(b1 +
"</br>")
// muestra false
```

```
var  b35 = new Boolean("0")
document.write(b35+
"</br>")
// muestra true
var  b4 = new Boolean(3)
document.write(b3 +
"</br>")
// muestra true
var  b5 = new
Boolean("Hola")
document.write(b5 +
"</br>")
// muestra true
```

# 2.- Objetos nativos de JavaScript.

- **El objeto String:**
  - Permite manipular las cadenas de texto.
  - Podemos utilizar comillas dobles o comillas sencillas.
  - Podemos crear un objeto string:
    - Por medio del constructor.

```
var texto = new String("Prueba");
```

    - Asignando a la variable una cadena directamente.

```
var texto = "Prueba de texto");
```

  - El primer carácter de una cadena se considera el 0.

# 2.- Objetos nativos de JavaScript.

| Propiedad | Descripción |
|---|---|
| Constructor | Crea una instancia del objeto, el valor que se le de, dependerá de lo que se pase como parámetro. |
| Lenght | Indica el número de caracteres que tiene una cadena. |

```
<script type="text/javascript">
     //Instancia de String.
    var nombre = new String("Eulogio");
    alert ("nombre: " + nombre + "tiene una longitud de: " +
  nombre.Lenght );
     // Serán 7.
    nombre = "";
    alert ("nombre: " + nombre + "tiene una longitud de: " +
  nombre.Lenght );
   // Serán 0.
</script>
```

# 2.- Objetos nativos de JavaScript.

| Métodos | Descripción |
|---|---|
| anchor() | Devuelve una cadena convertida en un ancla de HTML. |
| big() | Aumenta el tamaño de una cadena. |
| blink() | Crea el efecto de una cadena parpadeante. |
| blod() | Muestra una cadena en negrita. |
| charAt() | Permite acceder a un carácter en concreto de una cadena. |
| charCodeAt() | Devuelve un carácter en concreto de una cadena. |
| concat() | Concatena dos o más cadenas y devuelve una nueva de dicha concatenación. |
| fixed() | Convierte una cadena con fuente monoespaciada. |
| fontcolor() | Modifica el color de la fuente de una cadena. |
| fontsize() | Modifica el tamaño de una fuente de una cadena. |
| fromCharCode() | Convierte valores Unicode a caracteres. |
| indexOf() | Devuelve la posición de la primera ocurrencia del carácter pasado como parámetro. |
| Italics() | Muestra la cadena en cursiva. |
| lastIndexOf() | Devuelve la posición de la última ocurrencia del carácter pasado como parámetro. |
| link() | Muestra una cadena como un hipervínculo HTML con el enlace que le pasaremos como parámetro. |
| match() | Busca una coincidencia en una cadena y devuelve todas las coincidencias encontradas. |

| Métodos | Descripción |
|---|---|
| replace() | Busca una coincidencia en una cadena y si existe, la reemplaza por otra cadena pasada como parámetro. |
| search() | Busca una coincidencia en una cadena y devuelve la posición de la coincidencia. |
| slice() | Extrae una parte de una cadena en base a los parámetros que indiquemos como índices de inicio y final. |
| small() | Disminuye el tamaño de la cadena. |
| split() | Corta una cadena en base a un separador que pasemos como parámetro. |
| strike() | Muestra una cadena tachada. |
| sub() | Muestra una cadena como subíndice. |
| substr() | Devuelve una subcadena en base al índice y longitud pasados como parámetros. |
| substring() | Devuelve una subcadena en base a un índice de inicio y de final pasados como parámetros. |
| sup() | Muestra una cadena como superíndice. |
| toLowerCase() | Convierte una cadena a minúsculas. |

# 2.- Objetos nativos de JavaScript.

- **Ejemplo: Dar formato a una cadena de texto en JavaScript.**

```
<script type="text/javascript">
        var texto = new String("Prueba de texto ");
        document.write("Numero de letras de la cadena de
  texto: " + texto.length + "<br>")
        document.write("Cursiva: " + texto.italics() +
  "<br>");          document.write("Negrita: " +
  texto.bold() + "<br>");
        document.write("Rojo: " +
  texto.fontcolor("#FF0000") + "<br>");
        document.write("Muy grande: " +
  texto.fontsize(20) + "<br>");
        document.write("Tachado: " + texto.strike() +
  "<br>");
</script>
```

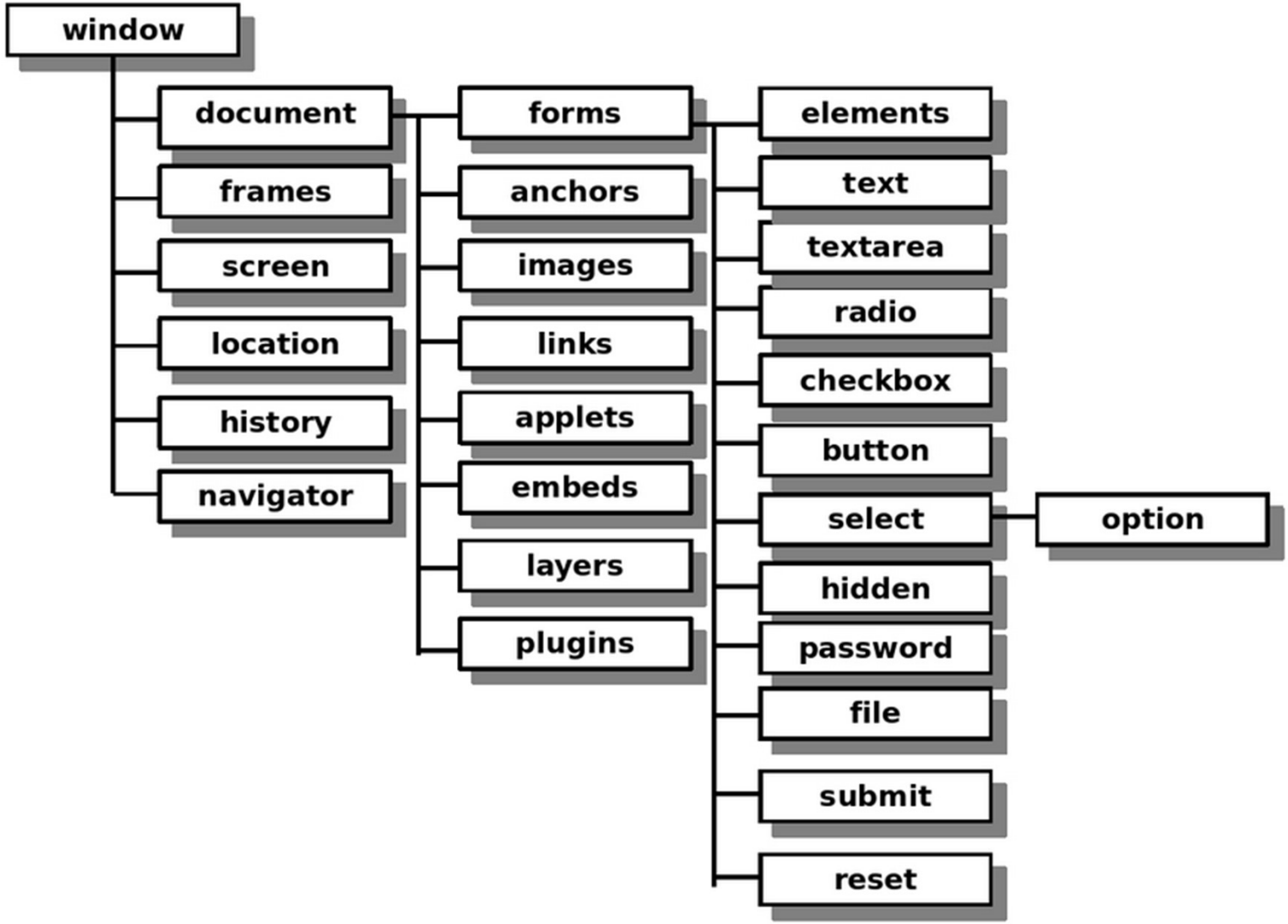
# Actividad 1

Realiza un script que calcule el área de un triángulo. Recuerda que area=(base*altura)/2.

# 3.- Interacción de los objetos con el navegador.

- Además de los objetos presentados anteriormente, existe otro tipo de objetos que permiten manipular diferentes características del navegador en sí mismo, como que mensajes mostrarnos en la barra de estado o como crear nuevas cadenas.
- Vamos a ver los siguientes objetos:
  - Objeto navigator.
  - Objeto screen.
  - Objeto window.
  - Objeto history.
  - Objeto document.
  - Objeto location.

Jerarquía de objetos creados por el navegador al cargar una página Web:



# 3.- Interacción de los objetos con el navegador.

## 3.1.- El objeto Window:

- Se considera el objeto más importante de JavaScript.
- Ocupa el nivel superior en la jerarquía de objetos de JavaScript, en cierto modo, es el contenedor de todo lo que se puede mostrar en una ventana del navegador Web. A través del objeto ***window*** se puede consultar y controlar la ventana del documento.
- También permite definir ventanas nuevas, asignando libremente las propiedades de ventana.
- A la ventana principal del navegador se puede acceder a través de los nombres reservados ***window*** o ***self***. Estos nombres permiten utilizar todas las propiedades y métodos del objeto window. Como se trata de un objeto implícito, habrá ocasiones según el código, que no será necesario nombrarlo para acceder a los objetos que se encuentran debajo de su nivel de jerarquía. Ejemplo: document.write();

# 3.- Interacción de los objetos con el navegador.

## 3.1.- El objeto window:

- Un script que esté ejecutándose en una de las ventanas principales del navegador, podrá crear o abrir nuevas sub-ventanas.
- El método que genera una nueva ventana es `window.open()`. Este método contiene hasta tres parámetros: la URL del documento a abrir, el nombre de esa ventana y su apariencia física (tamaño, color, etc.)

```
var subVentana=window.open("nueva.html","nueva","height=800,width=600");
```

- Si quisiéramos cerrar la nueva ventana desde nuestro script utilizaríamos el método `close()`.

```
subVentana.close();
```

# 3.- Interacción de los objetos con el navegador.

## 3.1.-El objeto Window:

| Propiedad | Descripción |
|---|---|
| closed | Corresponde al valor booleano que indica si la ventana está cerrada o no. |
| defaultStatus | Corresponde a la cadena de texto de la barra de estado del navegador. |
| document | Corresponde al documento actual de la ventana. |
| frames | Corresponde al conjunto de marcos de la ventana. |
| history | Corresponde al conjunto de elementos que representan las URL visitadas. |
| innerHeight | Corresponde a la altura utilizable de la ventana. |
| innerWidth | Corresponde al ancho utilizable de la ventana. |
| length | Corresponde al número de frames de la ventana. |
| location | Corresponde a la URL de la barra de direcciones. |
| locationbar | Corresponde a la barra de direcciones del navegador. |
| menubar | Corresponde a la barra del menú del navegador. |
| name | Corresponde al nombre de la ventana. |
| opener | Corresponde a la referencia del objeto window que haya abierto una ventana nueva. |
| outerHeight | Corresponde a la altura exterior de la página. |
| outerWidth | Corresponde al ancho exterior de la página. |
| pageXoffset | Corresponde a la posición horizontal de la ventana. |
| pageYoffset | Corresponde a la posición vertical de la ventana. |
| parent | Corresponde a la referencia del objeto window que contiene los marcos de una página con marcos |
| personalbar | Corresponde a la barra de herramientas personal. |
| scrollbars | Corresponde a las barras de desplazamiento horizontal y vertical. |
| self | Corresponde a la ventana actual. |
| status | Corresponde a la cadena con el mensaje que contiene la barra de estado. |
| toolbar | Corresponde a la barra de herramientas del navegador. |
| top | Corresponde a la ventana de nivel superior. |

# 3.- Interacción de los objetos con el navegador.

## 3.1.-El objeto Window:

| Métodos | Descripción |
|---|---|
| alert(texto) | Presenta una ventana de alerta donde se puede leer el texto que recibe por parámetro. |
| back() | Ir una página atrás en el historial de páginas visitadas. Funciona como el botón de volver de la barra de herramientas |
| blur() | Desactiva la ventana. |
| close() | Cierra la ventana. |
| confirm(texto) | Muestra una ventana de confirmación y permite aceptar o rechazar. |
| find() | Muestra una ventanita de búsqueda. |
| focus() | Coloca el foco de la aplicación en la ventana. |
| forward() | Ir una página adelante en el historial de páginas visitadas. Como si pulsásemos el botón de adelante del navegador. |
| home() | Ir a la página de inicio que haya configurada en el explorador. |
| moveBy(pixelsX,pixelsY) | Mueve la ventana del navegador los pixels que se indican por parámetro hacia la derecha y abajo. |
| moveTo(pixelsX,pixelsY) | Mueve la ventana del navegador a la posición indicada en las coordenadas que recibe por parámetro. |
| open() | Abre una ventana secundaria del navegador. |
| print() | Abre una ventana secundaria del navegador. |
| prompt() | Muestra una caja de diálogo para pedir un dato. Devuelve el dato que se ha escrito. |

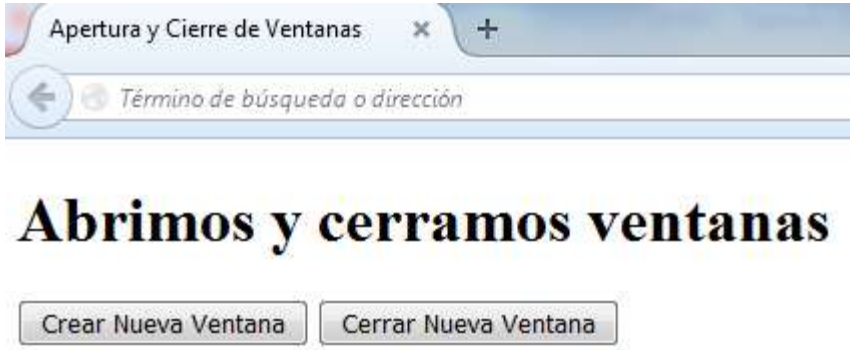
# 3.- Interacción de los objetos con el navegador.

## 3.1.-El objeto Window:

| | |
|---|---|
| resizeBy(pixelsAncho,pixelsAlto) | Redimensiona el tamaño de la ventana, añadiendo a su tamaño actual los valores indicados en los parámetros. Elprimero para la altura y el segundo para la anchura. Admite valores negativos si se desea reducir la ventana. |
| resizeTo(pixelsAncho,pixelsAlto) | Redimensiona la ventana del navegador para que ocupe el espacio en pixels que se indica por parámetro. |
| scroll(pixelsX,pixelsY) | Hace un scroll de la ventana hacia la coordenada indicada por parámetro. Este método está desaconsejado, pues ahora se debería utilizar scrollTo() |
| **scrollBy(pixelsX,pixelsY)** | Hace un scroll del contenido de la ventana relativo a la posición actual. |
| scrollTo(pixelsX,pixelsY) | Hace un scroll de la ventana a la posición indicada por el parámetro. Este método se tiene que utilizar en lugar de scroll. |
| setInterval() | Define un script para que sea ejecutado indefinidamente en cada intervalo de tiempo. |
| setTimeout(sentencia,milisegundos) | Define un script para que sea ejecutado una vez después de un tiempo de espera determinado |
| stop() | Detiene una página. |

# Actividad 2

Crear un script que permita abrir y cerrar una subventana.




- Crear dos botones en un formulario, uno para abrir y otro para cerrar.
- Crear las funciones necesarias para llevar a cabo la operación correspondiente al hacer clic sobre el botón.

# 3.- Interacción de los objetos con el navegador.

## 3.3.- El objeto navigator:

- Permite identificar las características de la plataforma sobre la cual se ejecuta la aplicación web.

| Propiedad | Descripción |
|---|---|
| appCodeName | Devuelve el código del nombre del navegador. |
| appName | Devuelve el nombre del navegador. |
| appVersion | Devuelve el código del nombre del navegador. |
| cookieEnable | Determina si las cookies están habilitadas o no |
| geolocation | Devuelve un objeto de utilidad para la geolocalización, útil para móviles. |
| language | Devuelve el idioma del navegador. |
| onLine | Indica si el navegador está funcionando en modo online (true) u offline (false). |
| platform | Devuelve la plataforma sobre la cual se está ejecutando el navegador. |
| userAgent | Devuelve una información completa sobre el agente de usuario, el cual es normalmente el navegador. |

| Métodos | Descripción |
|---|---|
| javaEnabled() | Informa si el navegador está habilitado para soportar la ejecución de programas escritos en Java. |

# 3.- Interacción de los objetos con el navegador.

- **Ejemplos de uso del objeto navigator:**

```
document.write("Navigator <b>appCodeName</b>: " + navigator.appCodeName + "<br>");
document.write("Navigator <b>appName</b>: " + navigator.appName + "<br>");
document.write("Navigator <b>appVersion</b>: " + navigator.appVersion + "<br>");
document.write("Navigator <b>language</b>: " + navigator.language + "<br>");
document.write("Navigator <b>platform</b>: " + navigator.platform + "<br>");
document.write("Navigator <b>userAgent</b>: " + navigator.userAgent + "<br>");
```

- **Realizar un script que verifique si el navegador está preparado para la ejecución de applets de Java, mostrando el mensaje en la pantalla.**

# 3.- Interacción de los objetos con el navegador.

## 3.3.- El objeto Document:

- Se refiere a los documentos que se cargan en la ventana del navegador.
- Permite manipular las propiedades y el contenido de los principales elementos de las páginas web (accede a todos los elementos html).
- Cuenta con una serie de sub-objetos como los vínculos, puntos de anclaje, imágenes o formularios.

# 3.- Interacción de los objetos con el navegador.

## 3.3.-El objeto Document:

| Propiedad | Descripción |
|---|---|
| alinkColor | Corresponde al color de los vínculos activos de la página. |
| anchors | Corresponde a los puntos de anclaje (etiquetas <a name>) del documento. |
| applets | Corresponde a los applets (etiquetas <applet>) Java del documento. |
| bgColor | Corresponde al color del fondo del documento. |
| cookie | Corresponde a un fichero guardado en el equipo del cliente del navegador con información sobre el usuario. |
| domain | Corresponde al nombre del dominio por defecto para el documento. |
| embeds | Corresponde a los objetos embebidos (etiqueta <embed> en el documento. |
| fgColor | Corresponde al color del texto del documento. |
| forms | Corresponde a los formularios (etiqueta <form>) del documento. |
| Images | Corresponde a las imágenes (etiqueta <images>) del documento. |
| lastModified | Corresponde a la última fecha en la que se modificó el documento. |
| layers | Corresponde a las capas (etiqueta <layer>) del documento. |
| linkColor | Corresponde al color de los enlaces aun no visitados. |
| links | Corresponde a los vínculos (etiqueta <a href>) del documento. |
| plugins | Corresponde a las referencias y llamadas de los plugins del documento. |
| referrer | Corresponde a la dirección del documento usado para ir al documento actual. |
| title | Corresponde al título (etiqueta <tittle>) del documento. |
| URL | Corresponde a la dirección del documento. |
| vlinkColor | Corresponde al color de los enlaces visitados. |

# 3.- Interacción de los objetos con el navegador.

## 3.3.-El objeto Document:

| Métodos | Descripción |
|---|---|
| captureEvents() | Intercepta un evento para que pueda ser manipulado por el documento. |
| close() | Cierra el documento activo. |
| getSelection() | Devuelve el texto seleccionado en el documento. |
| handleEvent() | Activa el manejador del evento especificado. |
| Home() | Carga la página de inicio. |
| Open() | Activa el documento. |
| releaseEvents() | Libera los eventos que han sido interceptados. |
| routeEvents() | Intercepta un evento y lo pasa a lo largo de la jerarquía del objeto que lo lanzó. |
| write() | Escribe datos en el documento. |
| writeln() | Escribe datos además de un salto de línea en el documento. |

# 3.- Interacción de los objetos con el navegador.

## 3.4.-El objeto History:

- Almacena las referencias de las páginas web visitadas.
- Las referencias se guardan en una lista utilizada principalmente para desplazarse entre dichas páginas web.
- No es posible acceder a los nombres de las URL, ya que es información privada.

# 3.- Interacción de los objetos con el navegador.

## 3.4.-El objeto History:

| Propiedad | Descripción |
|---|---|
| current | Corresponde a la cadena que contiene la URL de la entrada actual del historial. |
| length | Corresponde al número de páginas que han sido visitadas. |
| next | Corresponde a la cadena que contiene la siguiente entrada del historial. |
| previous | Corresponde a la cadena que contiene la anterior entrada del historial. |

| Métodos | Descripción |
|---|---|
| back() | Carga la URL del documento anterior del historial. |
| forward() | Carga la URL del documento siguiente del historial. |
| go() | Carga la URL del documento especificado por el índice que pasamos como parámetro dentro del historial. |

# 3.- Interacción de los objetos con el navegador.

## 3.5.- El objeto Screen:

- Corresponde a la pantalla utilizada por el usuario.
- No tiene ningún método y todas sus propiedades son de lectura.

| Propiedad | Descripción |
|---|---|
| availHeight | Corresponde a la altura disponible de la pantalla para el uso de ventanas. |
| availWidth | Corresponde a la anchura disponible de la pantalla para el uso de ventanas. |
| colorDepth | Corresponde al número de colores que puede representar la pantalla. |
| height | Corresponde a la altura total de la pantalla. |
| pixelDepth | Corresponde a la resolución de la pantalla expresada en bits por pixel. |
| width | Corresponde a la anchura total de la pantalla. |

# 3.- Interacción de los objetos con el navegador.

- **Ejemplos de uso del objeto screen:**

```
<script type="text/javascript">
    document.write("<br>Altura total: " + screen.height);
    document.write("<br>Altura disponible: " + screen.availHeight);
    document.write("<br>Anchura total: " + screen.width);
    document.write("<br>Anchura disponible: " + screen.availWidth);
    document.write("<br>Profundidad de color: " + screen.colorDepth);
</script>
```

# Actividad 3

Crear dos botones en un formulario para que el usuario pueda desplazarse adelante o atrás según su historial de navegación.

# 3.- Interacción de los objetos con el navegador.

## 3.6.-El objeto Location:

- Corresponde a la URL de la página web en uso.
- Su principal función es la de consultar las diferentes partes que forman una URL como por ejemplo:
  - El dominio.
  - El protocolo.
  - El puerto.
- Gracias a este objeto podemos recargar una página, cargar otra nueva o reemplazar una por otra.

# 3.- Interacción de los objetos con el navegador.

## 3.6.-El objeto Location:

| Propiedad | Descripción |
|---|---|
| hash | Corresponde a la cadena que representa el anclaje de la URL. Es decir, la parte de la URL que va después de la etiqueta #. |
| host | Corresponde a la cadena que representa el nombre del dominio del servidor y el número del puerto dentro de la URL. |
| hostname | Corresponde a la cadena que representa el nombre del dominio del servidor. |
| href | Corresponde a la URL completa. |
| pathname | Corresponde a la cadena que sigue al nombre del servidor. |
| port | Corresponde al número del puerto de la URL. |
| protocol | Corresponde al protocolo utilizado por la página web. |
| search | Corresponde a la cadena de búsqueda que se encuentra después de un signo de interrogación de a URL. |

| Métodos | Descripción |
|---|---|
| assign() | Carga un nuevo documento. |
| reload() | Carga de nuevo el documento actual. |
| replace() | Sustituye la URL del documento actual por otra URL. |

# Actividad 4

Crea un script donde se muestre la URL completa de la pagina web, el protocolo utilizado y el camino al recurso (path) dentro de la URL. Además crea un botón que al presionarlo, cargue de nuevo el documento actual.

# Actividad 5

En un documento HTML crea un botón que al presionarlo, utilice el método resizeTo (500,500) para modificar el tamaño de la ventana.

Nota: Los navegadores Firefox y Chrome «no permiten redimensionar ventanas» por motivos de seguridad, probarlo con Explorer.

# Actividad 5.2

Nota: En el caso de que se haya creado una ventana mediante window.open() Firefox y Chrome sí permitirán redimensionarla.

Modificar el ejercicio anterior para que redimensione la nueva ventana creada:

# 4.- Generación de elementos HTML desde código JavaScript.

- Uno de los principales objetivos de JavaScript es convertir un documento HTML estático en una aplicación web dinámica.
- Por ejemplo, es posible ejecutar instrucciones que crean nuevas ventanas con contenido propio, en lugar de mostrar dicho contenido en la ventana activa.

# 4.- Generación de elementos HTML desde código JavaScript.

- Con JavaScript es posible manipular los objetos que representan el contenido de una página web con el fin de crear documentos dinámicos.
- Por ejemplo, es posible definir el título de una página web basándose en el SO utilizado:

```
<script type="text/javascript">
    var SO = navigator.platform;
    document.write("<h1>Documento abierto con: " +
SO + "</h1>");
</script>
```

# 4.- Generación de elementos HTML desde código JavaScript.

- Otro ejemplo es crear documentos en ventanas emergentes:

```
<script type="text/javascript">
   var texto = prompt("Introduce un título
para   la nueva ventana: ");
   var ventanaNueva= window.open();
   ventanaNueva.document.write("<h1>" +
texto +   "</h1>");
</script>
```

# 4.- Generación de elementos HTML desde código JavaScript.

- La generación de código HTML a partir de JavaScript no se limita sólo a la creación de texto como en los ejemplos anteriores. Es posible crear y manipular todo tipo de objetos
- Por ejemplo, generar un formulario para modificar la propiedad del color de fondo de la página:

# 4.- Generación de elementos HTML desde código JavaScript.

- Es decir, obtener la siguiente página web dinámica:

# 4.- Generación de elementos HTML desde código JavaScript.

```
<script type="text/javascript">
  document.write("<form name=\"cambiacolor\">");
  document.write("<b>Selecciona un color para el fondo de página:</b><br>");
  document.write("<select name=\"color\">");
  document.write("<option value=\"red\">Rojo</option>");
  document.write("<option value=\"blue\">Azul</option>");
  document.write("<option value=\"yellow\">Amarillo</option>");
  document.write("<option value=\"green\">Verde</option>");
  document.write("</select>");
  document.write("<input type=\"button\" value=\"Modifica el color\"
  onclick=\"document.bgColor=document.cambiacolor.color.value\">");
  document.write("</form>");
</script>
```

# Actividad 6

Modifica el ejemplo anterior agregando dos colores más al conjunto de colores del fondo de la página.

# 5.- Aplicaciones prácticas de los marcos.

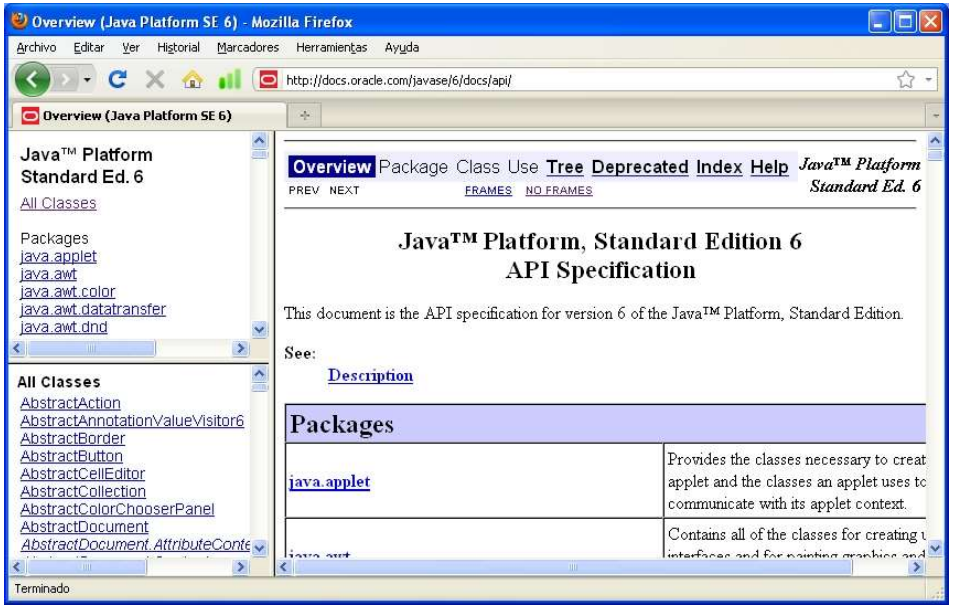
- Es posible dividir la ventana de una aplicación web en dos o más partes independientes.
- Con JavaScript se puede interactuar entre estos sectores independientes.
- Dichos sectores se denominan marcos.

# 5.- Aplicaciones prácticas de los marcos.

Algunas páginas web presentan una estructura en la cual una parte permanece fija mientras que otra va cambiando.

Por ejemplo la página de la API de Java:

http://docs.oracle.com/javase/6/docs/api/



# 5.- Aplicaciones prácticas de los marcos.

- Los marcos se definen utilizando HTML mediante estas etiquetas:
  - `<frameset>` : para indicar al navegador el número de marcos, tamaño, etc. Los principales atributos son `cols` y `rows`.
  - `<frame>`: esta etiqueta define las características de cada marco. Sus principales atributos son:

| Atributos | Descripción |
|---|---|
| frameborder | Define si mostrar o no el borde del marco. |
| marginheight | Permite cambiar los márgenes verticales del marco. |
| marginwidth | Permite cambiar los márgenes horizontales del marco. |
| name | Asigna un nombre al marco. |
| noresize | Evita que el usuario pueda modificar el tamaño del marco. |
| scrolling | Permite elegir si posiciona o no una barra de desplazamiento en el marco. |
| scr | Indica la URL de documento HTML que contendrá el marco. |

# 5.- Aplicaciones prácticas de los marcos.

- JavaScript permite manipular los marcos mediante las propiedades frames, parent y top del objeto window.
- Por ejemplo, se define un documento HTML con dos marcos:

```
<html><head><title>Ejemplos de control de
marcos</title></head>
    <frameset cols="50%,50%">
      <frame src="Marco1.html" name="Marco1" noresize>
      <frame src="Marco2.html" name="Marco2" noresize>
    </frameset>
  <body></body>
</html>
```

# 5.- Aplicaciones prácticas de los marcos.

- El primer marco (`Marco1`) contiene la página `Marco1.html`:

```
<html><body>
    <form name="form1">
      <select name="color">
        <option value="green">Verde
        <option value="blue">Azul
      </select><br><br>
      <select name="marcos">
        <option value="0">Izquierda
        <option value="1">Derecha
      </select>
    </form>
</body></html>
```

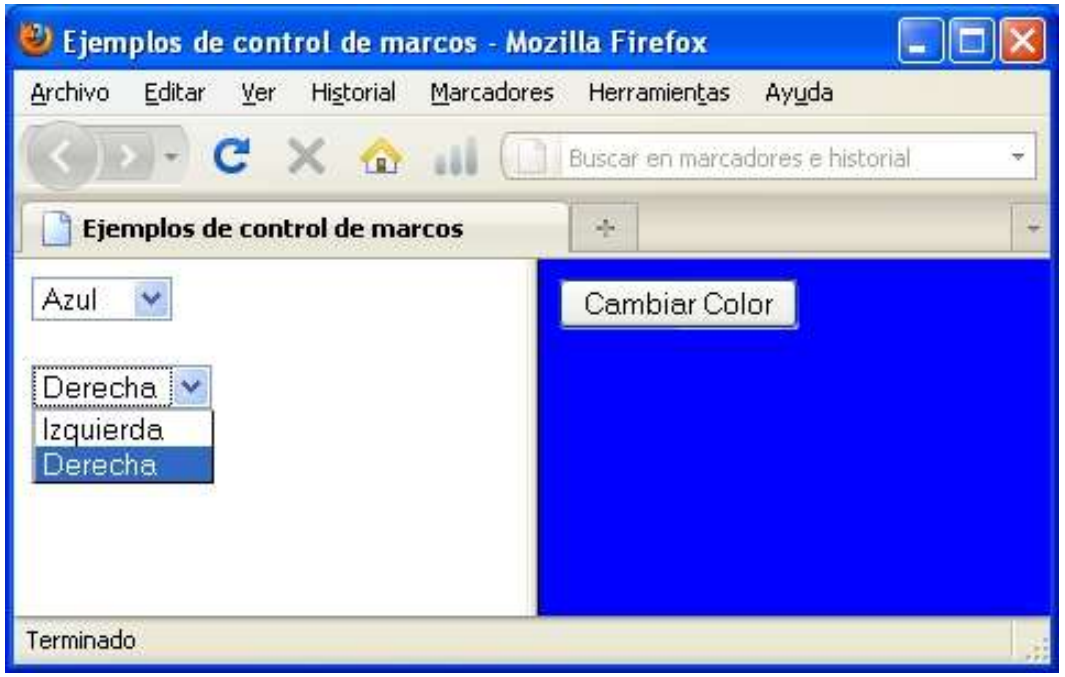
# 5.- Aplicaciones prácticas de los marcos.

- El segundo marco (`Marco2`) contiene la página `Marco2.html`:

```
<html><body><form>
      <input type="Button" value="Cambiar Color" onclick="
        campoColor = parent.Marco1.document.form1.color;
        if(campoColor.selectedIndex==0){colorin = 'green';}
        else{colorin = 'blue';}
        campoFrame = parent.Marco1.document.form1.marcos
        if(campoFrame.selectedIndex==0){
          window.parent.Marco1.document.bgColor = colorin
        }else{
          window.parent.Marco2.document.bgColor = colorin
        }">
</form></body></html>
```

# 5.- Aplicaciones prácticas de los marcos.

- El resultado se puede ver en esta imagen:



# Actividad 7

Modifica el código del ejemplo de marcos utilizado anteriormente, con el fin de ocultar el borde y que no se note la separación entre ellos.

# 6.- Gestión de las ventanas.

- JavaScript permite gestionar diferentes aspectos relacionados con las ventanas como por ejemplo abrir nuevas ventanas al presionar un botón.
- Cada una de estas ventanas tiene un tamaño, posición y estilo diferente.
- Estas ventanas emergentes suelen tener un contenido dinámico.

# 6.- Gestión de las ventanas.

- Abrir y cerrar nuevas ventanas:
  - Es una operación muy común en las páginas web.
  - En algunas ocasiones se abren sin que el usuario haga algo.
  - HTML permite abrir nuevas ventanas pero no permite ningún control posterior sobre ellas.

# 6.- Gestión de las ventanas.

- Abrir y cerrar nuevas ventanas:
  - Con JavaScript es posible abrir una ventana vacía mediante el método `open()`:

```
nuevaVentana = window.open();
```

De este modo la variable llamada `nuevaVentana` contendrá una referencia a la ventana creada.

# 6.- Gestión de las ventanas.

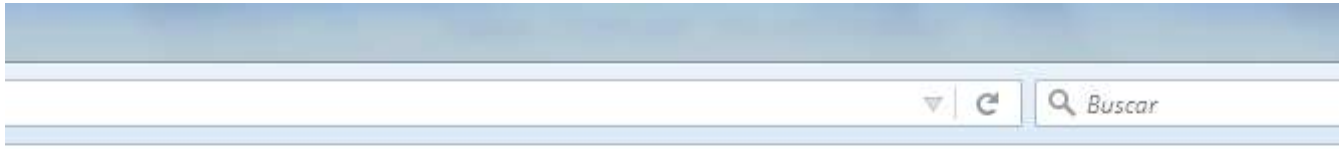
- Abrir y cerrar nuevas ventanas:
  - El método `open()` cuenta con tres parámetros:
    - URL.
    - Nombre de la ventana.
    - Colección de atributos que definen la apariencia de la ventana.
  - Ejemplo:

```
nuevaVentana = window.open("http://www.misitioWeb.com/ads",
"Publicidad", "height=100, width=100");
```

# Actividad 8

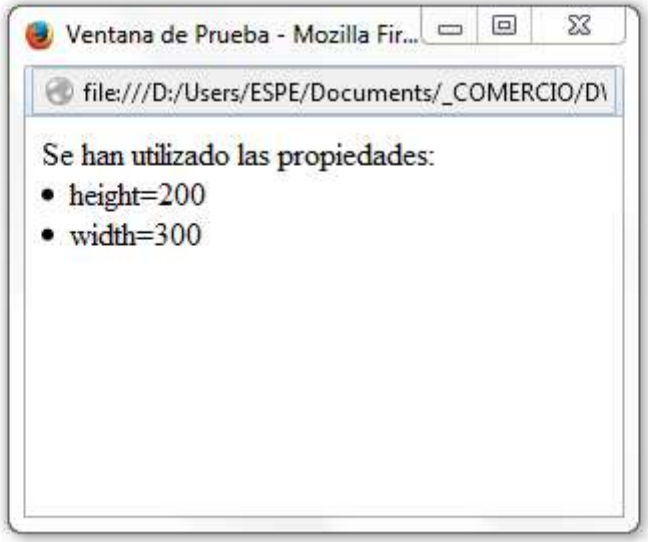
Crea un botón en una pagina web que abra una nueva ventana al hacer clic sobre él. En la nueva ventana, establecemos los atributos de altura y anchura, además de crear etiquetas HTML para generar el título de la ventana y un texto en el que especifiquemos las propiedades modificadas.

# Actividad 8




**Ejemplo de Apariencia de una Ventana**

Abre una Ventana



# 6.- Gestión de las ventanas.

- Para cerrar una ventana se puede invocar el método close():

```
myWindow1.document.write('<input type=button
value=Cerrar onClick=window.close()>');
```

Realiza esta modificación a la Actividad 8 para añadir un botón de cerrar en la ventana nueva.

# Actividad 9

Apertura de múltiples ventanas:

Crea un botón en una página web que abra cinco ventanas a la vez y que presenten a su vez un botón para poder cerrarlas.

# 6.- Gestión de las ventanas.

- Apariencia de las ventanas:
  - Las ventanas cuentan con propiedades que permiten decidir su tamaño, ubicación o los elementos que contendrá.

| Propiedad | Descripción |
|---|---|
| directories | Corresponde a los botones del directorio estándar del navegador. |
| height | Corresponde a la altura de la ventana. |
| menubar | Corresponde a la barra del menú. |
| resizable | Corresponde a la opción de cambiar o no el tamaño de la ventana. |
| scrollbar | Corresponde a las barras de desplazamiento. |
| status | Corresponde a la barra de estado. |
| toolbar | Corresponde a la barra de herramientas. |
| width | Corresponde a la anchura de la ventana. |

# 6.- Gestión de las ventanas.

- Comunicación entre ventanas:
  - Desde una ventana se pueden abrir o cerrar nuevas ventanas.
  - La primera se denomina ventana principal, mientras que las segundas se denominan ventanas secundarias.
  - Desde la ventana principal se puede acceder a las ventanas secundarias.

# 6.- Gestión de las ventanas.

- Comunicación entre ventanas:
  - En el siguiente ejemplo se muestra cómo acceder a una ventana secundaria:

```
<html><head></head><body>
  <script type="text/javascript">
   var ventanaSecundaria = window.open("", "ventanaSec", "width=500,
   height=500");
  </script>
  <center><h1> Comunicaci&oacute;n entre ventanas </h1><br>
  <form name=formulario>
   <input type=text name=url size=50 value="http://www.">
   <input type=button value="Mostrar URL en ventana secundaria"
  onclick="ventanaSecundaria.location = document.formulario.url.value;">
  </form></center></body></html>
```

# 6.- Gestión de las ventanas.

- Comunicación entre ventanas:
  - En el ejemplo anterior, se está enviando la información de la ventana principal a la ventana secundaria.
  - Cuando necesitemos hacer referencia al objeto window que haya abierto una ventana nueva, es decir, a la ventana padre, utilizaremos la propiedad opener del objeto window.

# Actividad 10

Modifica el ejemplo anterior para que la comunicación entre las dos ventanas sea bidireccional. Es decir, que la url que se escriba en la ventana principal, se muestre en la ventana secundaria y viceversa.