JAVASCRIPT

LENGUAJE DE OBJETOS

OBJETOS

- Objetos en la vida real: Coches, lavadoras, mesas
- Estos objetos tienen:
  - Carácterísticas (en JS se denomina propiedad)
    - Ej vida real Lavadora: un alto, un ancho, un color
    - Ej un botón de formulario: ancho, alto, color
  - Capacidades(en js se denomina método)
    - Ej lavadora:lava, centrifuga….
    - El botón de formulario: Clicar, seleccionar….

CONCLUSIÖN: Cómo JS es un lenguaje de programación Orientada a Objetos. Desde JS podremos modificar la propiedad color, ancho….

¿Cómo podremos conseguir dar propiedades a los objetos?
Mediante la nomenclatura del punto

Ejemplo, para cambiar la propiedad de un objeto mediante la nomenclatura del punto, la sintaxis sería

- Se utiliza la jerarquía y el operador punto
  - Ejemplos:
    - document.write();
    - window.alert();
    - boton.style.width="500px";
    - boton.style.backgroundColor="red";
    - boton.focus();

SINTAXIS para cambiar la propiedad de un objeto

Nombredelobjeto.propiedad=valor (entre comillas)

SINTAXIS para cambiar el método de un objeto

Nombredelobjeto.metodo();

Ojo!! todos los métodos llevan paréntesis también llamado zona de parámetro

Los puntos verdes son propiedades, y los rojos los métodos

EJEMPLO: Aplicar la nomenclatura del punto a un botón

Llevando esto, por ejemplo, al botón de JS, si quisieramos cambiar el color.

El objeto sería el **nombredelboton**

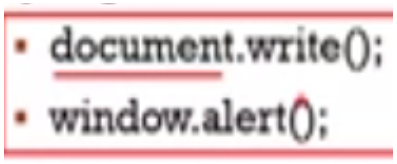
La propiedad para cambiar el color **style.backgroundColor=”red”**

Quedaría: **nombredelboton.style.backgroundColor=”red”**

En caso de querer dar un método a dicho botón, ejemplo que coja el foco

Quedaría: **nombredelboton.focus();**

Analicemos dos elementos de Programación orientada a Objetos, que hemos estado utilizando



- La instrucción document.write:
  OBJETO: DOCUMENTO
  MÉTODO WRITE()
  Nosotros cuando hacemos una página web estamos trabajando en **el documento**
  EL documento hace referencia a la página web. Y ese objeto, es decir nuestra página web, también tiene sus propiedades y métodos
  Uno de las capacidades, o método que tiene ese documento, es escribir (write())
- instrucción windows.alert()
  hemos estado utilizando el alert() sin poner el objeto delante.
  ¿por qué?
  Pues porque se trata de un método (alert), que pertenece a un objeto (windows) que está presente siempre. Nosotros cuando programamos en JS estamos programando para una ventana, luego siempre está presente, por eso lo podemos omitir
  Lo mismo pasa con prompt()

EJERCICIO 1 (CARPETA EJERCICIOS COMENTADOS>16.OBJETOS)

- VAMOS A APLICAR EN ESTE EJERCICIO EL MÉTODO getElementById, del objeto document al OBJETO document

3 ELEMENTOS DE LOS OBJETOS. PROPIEDADES MÉTODOS Y CONSTRUCTORES

- **Métodos**
  Son las funciones o acciones propias de cada objeto
  se podría decir que es un mensaje que recibe el objeto para realizar una acción previamente implementada.
  Sintaxis
  nombreObjeto.metodo(parámetros)
  Ej. window.reload() (hace que se recargue la página actual)
- **Constructores**
  un constructor es un método que se utiliza para crear e inicializar un objeto en tiempo de ejecución. Hay ciertos objetos en los que el constructor ya se invoca por defecto nada más cargar nuestra aplicación web, mientras que, en otros objetos si es necesario invocar a su constructor.
  SINTAXIS
  instanciaObjeto=newObjeto(parámetros)
  ejemplo **fecha=new Date()** creará un objeto tipo fecha con el valor de la fecha actual

OBJETOS PREDEFINIDOS

- Objeto Window
- Objeto document
- Objeto location
- Objeto screen
- Objeto navigator
- Objeto history
- Objeto date
- Objeto string

OBJETOS PREDEFINIDOS.

El objeto **window**

- Objeto definido más importante en JS
- Función: gestionar, administrar y proveer cualquier tipo de información acerca de la ventana que contiene la página actual
- En este objeto no hace falta invocar el constructor. JS ya lo hace por defecto al inicio de la página web

OBJETOS PREDEFINIDOS.

El objeto **window. Propiedades**

- **closed**: propiedad de tipo booleano que indica si una ventana ha sido cerrada o no
- **defaulStatus:** Define el texto que se mostrará en la barra de estado del navegador.
  Ejemplo para modificar el texto de la barra de estado:
  window.defaultStatus="Manual Imprescindible de HTML, CSS y JS"
- **history:** objeto que representa los enlaces a las páginas que el usuario ha visto con anterioridad. (mas adelante la tratamos con profundidad)
- **location:** Objeto que contiene y gestiona toda la información de la URL
- **name**: Propiedad que contiene el nombre de la ventana actual
- **status:** Valor de la barra de estado de la ventana actual
- **parent:** Hace referencia a la ventana padre de la ventana actual. Es realmente otro objeto de tipo window
- **self:** hacer referencia a la propia ventana, es similar a usar window
- **top:** ventana superior del navegador

OBJETOS PREDEFINIDOS.

El objeto window. Métodos

- **Open(url,nombre,opciones):**
  Abre una ventana emergente con la **url** indicada
  nombre que le daremos a la ventana
  opciones o configuración de nuestra ventana. Algunas de la opciones más comunes:
  - **height**: define el alto de la ventana en píxeles
  - **Width**: Ancho de la ventana en pixeles
  - **scrollbars:** Especifica si se muestran las barras de desplazamiento o no. Posibles valores: yes/no
  - **resizable:** Define si a nuestra ventana emergente se le puede modificar el tamaño con el uso del ratón. Posibles valores: yes/no
  - **status:** Dependiendo del valor muestra u oculta la barra de estado de la ventana en cuestión. Posibles valores: yes/no
  - **menubar:** especifica si se muestra o no la barra de menús**.** Posibles valores: yes/no
  - **toolbar:** Especifica si se muestra o no la barra de herramientas de la ventana. Posibles valores: yes/no

OBJETOS PREDEFINIDOS.

El objeto **window. Métodos**

- **Ejemplo del método open: Se creará una ventana emergente para abrir la página web Google.**
  Esta ventana tendrá las siguientes características:
  - nombre Google,
  - altura 600px,
  - anchura:600px,
  - barra de herramientas deshabilitada,
  - barra de menús deshabilitada y
  - desactivada la posibilidad de modificar su tamaño de forma manual

```
Window.open("http://www.google.es","google","heigth=600px,
width=600px, menubar=no, toolbar=no, resizable=no");
```

OBJETOS PREDEFINIDOS.

El objeto **window. Métodos**

- **close()** Cierra la ventana actual. Ejemplo. Window.close
- **blur().** Este método insta a la ventana actual a perder el foco. Ej. Window.blur ()
- **focus()** El contrario de el anterior. Este método hace que la ventana actual sea la que se encuentre activa. Lo que se conoce como obtener el foco. Ejemplo: window.focus()
- **alert(mensaje)**Muestra en pantalla el mensaje pasado por parámetro
- **confirm(mensaje),** también muestra un mensaje por pantalla, pero en esta ocasión, el usuario deberá elegir entre dos opciones Aceptar o Cancelar , dependiendo de la opción seleccionada, este método devolverá true o false Ejercicio 19 no comentado (CARPETA COMENTADOS)

OBJETOS PREDEFINIDOS.

El objeto **window. Métodos**

- **Prompt (mensaje, valor_por defecto)**
- método que se utiliza para solicitar un dato al usuario mediante un cuadro de diálogo.
- La información introducida en el cuadro de texto será el valor que retoque nuestro método.
- El primer parámetro, mensaje, es el mensaje que mostrará la ventana emergente y el segundo parámetro el valor por defecto del cuadro de texto
- EJ Ejercicio 20. No comentado. Método prompt

Como se puede observar, en los ejemplos de alert, confirm y prompt, cualquier propiedad o método del objeto window puede referenciarse directamente sin necesidad de indicar previamente el nombre del objeto. Es decir tendría el mismo efecto usar window.open(parámetros) que directamente open (parámetros), al tratarse de la ventana actual no es necesario hacer referencia al objeto window

OBJETOS PREDEFINIDOS.

El objeto **document.** Propiedades

- Función: administrar el contenido y los elementos presentes en nuestra web.
- Como la mayoría de los objetos de JS, depende del objeto Window
- PROPIEDADES:
  - **bgcolor** Especifica el color de fondo de nuestra WEB, EJ. Document.bgcolor=red,
  - **Fgcolor** el color de texto
  - **Linkcolor** Color de los enlaces o hipervínculos
  - **vlinkcolor** alinkcolor color de enlaces visitados y activos
  - **Images** Array con todas las imágenes presentes en la página web actual. Cada posición del array hace referencia a una imagen. Ej document.emages[0] haría referencia a la primera imagen de nuestro documento y con document.images[0].src accederíamos al atributo src de esta primera imagen.
  - **Links** Array que contiene todos los enlaces definidos en la página web
  - **Forms** Array con todos los formularios de nuestra web (se estudiará en más profundidad)
  - **Title.** Especifica el titulo de la web. Ej:document.title="john Wayne"

OBJETOS PREDEFINIDOS.

El objeto **document. Métodos**

- **Write(text)**Muestra en pantalla el teto pasado por parámetro. Este texto puede contener código HTML que será interpretado por JS. Ej:document.write (“<h2>Encabezado H2</h2>”)
- **writeIn(texto)**: Idéntico uso y sintaxis que la anterior propiedad. Pero añade una línea en blanco de forma automática después de escribir el texto pasado por parámetro

OBJETOS PREDEFINIDOS.

El objeto location. Propiedades

- Función: gestionar y consultar toda la información relativa a la URL de la web actual
- Como la mayoría de los objetos de JS, depende del objeto Window
- PROPIEDADES:
  - **host** Nos indica el nombre del servidor o del dominio y el puerto donde esta corriendo la pagina web, por ejemplo :www.miservidor.com:80
    ejercicio comentado 21
  - **hostname** propiedad idéntica a la anterior pero en este caso, únicamente se mostrará el nombre del servidor o dominio, como por ejemplo www.miservidor.com
  - **port** Puerto del servidor web. Uso: location.port
  - **protoco1** Protocolo usado por la URL de la página web, normalmente http o https
  - **href** Propiedad que especifica la URL de la pagina actual, es una propiedad de lectura/escritura. Esto significa que podremos hacer una redirección a otra web simplemente modificando el valor de esta propiedad. Ejercicio comentado 22
  - **patname** Retorna la parte de la URL que hace referencia la recurso del servidor web.
    Ej, para la URL: www.miservidor.com/admin/index.html esta propiedad retornaría el valor :admin/index.html

OBJETOS PREDEFINIDOS.

El objeto location. Métodos

- **Reload** (). Método que fuerza la recarga de la página actual. Su uso es: Location.reload()
- **Replace** (URL): Carga en la página actual la URL especificada en el parámetro de entrada. Es un mecanismo similar al que se consigue modificando la propiedad href, con la diferencia de que,, este caso, la entrada en el historial de páginas visitadas es reemplazada por la nueva URL

OBJETOS PREDEFINIDOS.

El objeto screen. Propiedades

- Función: proporcionar información útil acerca del dispositivo o pantalla de usuario
- También desciende directamente del objeto window, únicamente dispone de cinco propiedades, no implemente ningún método
- PROPIEDADES:
  - **availHeight** alto de la pantalla en px disponible para el navegador. **Sintaxis: screen. availHeight**
  - **availWidth** ancho de pantalla en pixeles disponible para el navegador**. Sintaxis: screen. availWidth**
  - **width**: Anchura total en píxeles de la pantalla de usuario**. Sintaxis: screen.width**
  - **height:** Altura total en pixeles de la pantalla de usuario **Sintaxis: screen.height**
  - **colorDepth:** Numero de bits por pixeles utilizados para representar la profundidad de colores de la pantalla**. Sintaxis: screen. colorDepth**

OBJETOS PREDEFINIDOS.

El objeto navigator. Propiedades

- Función: suministra toda la información disponible sobre el navegador que está mostrando la página Web.
  Este objeto solo tiene implementado un método, la mayor parte de la información útil está presente en sus propiedades
- PROPIEDADES
  - **appCodeName.** Especifica el nombre del código del navegador. No es muy útil porque todos los navegadores devuelven el mismo: Mozilla
  - **appName** Nombre o marca comercial del navegador
  - **appVersion.** Cadena de texto que muestra la versión completa de nuestro navegador
  - **Language** retoma la cadena de texto o código representativo del lenguaje del navegador Web
  - **Platform** describe el sistema operativo sobre el que está funcionando nuestro navegador
  - **Plugins** retorna un array o vector con todos los plugins instalados en nuestro navegador
  - **mimeTypes** retorna un array con todos los tipos MIME que soporta nuestro navegador

OBJETOS PREDEFINIDOS.

El objeto navigator. Propiedades

- PROPIEDADES
  - **userAgent** la más utilizada del objeto navegador, devuelve una cadena de texto con información completa del navegador web: código interno ,versión y nombre.
    Esta cadena se corresponde con la información que es enviada por el cliente al navegador en la cabecera HTTP

  - **cookieEnabled** Propiedad de tipo booleano que retorna el true si el navegador tiene activado las cookies y false en caso contrario

OBJETOS PREDEFINIDOS.

El objeto navigator. Método

- PROPIEDADES
  - **javaEnabled()** utilizado para determinar si el navegador en cuestión tiene activado el plugin de Java, retornará true o false dependiendo de si está activado o no

  - **CARPETA COMENTADOS.ejercicio 22 . ejemplo de uso de la propiedades y métodos del objeto navigator**

OBJETOS PREDEFINIDOS.

El objeto history. Propiedades y métodos

- Función: gestiona todas las direcciones de internet o URL que hemos visitado y que se almacenan en el historial del navegador. Básicamente lo que hace este objeto es almacenar la referencia a las páginas web visitadas para posteriormente poder recorrer las páginas web visitadas hacia delante o hacia atrás
- Descendiente de window

  - PROPIEDADES:
    - **Length**. Número entero que representa la cantidad de entradas almacenadas actualmente en el historial. Uso: history.length
  - METODOS
    - Back() para que se cargue en el navegador la página que se ha visitado justo anteriormente
      uso history.back()
    - fordward() al contrario de la anterior
    - go(numero): Este método nos lleva directamente a la web posicionada en el número indicado por el parámetro de entrada orden del historial del navegador.
      Ej history.back(-2) nos mandaría a la penúltima página visitada
      history.back(-1) nos mandará a la última web visitada ….

OBJETOS PREDEFINIDOS.

El objeto Date.

- Función: Permite manipular fechas y horas.
- A diferencia de los anteriores objetos, DATE no depende ni hereda ningún método o propiedad del objeto window.
- Tenemos que tener en cuenta:
  1. **JS no maneja fechas anteriores al 1 de enero de 1970**
  2. **El primer día de la semana es el domingo y tiene como valor 1, de esta forma el lunes será el 2, el martes 3 ….**
  3. **Los meses del año se contabilizan del o al 11, siendo enero el mes 0**
  4. **Las horas se representan siguiendo el formato HH:MM:SS**
  5. **JS gestiona y contabiliza las fechas en milisegundos**
  6. **Este objeto no tiene constructor por defecto, es necesario invocarlo e inicializarlo antes de comenzar a trabajar con el**

OBJETOS PREDEFINIDOS.

El objeto Date. EL CONSTRUCTOR DE DATE

- **El constructor de DATE, se puede invocar de 2 formas diferentes:**

1. Creando un objeto nuevo con la fecha actual
   SINTAXIS
   var fecha new Date(). //para crear e inicializar el objeto Date con la fecha actual del sistema
2. Creando el objeto con una fecha en concreto
   SINTAXIS
3. Var fecha new Date (año,mes, día, horas, minutos, segundos):
   //Constructor para inicializar un objeto date con una fecha en concreto

EJERCICIO CARPETA COMENTADOS.23.comentado.Objeto Date

OBJETOS PREDEFINIDOS.

El objeto Date. MÉTODOS

- **getDate()**retoma el día del mes de la fecha creada en el constructor
- **getDay()**devuelve un numero entero que representa el día de la semana de la fecha en cuestión
- **getHours()**retorna un entero entre o y 23 que representa la hora de la fecha creada.
- **getMinutes().**entero de 0 a 59 que representa los minutos de la hora en cuestión
- **getSeconds()** devuelve los segundos transcurridos en el minuto actual..Valor entre 0 y 59
- **getTime()** Método que retorna los milisegundos transcurridos desde el 1 de enero de 1970 hasta ahora
- **getMonth()**retorna un entero entre 0 y 11 que representa el mes de la fecha creada por el constructor
- **getYear()**Año de la fecha actual
- **toLocaleString()** retorna una cadena de texto con fecha actual y con el formato de la zona horaria especificada en el sistema

OBJETOS PREDEFINIDOS.

El objeto Date. MÉTODOS

- **toGMTString()** devuelve una cadena con la fecha y hora completa en formato o estándar GMT
- **setDate(diaMes)** Establece el día del mes pasado por parámetro en el objeto Date actual
- **setDay(diaSemana)** Define el día de la semana pasado por parámetro, en el objeto Date actual
- **setHours(horas)** Especifica la hora del día pasada por parámetro en la fecha creada por el objeto actual
- **setMinutes(minutos)** Define los minutos pasado por parámetro de la hora actual en el objeto Date
- **setMonth(mes)** Define el mes pasado por parámetro en la fecha definida por el objeto Date actual
- **setSeconds(segundos)**Establece los segundos pasados por parámetro en la fecha creada por el objeto Date
- **setYear(año)** Establece el año pasados por parámetro en la fecha actual
- **setTime(milisegundos).** Modifica la fecha actual con la fecha en milisegundos pasada por parámetro. Estos milisegundo se corresponderías con los milisegundos transcurridos desde el 1 de enero de 1970

OBJETOS PREDEFINIDOS.

El objeto String. Propiedades

- Uno de los objetos más utilizados y solicitados en JS
- Función: manipulación y administración de las cadenas de texto
- No es necesario invocar ningún constructor para este objeto, simplemente habría que declarar e inicializar una variable de tipo cadena antes de poder usar todos los métodos y propiedades disponibles
- PROPIEDADES
  - Length: retorna la longitud o número de caracteres de una cadena de texto

Ejemplo: Ejercicio de CARPETA COMENTADOS: 24comentado_length

OBJETOS PREDEFINIDOS.

El objeto String. Métodos

- **indexOf(cadenabusqueda, inicio)** método muy útil que nos devuelve la posición de la primera coincidencia de la cadena pasada por parámetro (**cadenaBusqueda**) dentro de la cadena actual. La búsqueda de esta coincidencia se hará a partir de la posición indicada por el parámetro **inicio**. Si se omite este parámetro, la búsqueda dará comienzo por la primera letra de la cadena. Si la cadena de búsqueda no existe, este método nos retornará el valor -1
- **lastIndexOf(cadena_buscada,indice)**: igual que la anterior, pero en ese caso el valor retronado se corresponde con la posición de la última ocurrencia de la cadena de búsqueda.
- **substring(inicio,fin):** devuelve la cadena de texto que se encuentra delimitada entre las posiciones indicadas por los parámetros de inicio y fin
- **link(url)**:Método que convierte la cadena de texto en un enlace que apuntara a la URL pasada por parámetro
- **toLowerCase():** Convierte a minúsculas la cadena de texto actual
- **toUpperCase():** convierte a mayúsculas la cadena de texto actual
- **Big():** Método que muestra la cadena de texto con un tamaño de fuente de gran tamaño
- **small()** Muestra la cadena de texto con tamaño de fuente muy pequeño

OBJETOS PREDEFINIDOS.

El objeto String. Métodos

- **blink()** asigna a la cadena de texto un efecto de parpadeo
- **Italics()** muestra el texto en cursiva
- **strike()** asigna a la cadena un efecto de tachado
- **sup()** muestra la cadena de texto con un formato de superíndice
- **sub()** muestra la cadena de texto con un formato de subíndice
- **charAt(indice)** Método que retoma el carácter situado en la posición indicada por el parámetro índice de la cadena actual
- **fontcolor(color)** modifica el color de la cadena de texto por el color pasado por parámetro
- **fontsize(tamaño):** Modifica el tamaño de la cadena de texto por el tamaño pasado por parámetro. Los valores permitidos para el parámetro de entrada oscilarán entre 1 y 7, siendo 1 el tamaño más pequeño y 7 el más grande

CARPETA COMENTADOS: Ejercicio 25. String.Ejemplo 2

OBJETOS PREDEFINIDOS.

El objeto Array

- **Los Arrays** también conocidos como vectores, son estructuras de datos lineales que almacenan la información en posiciones continuas, pudiendo se almacena r en cada posición del array cualquier tipo de dato
- Dos formas de crear array en JS
- **Var vector=new Array()** : Para crear un array vacio sin datos y sin tamaño determinado
  Se trata de declarar una variable vector y asignarle una nueva isntancia del objeto Array usando el constructor del mismo sin ningun parametro
- **Var vector=new Array(5)** : Similar al método anterior, pero en este caso hemos creado un array vacio de 5 posiciones
- **Var vector=new Array("carlos", "perez", 45)** : para crear e inicializar un array en una sola instrucción. El constructuro aquí tiene tres parámetros . Caa parámetro se corresponde con un dato almacenado en una posición de array.

  Se pueden especificar tantos parámetros de entrada como elementos almacenara nuestro vector, estos datos separados por comas irán ocupando posiciones del Array de forma continual

OBJETOS PREDEFINIDOS.

El objeto Array

- Las posiciones de un vector están nuemeradas por unos indices de 0 a N1, siendo N el número de poxiciones que tendrá nuestro Array.
- Teniendo esto en cuenta para acceder a un dato o una posición en contreto del vector, usaremos la expresión nombreVector(n), donde n será la posición solicitada
- Ej

```
Var vector= new Array ("carlos","perez",45)//inicializamos el vector
var nombre=vector [0]; //asignamos a la variable nombre la primera posición del
                       array : carlos
var apellido= vector [1]; //la variable apellido tendrá el valor perez
vector[2]=50; //modificamos el valor de la tercera posicion del vector

Var vectpr2=new Array(); //creamos un vector vacio y a continuacion
vector2[0]="pedro";          inicializamos las posiciones del vector
vector2[1]="león";
vector2[2]="20"
```

OBJETOS PREDEFINIDOS.

El objeto Array

//VAMOS A INSERTAR NUEVOS DATOS EN LOS VECTORES

VECTOR [3]="VECINO"

VECTOR2 [3]=" AMIGO"

Conclusion el acceso a la informacion almacenada en una rray no tiene comñicacion, todo consiste en acceder directamente a las diferente pposiciones del vector o array.

EL problema puede venir si el vector es grande, para lo cual la solucion es acudri al bucle for para poder recorrer todas las posiciones de forma sencilla

ej

```
Var vector2=new Array (); //creamos el array vacio
vector2[0]="pedro"        //a continuación inicializamos las posiciones del vector
vector2[1]="león";
vector2[2]="20"
   For(var i=0;i<6; i++)
   document.write (vector[i]+"<br/>");
```

el bucler for recorrerá de forma automática las tres posiciones del vector

OBJETOS PREDEFINIDOS.

El objeto Array

- Matrices

Cuando los array son bidimensionales se les llama Matriz, que no es otra cosa que un vector de vectores en el que cada posicion del Array llevara definido otro Array.
Son estructuras de datos muy útiles para almacenar grupos de información que requieran una organización dividida en filas y columnas

La manera de acceder a la informaciópn alojada en este tipo de estructuras es nombreMatriz[fila][columna]

El siguente código muestra un ejemplo de cómo crer una matriz de 3x4, es decir de tres filas y cuatro columnas

OBJETOS PREDEFINIDOS.

El objeto Array

Ejemplo matriz

```
Var matrix=new Array (4); //creamos el array vaciode 4 posiciones //
en cada posicon del aarray nos definimos e iniciializamos un vector
matriz[0]=new array( “pedro” , “leon”, “20”, “decorador”);
matriz[1]=new array( “carmen” , “perez”, “30”, “medico”);
matriz[2]=new array( “cristina” , “rodriguez”, “20”, “comercial”);
matriz[3]=new array( “Paloma” , “diaz”, “20”, “comercial”);
```

```
Var almuno= matriz[0][0];// nombre del primer alumno
Var apellido= matriz[0][1];//apellido del primer alumno
```

OBJETOS PREDEFINIDOS.

El objeto Array

Si para recorrerse todas las posiciones de un vector o Array echabamos mano de un bucle for, para hacer lo mismo con una matriz, tendremos que hacer un bucle for anidado

Siguiendo con el elemmplo de la matriz anterior, mostramos a continuacion un ejemplo de bucle for anidado

```
For (var i=0;i<5;i++){
For (var j=0; j<4; j++)
Document.write(matriz[i] [j] +<br/>)}
```

OBJETOS PREDEFINIDOS.

El objeto Array

PROPIEDADES DEL OBJETO ARRAY

Lenght devuelve el total de elemtneos de un array

METODOS DEL ARRAY

Concat(array): concatena el array actual con el array pasado por parámetro. Resultado un nuevo vector producto de unir los elementos de ambos arrays

Ej

```
var vector1[=new array( “pedro”  , “leon”, “20”, “decorador”);
var vector2[1]=new array( “carmen”  , “perez”, “30”, “medico”);
//unimos el vecto r1 con el 2 y almacenamos el resultado en vectorFInal

Var vectorFinal=vector1.concat(vector2);
```

OBJETOS PREDEFINIDOS.

El objeto Array

Join(separador): devuelve una cadena de texto con el contenido de todas las posiciones de un array separadas por el carácter pasado por parámetro. Si se omite el carácter pasado por parámetetro, estarán separadas por comas.

Ej

Reverse():Invierte el orden de los elementos almacenados en un vector

Sort(). Oredenas los datos del array de menor a mayor o alfabéticamente

OBJETOS PREDEFINIDOS.

El objeto Array

PROPIEDADES DEL OBJETO ARRAY

Lenght devuelve el total de elemtneos de un array

METODOS DEL ARRAY

Concat(array): concatena el array actual con el array pasado por parámetro. Resultado un nuevo vector producto de unir los elementos de ambos arrays

Ej

Join(separador): devuelve una cadena de texto con el contenido de todas las posiciones de un array separadas por el carácter pasado por parámetro. Si se omite el carácter pasado por parámetetro, estarán separadas por comas.

Reverse()

Sort()