

CONFECCION Y PUBLICACIÓN DE PÁGINAS WEB. MODULO 2

MODULO 2:

MF0951_2: Integrar componentes software en páginas web

UF1305: Programación con lenguajes de guión en páginas web. (90 horas).

UF1306: Pruebas de funcionalidades y optimización de páginas web. (90 horas).

MF0951_2: Integrar componentes software en páginas web

UF1305: Programación con lenguajes de guión en páginas web. (90 horas).

OBJETOS

- Objetos y arrays: elementos más importantes del lenguaje JS

- OBJETOS en JS:

Elementos que contienen toda la información que permiten identificar a una entidad de forma completa

Disponen de mecanismos tales como los Métodos para interactuar con otros objetos.

- Se trata de simplificar la antigua programación orientada a procedimientos. Muy complicada a la hora de actualizar, agregar una nueva funcionalidad al programa, corregir...sobretudo cuando se trataba de que el programador que no era el que había creado el código.
- Por este motivo la Comunidad de Programación se vio en la necesidad de inventar algo que simplificara el trabajo de los programadores y así se inventó algo revolucionario: La llamada Programación Orientada a Objetos (POO), que consiste en asemejar los elementos de la vida real con el código de programación

OBJETOS

Teniendo en cuenta que la vida real está repleta de **objetos**: Coches, Lavadoras, mesas

Todos ellos tienen algo en común. Tienen **características y capacidades**

Ej

una lavadora

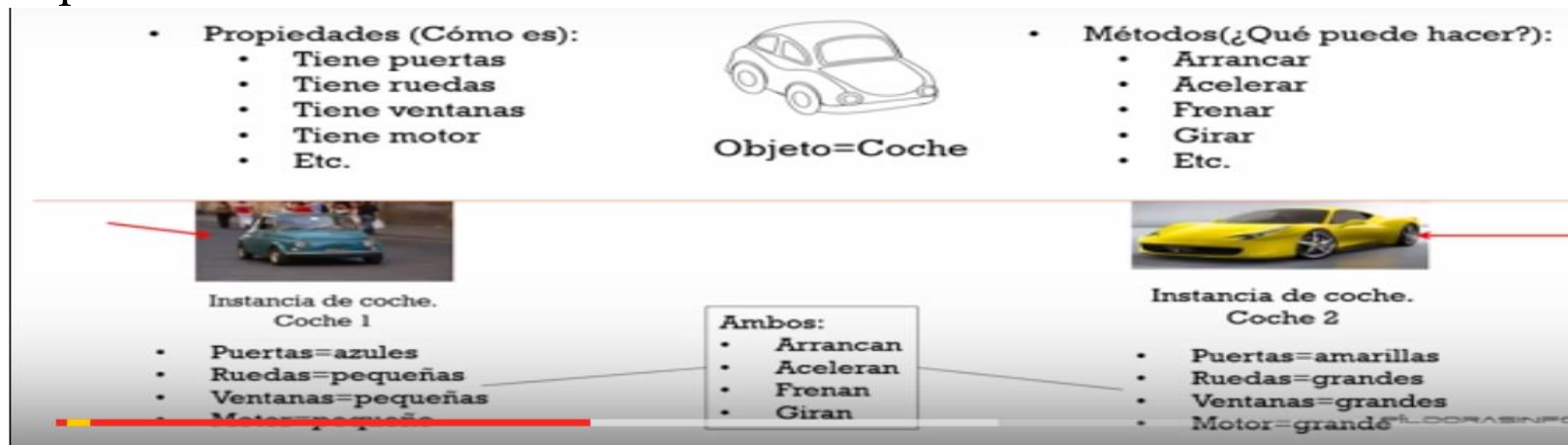
tiene las siguientes características: un alto, una ancho, un color.....

Tiene las siguiente capacidades: lava, centrifuga...

OBJETOS

Pues todos los objetos van a tener sus características(propiedades) y sus capacidades(métodos).

Esto se llevó al mundo de la programación cambiando el nombre, puesto que a las características de un objeto se le denomina **propiedad** y a las capacidades **Métodos**



Siguiendo con la comparativa, en el mundo real habrá muchos modelos de coche, caros baratos...grandes pequeños....a eso en programación se llama **Instancias**

OBJETOS

¿Como llevamos esto al mundo de la programación?

Sabemos que por ejemplo en JS podemos tener botones, ejemplo un botón de formulario.

Pues a este botón se le considera un objeto con sus propiedades (ancho, alto, color) y sus métodos(se le puede seleccionar, clicar...).

Esto quiere decir que como JS es un lenguaje de Programación O a Objetos. Desde JS podremos modificar la propiedad color, ancho...de un objeto botón

¿Y cómo hacemos esto? Como decimos desde JS a un botón que cambie el color, el tamaño...que active el foco, se seleccione, etc

Nomenclatura del punto

Esto se consigue mediante la nomenclatura del punto. Ejemplo, para cambiar la propiedad de un objeto 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 parentesis tambien llamado zona de parámetro

Nomenclatura del punto

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

El objeto sería el **nombre del boton**

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

Quedaría: **nombre del boton.style.backgroundColor="red"**

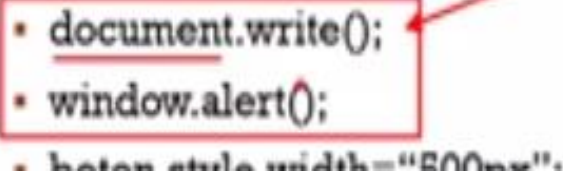
En caso de querer dar un método a dicho botón, ejemplo que coja el foco

Quedaría: **nombre del boton.focus();**

•

- Se utiliza la jerarquía y el operador punto

- Ejemplos:



- document.write();
- window.alert();
- button.style.width="500px";

En el caso de estos dos elementos, se trata de elementos de Programación orientada a objetos que hemos estado utilizando.

Por ejemplo con la instrucción `document.write` estábamos haciendo programación orientada a objetos

OBJETO: DOCUMENTO

MÉTODO `WRITE()`

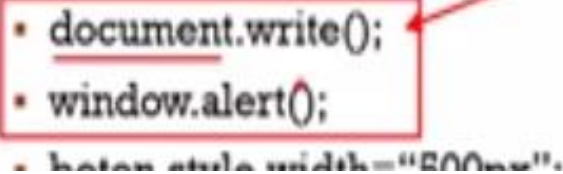
Nosotros cuando hacemos una página web estamos trabajando en el documento

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()`)

- Se utiliza la jerarquía y el operador punto

- Ejemplos:



- document.write();
- window.alert();
- boton.style.width="500px";

En el caso de windows.alert() lo hemos estado utilizando sin poner el objeto delante.

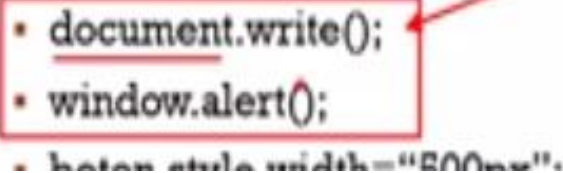
¿por qué?

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()

- Se utiliza la jerarquía y el operador punto

- Ejemplos:



- document.write();
- window.alert();
- button.style.width="500px";

En el caso de estos dos elementos, se trata de elementos de Programación orientada a objetos que hemos estado utilizando.

Por ejemplo con la instrucción `document.write` estábamos haciendo programación orientada a objetos

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()`)

Nomenclatura del punto

Esto se consigue mediante la nomenclatura del punto. Ejemplo, para cambiar la propiedad de un objeto 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 parentesis tambien llamado zona de parámetro

Ejercicio:

Creamos un botón

Le aplicamos un id

Después ponemos un script

Una vez declaramos la variable y la asignamos a un objeto, ponemos el punto para poner la propiedad o el método.

Los puntos verdes son propiedades, y los rojos los métodos

Acedemos a un método perteneciente al objeto document, que es el método getElementById, y ese método identifica todos los objetos que podamos tener en nuestra página, a través de su id

En concreto identificamos el id=boton1, para asignárselo a la variable miboton

Así podremos, a partir de ese momento, cambiar las propiedades, o los metodos a través de esa variable (en este caso la propiedad del ancho del botón)

Actualizamos el navegador y vemos que se ha aumentado el tamaño del botón

SOLUCIÓN EJERCICIO

```
1 <!doctype html>
2 <html>
3 <head>
4 <meta charset="utf-8">
5 <title>Documento sin título</title>
6 </head>
7
8 <body>
9
10 <input type="button" id="boton1">
11
12
13 <script>
14
15     var miboton=document.getElementById("boton1");
16
17     miboton.style.width="250px";
18
19 </script>
20
21 </body>
22 </html>
```

Añadimos ahora las propiedades alto y color al botón

```
hay errores de sintaxis.

<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Documento sin título</title>
</head>

<body>

<input type="button" id="boton1">

<script>

    var miboton=document.getElementById("boton1");

    miboton.style.width="300px";

    miboton.style.height="300px";

    miboton.style.backgroundColor="blue";

</script>

</body>
</html>
```

hay errores de sintaxis.

```
<input type="button" id="boton2">

<input type="text" id="micuadro">

<script>

    var miboton=document.getElementById("boton1");

    var miboton2=document.getElementById("boton2");

    var cuadroTexto=document.getElementById("micuadro");

    miboton.style.width="300px";

    miboton.style.height="300px";

    miboton2.style.width="300px";

    miboton2.style.height="300px";

    miboton2.focus();

    cuadroTexto.style.background="red";

    cuadroTexto.value="Escriba aqui....";

    cuadroTexto.style.height="200px";

    //miboton.style.backgroundColor="blue";

</script>

</body>
</html>
```

Ahora los métodos. En concreto el método focus()

Tiene la capacidad de cambiarse cuando está seleccionado o en foco

EJERCICIO

Creamos dos botones más y les damos las mismas propiedades a los dos primeros y al tercero será un cuadro de texto

Probamos cual tiene el foco (el que esté seleccionado)

OBJETOS

- En JS, un objeto se puede dividir básicamente en 3 elementos bien diferenciados, **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=new Objeto(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
- **defaultStatus:** Define el texto que se mostrará en la barra de estado del navegador.
Ejemplo para modificar el texto de la barra de estado:
`windo.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", "height=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 título de la web. Ej: `document.title="john Wayne"`

OBJETOS PREDEFINIDOS.

El objeto **document**. Métodos

- **Write(text)** Muestra en pantalla el texto pasado por parámetro. Este texto puede contener código HTML que será interpretado por JS. Ej: `document.write("<h2>Encabezado H2</h2>")`
- **writeln(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`
 - **protocol** 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
 - **pathname** Retorna la parte de la URL que hace referencia al 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, en 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 descende 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()`
 - `forward()` 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 0 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()** retorna 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 0 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
- **getFullYear()** 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