Objetos nativos: Date, Math, Number, Boolean, String

- Son objetos que no dependen del navegador
- Un objeto es una colección de **propiedades** (valores), **métodos** (funciones) y **eventos** (acciones)
- Para acceder a una propiedad o método, se utiliza el punto (.)
- Sintaxis para crear un objeto:

```
let miObjeto = new Objeto_de_JavaScript();
```

Objeto Date

- Constructores
  - let fecha = new Date();
  - let fecha2 = new Date(milisegundos);
  - let fecha3 = new Date(año, mes, dia, hora, minutos, segundos);
  - let fecha4 = new Date(año, mes, dia); // Se crea el objeto con hora 00:00:00

LOS MESES COMIENZAN POR 0: ENERO ES 0, FEBRERO 1, MARZO 2, ...

- Métodos
  - getDate(): Devuelve el día del mes (de 1 a 31).
  - getDay(): Devuelve el día de la semana (de 0 (domingo) a 6 (sábado))
  - getHours(): Devuelve la hora.
  - getMinutes(): Devuelve los minutos.
  - getMonth(): Devuelve el mes (Cuidado: el mes empieza por 0).
  - getSeconds(): Devuelve los segundos.
  - getFullYear(): Devuelve el año con todos los dígitos
  - getTime(): Devuelve la fecha en milisegundos desde el 1 de enero de 1970
  - setHours(): ajusta la hora (de 0 a 23)
  - setDate(): Actualiza el dia del mes.
  - setHours(): Actualiza la hora.
  - setMinutes():Cambia los minutos.
  - setMonth():Cambia el mes (Atención al mes que empieza por 0).
  - setSeconds():Cambia los segundos.
  - setTime():Actualiza la fecha completa. Recibe un numero de milisegundos desde el 1 de enero de 1970.
  - setFullYear(): Cambia el año de la fecha con el número que recibe por parámetro. El número se indica completo ej: 2005 o 1995
  - toString(): Devuelve una representación de la fecha donde día y mes están expresados en ingles. // Mon Dec 26 2050 08:30:00 GMT+0300 (Hora de verano de Rusia)

- toDateString(): Devuelve una representación abreviada de la fecha donde día y mes están expresados en ingles. // Mon Dec 26 2050
- toLocaleDateString(): (*Sin Parámetros) Devuelve una representación de la fecha de hoy (día, mes y año) según la forma de expresar fechas que venga determinada por el tiempo del sistema del computador donde se ejecute: // 26/12/2050
- toLocaleString(): Devuelve una representación de la fecha completa según la configuración local del computador donde se ejecute. // 26/12/2050 9:30:00
- toLocaleTimeString(): Devuelve la hora de la fecha // 9:30:00

Objeto Math

- Permite realizar operaciones matemáticas sin necesidad de librerías externas.
- No podemos crear objetos de tipo Math pero sí llamar a propiedades o métodos:
- Constantes:

Constante	Descripción	Valor
Math.E	Número de Euler	2.718281828459045
Math.LN2	Equivalente a Math.log(2)	0.6931471805599453
Math.LN10	Equivalente a Math.log(10)	2.302585092994046
Math.LOG2E	Equivalente a Math.log2(Math.E)	1.4426950408889634
Math.LOG10E	Equivalente aMath.log10(Math.E)	0.4342944819032518
Math.PI	Número Pi Π	3.141592653589793
Math.SQRT1_2	Equivalente a Math.sqrt(1/2).	0.7071067811865476
Math.SQRT2	Equivalente a Math.sqrt(2).	1.4142135623730951

  - ✔ Ejemplo
    - Math.PI // let pi = Math.PI;

- **Métodos matemáticos:**

Método	Descripción	Ejemplo
Math.abs(x)	Devuelve el valor absoluto de x.	|x|
Math.sign(x)	Devuelve el signo del número: 1 positivo, -1 negativo	
Math.exp(x)	Exponenciación. Devuelve el número e elevado ax.	e^x
Math.expm1(x)	Equivalente aMath.exp(x) - 1.	e^x-1
Math.max(a, b, c...)	Devuelve el número más grande de los indicados por parámetro.	
Math.min(a, b, c...)	Devuelve el número más pequeño de los indicados por parámetro.	
Math.pow(base, exp)	Potenciación. Devuelve el número base elevado a exp.	$base^{exp}$
Math.sqrt(x)	Devuelve la raíz cuadrada de x.	√x
Math.cbrt(x)	Devuelve la raíz cúbica de x.	$\sqrt[3]{x}$
Math.imul(a, b)	Equivalente a a * b, pero a nivel de bits.	
Math.clz32(x)	Devuelve el número de ceros a la izquierda de x en binario (32 bits).	

- **Métodos de redondeo**

Método	Descripción
Math.round(x)	Devuelve x con redondeo (**el entero más cercano**)
Math.ceil(x)	Devuelve x con redondeo superior (**el entero más alto**)
Math.floor(x)	Devuelve x con redondeo inferior (**el entero más bajo**)
Math.fround(x)	Devuelve x con redondeo (**flotante con precisión simple**)
Math.trunc(x)	Trunca el número x (**devuelve sólo la parte entera**)

✔ Ejemplos:

```
Math.round(3.75);  //4
Math.round(3.25);  //3
Math.ceil(3.75);      //4
Math.ceil(3.25)       //4
Math.floor(3.75);     //3
Math.floor(3.25);     //3
Math.round(3.123456789);   //3
Math.fround(3.123456789);  //3.1234567165374756
Math.trunc(3.75);      // 3
Math.round(-3.75);     //-4
Math.trunc(-3.75);     //-3
```

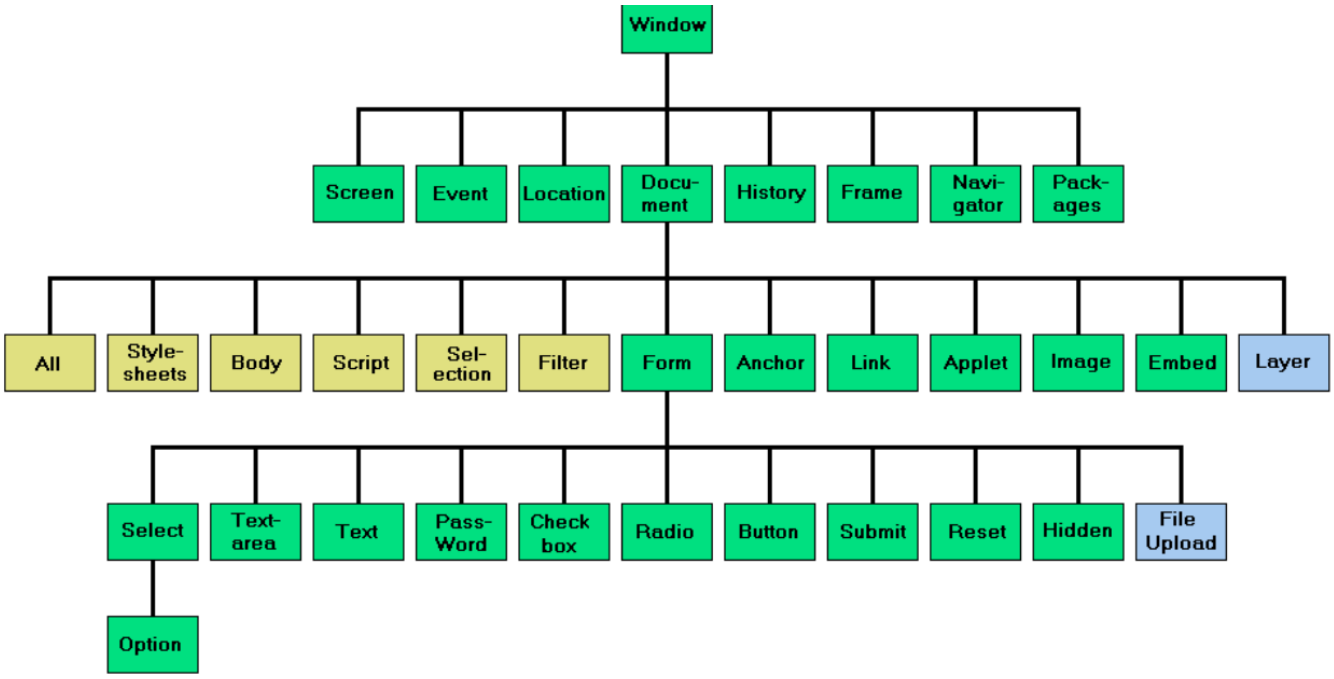
- **Método Math.random()**
  - Devuelve un número al azar entre 0 (inclusive) y 1(exclusivo) con 16 decimales.
  - Normalmente cuando queremos trabajar con números aleatorios lo que buscamos es obtener un número al azar entre 2 números. Se suele usar: Math.floor, o parseInt. El proceso es:
    - Ejemplo: Número al azar entre 0 y 5:

```
let x = Math.random();  // Devuelve un número al azar entre [0,1) con 16 decimales.
x = x* 5;               // Multiplicamos dicho número por el valor máximo que buscamos.
x = Math.floor(x);      // Redondeamos hacia abajo quedándonos solo con la parte entera.
```

    - let y = Math.floor(Math.random() * 5); // Número entre 0 y 5 (5 excluido)
  - ¿Cómo se obtiene un número aleatorio entre 6 y 10?
    - Math.floor(Math.random() * 5) + 6;
  - Si queremos obtener un número aleatorio entre dos números min y max (incluidos)
    - **Math.floor(Math.random() * (max – min + 1)) + min**
    - **parseInt(Math.random() * (max – min + 1)) + min**

Objeto Window

- Cuando se carga una página en el navegador, el intérprete de JavaScript crea automáticamente una serie de objetos que pueden ser usados por nuestros programas.
- Window es el objeto principal del cual "cuelgan" los demás como si fueran propiedades suyas. Se crea un objeto de este tipo para cada ventana que pueda ser lanzada desde una página Web .



- A partir del objeto window se pueden crear nuevos objetos window que serán nuevas ventanas. Éstas se pueden controlar desde la ventana padre. Además permite cerrar ventanas, actúa sobre los marcos, puede sacar mensajes (de error u otro tipo), confirmación y entrada de datos.
- **Apertura de una ventana**
  - Hay que tener en cuenta que los actuales navegadores tienen bastante restringida la apertura de ventanas desde Javascript por culpa de haber abusado para la publicidad mediante pop-ups, por lo que puede ser necesario tener que modificar la configuración del navegador para permitir las ventanas emergentes.
  - **let nuevaVentana = window.open("direccion URL", "nombre de la ventana", "parametros");**
    - ➔ dirección URL es la página que se va a cargar en la nueva ventana. Podemos abrir una ventana con una página existente o una página desde cero. En este último caso, la dirección va vacía y se crea el contenido de la misma utilizando el método write.
    - ➔ Nombre de la ventana: el nombre que se usará el target de los enlaces.
    - ➔ Parámetros: atributos de la ventana que son:

- *toolbar: si se muestra o no la barra de herramientas*
- *location: si se muestra o no la barra de direcciones*
- *status: si se muestra o no la barra de estado*
- *menubar: si se muestra o no la barra de menú*
- *scrollbars: si se muestran o no las barras de scroll*
- *resizable: si se puede redimensionar o no la página*
- *width: ancho en píxeles*
- *height: altura en píxeles*
- *top: la posición de la ventana desde su borde superior hasta la parte superior de la pantalla.*
- *left: la posición de la ventana desde su borde izquierdo hasta la parte izquierda de la pantalla.*

Estos atributos se separan mediante comas:

"toolbar=0,location=1,directories=0,resizable=0,width=200,height=100"

✔ Ejemplo:

let **nuevaVentana** = window.open("http://www.ceu.es","segundaPag", "toolbar=yes, height=200");

- Por lo tanto, desde el padre, una vez tenemos la referencia a la ventana secundaria, podemos acceder a ella utilizando su referencia:

  Si queremos escribir en la nueva ventana: **nuevaVentana.document.write**("<h1>Hola</h1>");

  Si queremos acceder al estilo de su body: **nuevaVentana.document.body.style**

- Si lo que queremos es desde la ventana secundaria, acceder a la ventana que abrió la ventana actual (la ventana 'padre') se utiliza **window.opener.document**

- **Cerrar ventanas**
  - Para cerrar una ventana se utiliza el método close(): variable_ventana.**close**();
    - Ejemplo: **nuevaVentana.close();**
  - Si la ventana a cerrar es la propia: window.**close**();
  - Para saber si una ventana está o no cerrada se utiliza: **closed**() sobre la ventana a consultar. Devuelve true si ya está abierta o false en caso contrario.

- **Propiedad innerHTML**
  - innerHTML es una propiedad en JavaScript que permite acceder o modificar el contenido HTML de un elemento de la página. Esta propiedad te permite obtener o establecer el código HTML que se encuentra dentro de un elemento específico.
    - Ejemplo: document.body.innerHTML += " <p> nuevo texto <p>";

- **Temporizadores**
  - Permite crear animaciones, publicidad, son fundamentales en JavaScript.
  - Un temporizador es una función que permite ejecuciones de tareas después de un determinado tiempo.
  - Se consigue con los métodos: **setTimeout y clearTimeout**
  - **Para lanzar un temporizador: setTimeout(instrucciones, tiempo);**
    - Instrucciones: acciones que se ejecutan cuando se cumpla un tiempo. Lo habitual es llamar una **función** que ejecute un conjunto de instrucciones.

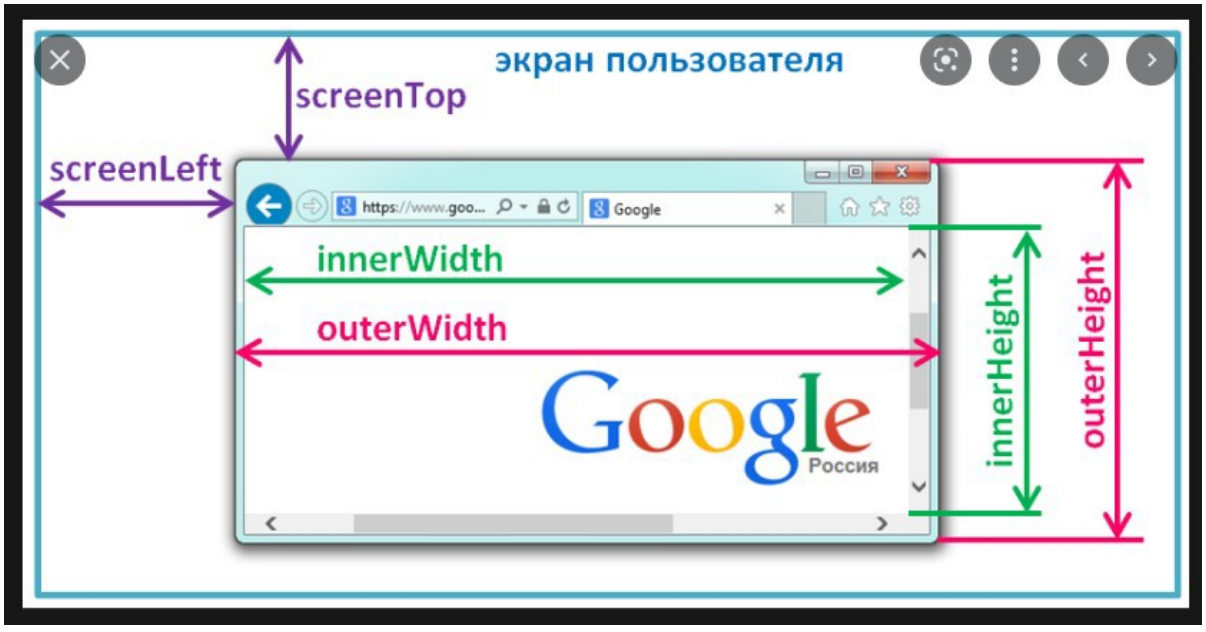
```
setTimeout( nombre_funcion, tiempo );
function nombre_funcion{
    instrucciones....
}
```

    - Tiempo: milisegundos que deben pasar para que se ejecuten las instrucciones.
    - Ejemplo: **¿Qué hace este temporizador?**

      let v_temporizador = setTimeout("alert('Buenas tardes'), 5000");

  - **Para anular un temporizador: clearTimeout(variable_temporizador)**
    - clearTimeout(v_temporizador);
  - Importante: el temporizador sólo se ejecuta una vez salvo que se invoque a otro temporizador o se use:
  - **SetInterval:** Genera un evento cada n milisegundos:
    - let temporizador = **setInterval**(instrucciones, tiempo);
  - **ClearInterval:** Cancela el temporizador lanzado con setInterval:
    - **clearInterval**(temporizador);

- **Propiedades del objeto window**
  - ***Closed***: Devuelve un valor Boolean indicando cuando una ventana ha sido cerrada o no.
  - ***DefaultStatus***: Ajusta o devuelve el valor por defecto de la barra de estado de una ventana.
  - ***Document***: Devuelve el objeto document para la ventana.
  - ***Frames***: Devuelve un array de todos los marcos (incluidos iframes) de la ventana actual.
  - ***History***: Devuelve el objeto history de la ventana.
  - ***Length***: Devuelve el numero de frames (incluyendo iframes) que hay en dentro de una ventana.
  - ***Location***: Devuelve la Localizacion del objeto ventana (URL del fichero).
  - ***Name***: Ajusta o devuelve el nombre de una ventana.
  - ***Navigator***: Devuelve el objeto navigator de una ventana.
  - ***Opener***: Devuelve la referencia a la ventana que abrio la ventana actual (con un window.open)
  - ***Parent***: Se refiere al padre de una ventana en un iframe
  - ***Self***: Devuelve la ventana actual.
  - ***Status***: Ajusta el texto de la barra de estado de una ventana.
  - ***InnerHeight***: Tamaño en pixels del espacio donde se visualiza la pagina, en vertical
  - ***InnerWidth***: Tamaño en pixels del espacio donde se visualiza la pagina, en horizontal
  - ***OuterHeight***: Representa el alto de toda la ventana, incluyendo la barra de notificaciones y bordes
  - ***OuterWidth***: Representa el ancho total de la ventana incluyendo las barras laterales
  - ***ScreenTop***: Números de píxeles desde el borde superior del navegador al de la pantalla.
  - ***ScreenLeft***: Números de píxeles desde el borde izquierdo del navegador al de la pantalla.
  - ***ScrollX***: Píxeles desde la izquierda a la barra de scroll horizontal
  - ***ScrollY***: Píxeles desde el borde superior a la barra de scroll vertical



- **Métodos del objeto window**

  - ***alert***(): Muestra una ventana emergente de alerta y un botón de aceptar.
  - ***blur***(): Elimina el foco de la ventana actual.
  - ***clearInterval***(): Anula el temporizador ajustado con setInterval().
  - ***setInterval***(): ejecuta unas acciones en un intervalo especificado (en milisegundos).
  - ***setTimeout***(): solo ejecuta una vez las acciones en un tiempo especificado en milisegundos.
  - ***close***(): Cierra la ventana actual.
  - ***confirm***(): Muestra una ventana emergente con un mensaje, un botón de aceptar y un botón de cancelar.
  - ***focus***(): Coloca el foco en la ventana actual.
  - ***open***(): Abre una nueva ventana de navegación.
  - ***prompt***(): Muestra una ventana de dialogo para introducir datos.
  - ***moveBy***(x,y): Mueve la ventana actual el número de píxeles especificados
  - ***moveTo***(x,y): Mueve la ventana a la posición especificada
  - ***scrollBy***(x,y): Mueve el scroll de la venta el número de píxeles indicados
  - ***scrollTo***(x,y): Mueve el scroll de la venta a la posición indicada
  - ***ResizeBy***(x,y): Redimensiona la página en el número de píxeles especificados. Ej: aumenta la ventana en 10 pixeles de ancho y largo.
  - ***ResizeTo***(x,y): Redimensiona la página a las posiciones indicadas. Ej: aumenta la ventana a 400x200

Objeto Navigator

- Guarda información sobre el navegador que se está utilizando.
  - ***appCodeName***: Mozilla
  - ***appName***: Netscape
  - ***appVersion***: 5.0 (X11; es-ES)
  - ***language***: es-ES
  - ***platform***: Linux x86_64
  - ***oscpu***: Linux x86_64
  - ***vendor***: Ubuntu
  - ***vendorSub***: 10.04
  - ***product***: Gecko
  - ***productSub***: 20110422
  - ***securityPolicy***:
  - ***userAgent***: Mozilla/5.0 (X11; U; Linux x86_64; es-ES; rv:1.9.2.17) Gecko/20110422 Ubuntu/10.04 (lucid) Firefox/3.6.17
  - ***cookieEnabled***: true

- ***onLine***: true
- ***buildID***: xxxxxxxxxxxx
- ***javaEnabled***: function javaEnabled() { [native code] }
- ***taintEnabled***: function taintEnabled() { [native code] }
- ***preference***: function preference() { [native code] }
- ***geolocation***: [object GeoGeolocation]
- ***registerContentHandler***: function registerContentHandler() { [native code] }
- ***registerProtocolHandler***: function registerProtocolHandler() { [native code] }
- ***mozIsLocallyAvailable***: function mozIsLocallyAvailable() { [native code] }
- ***plugins*** [object PluginArray]
  - description: The Totem 2.30.2 plugin handles video and audio streams.
    - filename: libtotem-cone-plugin.so
    - version:
    - name: VLC Multimedia Plugin (compatible Totem 2.30.2)
    - item: function item() { [native code] }
    - namedItem: function namedItem() { [native code] }

- ***mimeTypes***: [object MimeTypeArray]
  - Tipo MIME numero 0
    - description VLC Multimedia Plugin
    - enabledPlugin [object Plugin]
    - suffixes
    - type application/x-vlc-plugin
  - Tipo MIME numero 1
    - description VLC Multimedia Plugin
    - enabledPlugin [object Plugin]
    - suffixes
    - type application/vlc

✔Ejemplo: console.log(navigator.userAgent);

Objeto History

- Representa el historial de páginas visitadas por el usuario.
- ***back***(): carga la página anterior del historial
- ***forward***(): carga la página siguiente del historial
- ***go***(): recibe un número que nos permite navegar por el historial de paginas
  - Go(-1); // iría a la pagina anterior
  - Go(); // recarga la actual
  - Go(1); // iría a la pagina siguiente
- ***length***: devuelve el tamaño actual del historial de páginas
  - ✔ Ejemplo:

    history.go(-(history.length-1)); // iría a la primera pagina desde el historial

Objeto Location

- Contiene información acerca de las **propiedades de la página** que hemos cargado, en particular de su nombre. Así mantiene por separado las propiedades protocol, host, port, pathname, hash y search. Además de href (todas las anteriores concatenadas) y hostname (el nombre completo del host).

- *https://www.digitallearning.es:80/modalidades-de-formacion.html?topic=cursos*
  - ***host***: www.digitallearning.es
  - ***hostname***: www.digitallearning.es
  - ***href***: https://www.digitallearning.es:80/modalidades-de-formacion.html?topic=cursos
  - ***pathname***: https://www.digitallearning.es:80/modalidades-de-formacion.html
  - ***port***: 80
  - ***protocol***: https:
  - ***search***: ?topic=cursos
  - ***reload***: function reload() { [native code] }
  - ***replace***: function replace() { [native code] }
  - ***assign***: function assign() { [native code] }
- La propiedad más importante es **href** que contiene la URL completa de la página.
- Ejemplo: console.log(location.href);

Objeto Screen

- Representa la pantalla del dispositivo.

- ***screen.width:*** devuelve el ancho de la pantalla en píxeles.
- ***screen.height:*** devuelve la altura de la pantalla en píxeles.
- ***screen.availWidth:*** devuelve el ancho de la pantalla, en píxeles, menos las características de la interfaz como la barra de tareas de Windows
- ***screen.availHeight:*** devuelve la altura de la pantalla, en píxeles, menos las características de la interfaz como la barra de tareas de Windows.
- ***screen.colorDepth:*** devuelve el número de bits utilizados para mostrar un color
- ***screen.pixelDepth:*** devuelve la profundidad de píxeles de la pantalla.

Ejercicios

1. Crear una página cuyo código JS cree dos fechas: una actual y otra con vuestro cumpleaños:
   1. Imprimir las dos fechas por pantalla con la información completa
   2. Extraer su año e imprimirlos por pantalla
   3. Actualizar el año de la fecha actual por la del año pasado. Escribir la fecha con un formato más legible.
   4. Escribir por pantalla la fecha actual obteniendo el día, el mes y el año por separado y concatenando el símbolo “/”: (dd/mm/yyyy)

2. Crear un programa que muestre por pantalla el número de días que quedan desde hoy hasta fin de curso (30 de junio del año próximo).

   PISTA1: si restamos dos fechas nos da el resultado en milisegundos.

   PISTA2: ¿cómo pasamos de milisegundos a días?

   1 dia → 24 horas * 60minutos * 60segundos * 1000milisegundos

3. Crear un programa que pida al usuario dos números: el día y el mes de tu cumpleaños, y saque todos los años en que tu cumpleaños va a caer en domingo desde este año hasta el año 2100.

   NOTA: los meses comienzan por 0.

4. Crear un programa que muestre la fecha actual en distintos formatos:
   1. dd/mm/yyyy
   2. Dia de la semana mes día año, en inglés: Ej Mon Sep 30 2024
   3. dd/mm/yyyy hh:mm:ss
   4. Hh:mm:ss

5. Crear un programa que calcule la edad de una persona. Para ello pedirá al usuario la fecha de nacimiento en formato dd/mm/yyyy. Se da por hecho que el formato es correcto.

6. Escribir por pantalla una representación textual con el nombre completo del día de la semana en español.

7. Escribir por pantalla los dos últimos dígitos del año de un objeto de tipo Date (por ejemplo de la fecha actual)
8. Crea un programa que simule el lanzamiento de una moneda. Cada vez que se ejecute el programa, debe generar aleatoriamente "Cara" o "Cruz". Luego, muestra el resultado en una alerta, en la consola y en la página.

9. Crea un programa que genere un número aleatorio entre 1 y 10 (incluido). Luego, pide al usuario que intente adivinar el número utilizando un cuadro de diálogo (prompt). Si el usuario adivina correctamente, muestra una alerta con el mensaje "¡Felicidades, adivinaste el número!". Si no adivina, muestra una alerta diciendo "Lo siento, el número era [número]".

10. Genera 10 números aleatorios **entre 0 y 20 (incluido)** y calcula la suma de ellos.

    Debe mostrar por pantalla: “La suma de los numeros xxxxx es x”

11. Solicita al usuario su nombre y muestra con un mensaje alert ‘Buenos días Nombre’, Buenas tardes Nombre’ o ‘Buenas noches Nombre’ dependiendo de la hora: (7 - 12 – 20 h)

12. Crea un programa que genere una fecha aleatoria en el pasado. El programa debe mostrar la fecha generada en la consola, en la página y en una alerta. El rango de la fecha aleatoria será entre el año 2000 y el año actual.

13. Crea un programa que pida al usuario su nombre y apellidos (de una vez) y muestre por pantalla:
    - El tamaño del nombre mas los apellidos(sin contar espacios)
    - La cadena en minúsculas y en mayúsculas
    - Que divida el nombre y los apellidos y los muestre en 3 líneas
    - Que devuelva una cadena compuesta por la inicial del nombre, el primer apellido y la inicial del segundo apellido. EJ: Para Belén Tudó Ramírez → btudor
    - Que devuelva una cadena compuesta por las tres primeras letras del nombre y de los dos apellidos. EJ: beltudram

14. Crea un programa que genere un número aleatorio entre 10 y 20. Luego, pide al usuario que adivine el número. El programa deberá indicarle al usuario si su intento es demasiado alto, demasiado bajo, o si ha acertado. El juego continúa hasta que el usuario adivine el número correcto. Cuando lo adivine, debe escribir en pantalla el número adivinado junto al número de intentos utilizado.

15. Crear un programa que pregunte al usuario si acepta o no abrir una ventana nueva. Sólo si acepta, se abrirá la nueva ventana. La nueva ventana no tendrá barra de herramientas, ni location, ni barra de menú. Tendrá 200x80 pixeles y y se posicionará en la posición 500x500. Será resizable y tendrá scrollbars.
    La nueva ventana incluirá el texto: “Entorno cliente”

16. Crear una página index.html con un título h1 ‘Página principal’. Tendrá 6 botones.
    Crear otra página segunda.html con un título h2 ‘Página secundaria’ con 3 botones.
    - index.html:
        - Botón 1. Abrirá la ventana segunda.html con 300x400 pixeles y en la posición 400x400.
        - Botón 2. Mover la ventana segunda.html 200 pixeles a la derecha y 100 pixeles abajo.
        - Botón 3. Mover la ventana segunda.html a la posición 500x200
        - Botón 4. Aumentar el tamaño de la ventana segunda.html 100 pixeles de ancho y 100 alto
        - Botón 5. Cambiar el tamaño de la ventana segunda.html a 100x150.
    - segunda.html:
        - Botón 1: pondrá el fondo de la ventana secundaria de color verde.
        - Botón 2: pondrá el fondo de la ventana principal de color rojo.
        - Botón 3. Cerrar la ventana segunda.html.
        - Botón 4: cerrará la ventana secundaria y a continuación la ventana principal

    NOTA1: ejecutar instrucciones js en el evento onclick:
    <input type=”button” onclick=” instrucción ” />

    NOTA2: color de fondo de una página:
    document.body.style.backgroundColor=’xxx’;

    NOTA3: Para que una ventana se muestre por encima de otra, hay que poner el foco en dicha ventana:
    ventana.focus()

17. Crear una pagina que contenga un botón, de forma que al pulsar el botón, se abra otra ventana de 600x600: ‘selector_color.html’. Al cargar esta segunda página, preguntar al usuario un color. (EJ: red, #3498db) con un cuadro de dialogo. Al aceptar, mostrar dicho color como fondo de la **ventana principal** y cerrar la ventana selector_color.

18. Vamos a utilizar temporizadores: Crear una pagina y que cada segundo cambie de color el fondo: naranja / azul hasta pulsar un botón Detener.
19. Mostrar una pagina que diga 'Nos vamos!' y que tras 5 segundos nos redirija al buscador de google: https://www.google.es
20. Crear una página con las siguientes características:
    1. Mostrar el mensaje 'En un lugar de la Mancha' por consola cada 2 segundos.
    2. Usar un botón en la página para parar el temporizador anterior:

       <input type="button" onclick="para()" >
    3. Usar otro botón en la página para mostrar por consola el mensaje 'Han pasado 3 segundos' después de 3 segundos, una sóla vez.
    4. Usar otro botón para abrir una ventana nueva de 300x300 y escribir un texto con <p> en su interior desde js. Esta ventana nueva debe mostrar la hora actual en formato HH:MM:SS. Se tiene que actualizar continuamente como un reloj.

       Parar el reloj a los 5 segundos.

       Cerrarla tras 10 segundos.

    NOTA1: Usar el código js en un fichero independiente: accion.js

    NOTA2: Usar funciones dentro de los temporizadores:

    ```
    setInterval(funcion, tiempo);
    function funcion{
        ---
    }
    ```
21. Crear una página index.html que contenga los siguientes botones:
    - Botón que abre dos ventanas: ventana1.html y ventana2.html, ambas de 300x200. La ventana1 en la posición 100,300 y la ventana2 en la posición 500,300
    - Botón que cierra las dos ventanas
    - Botón para cambiar el fondo de la primera ventana a:#FFECA1
    - Botón para cambiar el fondo de la segunda ventana a:#EFC3CA

    ventana1.html tiene dos botones:
    - Botón para enviar a la ventana2 el mensaje: "La ventana 1 te saluda"
    - Botón para cambiar el fondo de la segunda ventana a: #7DDA58

    ventana2.html tiene dos botones:
    - Botón para enviar a la ventana1 el mensaje: "La ventana 2 te saluda"
    - Botón para cambiar el fondo de la segunda ventana a: #5DE2E7

    Incluir todo el código JavaScript en un único fichero accion.js

Ejercicios de repaso:

22. Crear una aplicación web que muestre 2000 cuadrados de color aleatorio de 50 x50 píxeles. Su posición también será aleatoria
23. Crear una página html con un botón, tal que al pulsarlo, se abra una nueva ventana emergente de 400x200 con un mensaje. Esta ventana nueva debe cerrarse automáticamente tras 10 segundos. En la ventana principal, se mostrará la cuenta atrás con los segundos que faltan para que la ventana emergente se cierre.
24. Variar el ejercicio anterior para que la cuenta atrás no sea de 10 segundos, si no que en la página index.html, haya además, un campo de tipo input que reciba el número de segundos tras los que la ventana emergente se cierre.
25. Crear una página html con el siguiente body:

```
<body>
  <h1>Ventana Emergente con Temporizador Personalizado</h1>
  <p>
    Ingresa el número de segundos para que la ventana emergente se cierre
    automáticamente.
  </p>
  <label for="segundos">Número de segundos: </label>
  <input type="text" id="segundos" min="1" placeholder="Escribe el tiempo en segundos" />
  <button onclick="abrirVentana()">Abrir Ventana Emergente</button>
  <p id="resultado"></p>
  <!-- Aquí se mostrará el mensaje cuando la ventana se cierre -->
</body>
```

Crear código JavaScript tal que al pulsar el botón, se abra una ventana emergente donde se muestre la cuenta atrás de segundos que quedan para que se cierre automáticamente. El número de segundos vendrá en el campo cuyo id es ‘segundos’. Cuando se cierre, debe escribirse en la página principal, “Ya han pasado X segundos y se ha cerrado la ventana emergente”.

26. Repetir el ejercicio anterior pero en este caso, la página ‘emergente.html’ está previamente creada.

Index.html:

```
<body>
  <h1>Ventana Emergente con Temporizador</h1>
  <p>
    Ingresa el número de segundos para que la ventana emergente se cierre
    automáticamente.
  </p>

  <!-- Campo de entrada para el número de segundos -->
  <label for="segundos">Número de segundos: </label>
  <input type="number" id="segundos" min="1"placeholder="Escribe el tiempo en segundos"
  />
  <button onclick="abrirVentana()">Abrir Ventana Emergente</button>
  <p id="resultado"></p>
  <script src="script.js"></script>
</body>
```

Emergente.html:

```
<body>
  <h2>Cuenta atrás en la ventana emergente</h2>
  <p id="cuentaAtras">Esperando...</p>

  <!-- Enlace al archivo JavaScript externo -->
  <script src="emergente.js"></script>
</body>
```