Javascript

Semana 4

Índice

Resultado de aprendizaje, contenido y criterios de evaluación

Resultado de aprendizaje
RA3. Escribe código, identificando y aplicando las funcionalidades aportadas por los objetos predefinidos del lenguaje.
Contenido
7 - Mecanismos del navegador para el almacenamiento y recuperación de información. 8 - Depuración y documentación del código.
Criterios de evaluación
a) Se han identificado los objetos predefinidos del lenguaje c) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para cambiar el aspecto del navegador y el documento que contiene d) Se han generado textos y etiquetas como resultado de la ejecución de código en el navegador. e) Se han escrito sentencias que utilicen los objetos predefinidos del lenguaje para interactuar con el usuario h) Se ha depurado y documentado el código

Resultado de aprendizaje
R4. Programa código para clientes web analizando y utilizando estructuras definidas por el usuario
Contenido
5 - Creación de objetos. 6 - Definición de métodos y propiedades
Criterios de evaluación
a) Se han clasificado y utilizado las funciones predefinidas del lenguaje b) Se han creado y utilizado funciones definidas por el usuario. g) Se ha creado código para definir la estructura de objetos. h) Se han creado métodos y propiedades k) Se ha depurado y documentado el código

Resultado de aprendizaje
RA5. Desarrolla aplicaciones web interactivas integrando mecanismos de manejo de eventos
Contenido
2 - Utilización de formularios desde código. 3 - Modificación de apariencia y comportamiento. 4 - Validación y envío. 5 - Expresiones regulares. 6 - Prueba y documentación del código
Criterios de evaluación
a) Se han reconocido las posibilidades del lenguaje de marcas relativas a la captura de los eventos producidos b) Se han identificado las características del lenguaje de programación relativas a la gestión de los eventos c) Se han diferenciado los tipos de eventos que se pueden manejar d) Se ha creado un código que capture y utilice eventos f) Se han validado formularios web utilizando eventos h) Se ha probado y documentado el código

Resultado de aprendizaje
RA6. Desarrolla aplicaciones web analizando y aplicando las características del modelo de objetos del documento
Contenido
4 - Programación de eventos 5 - Diferencias en las implementaciones del modelo 6 - Independencia de las capas de implementación de aplicaciones web
Criterios de evaluación
a) Se ha reconocido el modelo de objetos del documento de una página web b) Se han identificado los objetos del modelo, sus propiedades y métodos. c) Se ha creado y verificado un código que acceda a la estructura del documento d) Se han creado nuevos elementos de la estructura y modificado elementos ya existentes. e) Se han asociado acciones a los eventos del modelo h) Se han independizado las tres capas de implementación (contenido, aspecto y comportamiento), en aplicaciones web

Resultado de aprendizaje
RA7. Desarrolla aplicaciones web dinámicas, reconociendo y aplicando mecanismos de comunicación asíncrona entre cliente y servidor
Contenido
1 - Mecanismos de comunicación asíncrona. 2 – Modificación dinámica del documento utilizando comunicación asíncrona 3 - Formatos para el envío y recepción de información. 4 - Librerías y frameworks de actualización dinámica. 5 - Integración en diferentes navegadores 6 - Prueba y documentación del código
Criterios de evaluación
a) Se han evaluado las ventajas e inconvenientes de utilizar mecanismos de comunicación asíncrona entre cliente y servidor web b) Se han analizado los mecanismos disponibles para el establecimiento de la comunicación asíncrona d) Se han identificado sus propiedades y sus métodos

Semana 4. Manipulación del BOM

Vamos a ver lo relativo a la manipulación de la ventana del navegador y del navegador en sí.

Sesión 1

4.1. Ejemplo guiado

4.1.1 El Monitor del Consejo Andoriano

Comenzamos con un ejemplo guiado.

Ejemplo guiado: El Monitor del Consejo Andoriano

En el planeta capital de Andoria, los ciudadanos usan interfaces holográficas gestionadas por los Navegadores de Cristal. Tu misión como programador es dominar el lenguaje que controla estas interfaces: el lenguaje del DOM y del BOM. ¡Solo así podrás crear interacciones dinámicas y adaptadas a los entornos de guerra estelar!
Para ello Desarrolla un script en JavaScript puro (sin frameworks ni librerías externas) que implemente las funcionalidades de un "Monitor del Consejo Andoriano" según el siguiente código HTML proporcionado.

1. *Antes de comenzar leer lo apartados ANEXO VIII. BOM y DOM para comprender como JS se aplica al acceso del BOM y del DOM.*
2. *Crea una página HTML con el siguiente contenido*

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Monitor del Consejo Andoriano</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <h1 id="titulo">Estado del Consejo</h1>
  <p id="estado">Sin conexión a la red intergaláctica.</p>
  <button id="conectar">Conectar</button>
  <button id="salir">Salir del Sistema</button>
  <p id="infoVentana"></p>

  <script src="andoria.js"></script>
</body>
</html>
```

2. *Creamos el archivo andoria.js:*

   a) *Usa document.getElementById() para capturar los elementos:*
      - *El elemento con id titulo*
      - *El elemento con id estado*
      - *El botón con id conectar*
      - *El botón con id salir*
      - *El elemento con id infoVentana*

   b) *Crea una función que cambie el texto del <p> con id "estado" al hacer clic en "Conectar". El nuevo texto debe indicar que se ha establecido conexión con la red intergaláctica.*

   c) *Crea una función que intente cerrar la ventana usando window.close() cuando se haga clic en el botón "Salir del Sistema". Debes manejar adecuadamente el caso en que el navegador bloquee esta acción (lo cual ocurre por razones de seguridad) y explicar estos límites de seguridad del BOM (Browser Object Model) en comentarios dentro del código.*

   d) *Muestra en el elemento infoVentana el ancho y alto de la ventana usando window.innerWidth e innerHeight.*

   e) *Usa setInterval para actualizar el tamaño cada 2 segundos.*

Solución:

2.a Usa document.getElementById() para capturar los elementos necesarios.

1. En el fichero andoria.js primeramente capturamos los elementos:

```
const titulo = document.getElementById('titulo');
const estado = document.getElementById('estado');
const botonConectar = document.getElementById('conectar');
const botonSalir = document.getElementById('salir');
const infoVentana = document.getElementById('infoVentana');
```

2.b Crea una función que cambie el texto del <p> con id "estado" al hacer clic en "Conectar". El nuevo texto debe indicar que se ha establecido conexión con la red intergaláctica.

Asignamos un l*istener* al botón conectar que llame a la función conectar:

botonConectar.addEventListener('click', conectar);

Luego desarrollamos la función conectar para cambiar el texto del *estado* por: "Conectado a la red intergaláctica. Bienvenido al Consejo Andoriano.". Y el texto del botón por "Conectado". Queda de la siguiente forma:

```
function conectar() {
    estado.textContent = "Conectado a la red intergaláctica.
                          Bienvenido al Consejo Andoriano.";
    estado.style.color = "green";
    botonConectar.textContent = "Conectado";
    botonConectar.disabled = true;
}

// Asignar la función al evento click del botón Conectar
botonConectar.addEventListener('click', conectar);
```

2.c. Crea una función que intente cerrar la ventana usando window.close() cuando se haga clic en el botón "Salir del Sistema".

Recuperamos la ventana actual y la guardamos en la variable *ventana*:

let ventana = window.open('', '_self');

Luego tratamos de cerrarla:

ventana.close();

Esto puede fallar porque los navegadores modernos bloquearán el cierre, **Averigua por qué**.

```
function salirDelSistema() {
    try {
        // Intentar cerrar la ventana
        let ventana = window.open('', '_self');
        ventana.close();

        // Si no se pudo cerrar, mostrar mensaje
        setTimeout(() => {
            estado.textContent = "El navegador ha bloqueado el
                cierre automático. Por favor,
                cierre esta pestaña manualmente.";
            estado.style.color = "orange";
        }, 100);
    } catch (error) {
        estado.textContent = "Error: " + error.message;
        estado.style.color = "red";
    }
}

// Asignar la función al evento click del botón Salir
botonSalir.addEventListener('click', salirDelSistema);
```

2.d. Muestra en el elemento infoVentana el ancho y alto de la ventana usando window.innerWidth e innerHeight.

Para mostrar estos valores los recuperamos de la información que nos proporciona *window*. Mostrandolo en *infoVentana*.

```
const ancho = window.innerWidth;
const alto = window.innerHeight;
infoVentana.textContent = `Tamaño de ventana:
                          ${ancho}px x ${alto}px`;
```

2.e. Usa setInterval para actualizar el tamaño cada 2 segundos.

La forma de establecer un intervalo de tiempo que se repita es con la función setInterval(), que necesita como parámetros de entrada una función y el tiempo de repetición en milisegundos.

Con lo que nos piden tendremos que hacer algo así:

setInterval(actualizarTamañoVentana, 2000);

Llamamos a la función *actualizarTamañoVentana cada 2 segundos.*

Combinando el apartado anterior y este el código quedaría:

```
function actualizarTamañoVentana() {
    const ancho = window.innerWidth;
    const alto = window.innerHeight;
    infoVentana.textContent = `Tamaño de ventana:
                              ${ancho}px x ${alto}px`;
}

// Mostrar el tamaño inicial
actualizarTamañoVentana();

// Actualizar cada 2 segundos
setInterval(actualizarTamañoVentana, 2000);
```

4.1.2 Ejercicio propuesto: Gravity Falls: Panel de anomalías y Diario 3

Ejemplo guiado: Gravity Falls: Panel de anomalías y Diario 3

Dipper y Mabel investigan anomalías del portal interdimensional. Necesitan una «consola» web que lea y manipule el BOM (Browser Object Model) para recolectar pruebas: información del navegador, de la pantalla, de la URL y del historial. También deberán provocar acciones controladas: cambiar el hash, modificar la barra de direcciones con History API, navegar por el historial, abrir y cerrar una ventana auxiliar, recargar, imprimir, detectar cambios de conexión y medir el tamaño de la ventana.

Objetivo: Implementar con JavaScript puro (sin librerías ni frameworks) una página que:

Lea propiedades clave del BOM: window, location, history, navigator y screen.

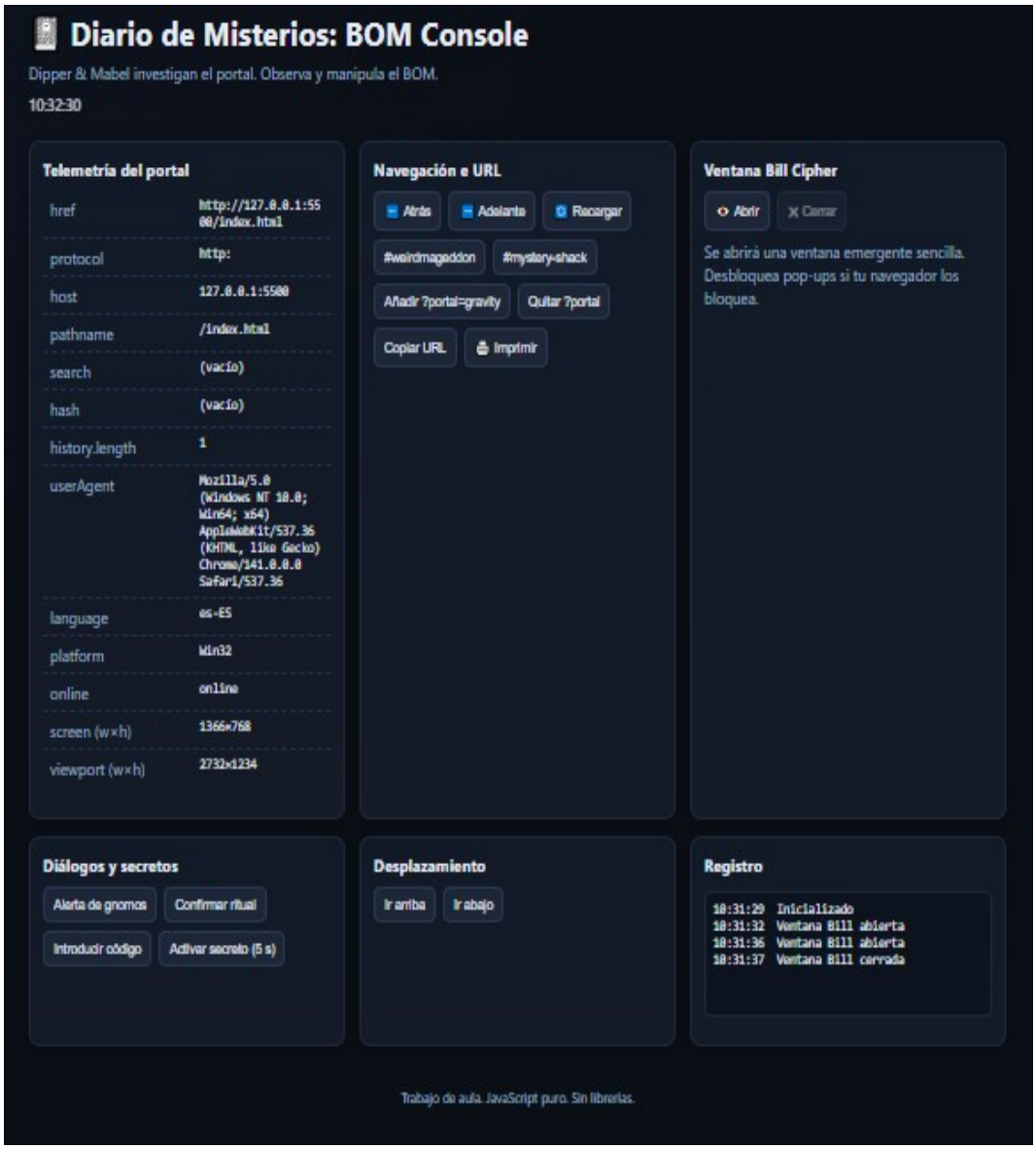
Efectúe acciones del BOM mediante botones seguros: back/forward, reload, pushState, hash, abrir/cerrar ventana, imprimir y copiar la URL.

Gestione eventos de BOM: online/offline, resize, beforeunload opcional, scroll.

Use temporizadores del BOM: setInterval y setTimeout.

Interactúe con el usuario mediante alert, confirm y prompt.

Tareas:
A. Panel “Telemetría del portal” que muestre: location.href, protocol, host, pathname, search, hash; history.length; navigator.userAgent, language, platform, onLine; screen.width/height; window.innerWidth/innerHeight.
Actualiza en tiempo real al cambiar tamaño o conectividad.
B. Botones de acción:
• Atrás y Adelante (history.back/history.forward).
• Recargar (location.reload).
• Cambiar hash a #weirdmageddon y #mystery-shack.
• pushState para añadir o quitar el parámetro ?portal=gravity sin recargar.
• Copiar URL al portapapeles.
• Imprimir (window.print).
• Abrir “Ventana Bill Cipher” y Cerrar ventana. La ventana debe mostrar un título y una imagen emoji o ASCII.
C. Temporizadores:
• Reloj en vivo con setInterval.
• “Activar secreto”: tras 5 s setTimeout muestra un mensaje y resalta el panel.
D. Diálogos:
• Botón “Alerta de gnomos” (alert).
• Botón “Confirmar ritual” (confirm) que registra el resultado.
• Botón “Introducir código” (prompt) que guarda y muestra el valor.
E. Utilidades de scroll: “Ir arriba” y “Ir abajo” con scrollTo suave.
F. Opcional clásico: beforeunload que pregunte antes de salir si hay un “ritual” activo.


Diario de Misterios: BOM Console
Dipper & Mabel investigan el portal. Observa y manipula el BOM.
10:32:30
Telemetría del portal
href
http://127.0.0.1:5500/index.html
protocol
http:
host
127.0.0.1:5500
pathname
/index.html
search
(vacío)
hash
(vacío)
history.length
1
userAgent
Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
language
es-ES
platform
Win32
online
online
screen (w×h)
1366×768
viewport (w×h)
2732×1234
Navegación e URL
Atrás
Adelante
Recargar
#weirdmageddon
#mystery-shack
Añadir ?portal=gravity
Quitar ?portal
Copiar URL
Imprimir
Ventana Bill Cipher
Abrir
Cerrar
Se abrirá una ventana emergente sencilla. Desbloquea pop-ups si tu navegador los bloquea.
Diálogos y secretos
Alerta de gnomos
Confirmar ritual
Introducir código
Activar secreto (5 s)
Desplazamiento
Ir arriba
Ir abajo
Registro
10:31:29 Inicializado
10:31:32 Ventana Bill abierta
10:31:36 Ventana Bill abierta
10:31:37 Ventana Bill cerrada
Trabajo de aula. JavaScript puro. Sin librerías.

Sesión 2.

4.2. Ejemplo guiado

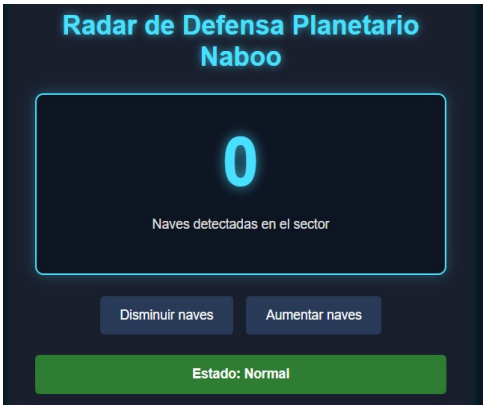
4.2.1 Actualización del sistema de defensa de Naboo

Ejemplo guiado: Actualización del sistema de defensa de Naboo

La Princesa Amidala os ha encomendado una misión crítica: actualizar el sistema de defensa del planeta Naboo. El radar de defensa necesita mostrar en tiempo real el número de naves detectadas y cambiar el estado de alerta según la cantidad de naves.
La interface contendrá:

- Un indicador numérico que nos muestra el número de naves detectadas.
- Un botón para simular el incremento de naves
- Un botón para simular el decremento de naves
- Un texto y un fondo de color que ambos cambiarán de la siguiente manera según aumente o disminuya el número de naves:
  * “Estado: normal” menos o igual a 5 naves detectadas. Color de fondo verde (#0e1523).
  * “Estado: alerta” menos o igual a 10 naves detectadas. Color de fondo amarillo (#1a1500).
  * “Estado: peligro” más de 10 naves detectadas. Color de fondo rojo (#1a0000).

A continuación se muestra el front de la web.



1.- Vamos a crear el fichero con el nombre que aparece en el HTML e iniciamos con:

```
document.addEventListener('DOMContentLoaded', function() { …… }
```

*La función se ejecuta cuando el HTML está **cargado y parseado**. No espera a imágenes, iframes o estilos.*

Entre las llaves irá todo nuestro código.

2.- Ahora seleccionamos lo elementos con los que vamos a trabajar

```
document.addEventListener('DOMContentLoaded', function() {
    const countDisplay = document.getElementById('nave-count');
    const decreaseBtn = document.getElementById('decrease-btn');
    const increaseBtn = document.getElementById('increase-btn');
    const statusIndicator = document.getElementById('status-indicator');
    const radarDisplay = document.getElementById('radar-display');
}
```

NOTA: siempre que podamos usemos id en los elemento ya que al ser únicos es fácil identificarlos.

3 - Añadimos los listener a los botones para que incremente y decrementen.

Recuerda el formato de addEventLitener → . addEventLitener(evento, función).

```
document.addEventListener('DOMContentLoaded', function() {
    const countDisplay = document.getElementById('nave-count');
    const decreaseBtn = document.getElementById('decrease-btn');
    const increaseBtn = document.getElementById('increase-btn');
    const statusIndicator = document.getElementById('status-indicator');
    const radarDisplay = document.getElementById('radar-display');

    // Aumentar el número de naves
    increaseBtn.addEventListener('click', () => {
        navesCount++;
        updateDisplay();
    });

    // Disminuir el número de naves (no permitir valores negativos)
    decreaseBtn.addEventListener('click', () => {
        if (navesCount > 0) {
            navesCount--;
            updateDisplay();
        }
    });
}
```

4 – Creemos ahora la variable contador de nave y el método updateDisplay(), que es la que tiene la lógica para pintar los menaje de alerta:

```
document.addEventListener('DOMContentLoaded', function() {
    const countDisplay = document.getElementById('nave-count');
    const decreaseBtn = document.getElementById('decrease-btn');
    const increaseBtn = document.getElementById('increase-btn');
    const statusIndicator = document.getElementById('status-indicator');
    const radarDisplay = document.getElementById('radar-display');

     // Estado inicial
     let navesCount = 0;

   function updateDisplay() {
    countDisplay.textContent = navesCount;

    // Determinar el nivel de alerta y cambiar colores
    if (navesCount <= 5) {
      // Nivel bajo (verde)
      radarDisplay.style.backgroundColor = "#0e1523";
      statusIndicator.textContent = "Estado: Normal";
      statusIndicator.className = "status nivel-bajo";
    } else if (navesCount <= 10) {
      // Nivel medio (amarillo)
      radarDisplay.style.backgroundColor = "#1a1500";
      statusIndicator.textContent = "Estado: Alerta";
      statusIndicator.className = "status nivel-medio";
    } else {
      // Nivel alto (rojo)
      radarDisplay.style.backgroundColor = "#1a0000";
      statusIndicator.textContent = "Estado: Peligro";
      statusIndicator.className = "status nivel-alto";
    }
   }
    // Aumentar el número de naves
    increaseBtn.addEventListener('click', () => {
        navesCount++;
        updateDisplay();
    });

    // Disminuir el número de naves (no permitir valores negativos)
    decreaseBtn.addEventListener('click', () => {
        if (navesCount > 0) {
            navesCount--;
            updateDisplay();
        }
    });
}
```

Nota que lo primero que hace la función es actualizar el texto de `<div id="nave-count">0</div>` *con el valor que contiene en ese momento la variable navesCount a través del método .textContent.* `countDisplay.textContent = navesCount;`

5.- Ahora solo queda llamar a la función ante del cierre de
document.addEventListener("DOMContentLoaded", function ()

```
document.addEventListener("DOMContentLoaded", function () {
 …...
  // Inicializar la visualización
  updateDisplay();
});
```

4.2.2 Ejercicio propuesto: Actualización del radar Naboo

Realizar las siguientes modificaciones dentro de :

- Añadir un input para poner de forma directa la cantidad de naves detectadas. Y un botón para que al hacer click, llame a la función updateDisplay() que hará que aparezca el valor del input en el radas, concretamenter actualizar 0 . Validar previamente que el valor del input está entre [0…999] .

- Añadir un botón “Pánico” con pulsación larga, es decir si maousedown >= 800 ms, se fijará navesCount a 15, usando updateDisplay() para actualizar el valor y fuerza “Peligro” (rojo). Usa mouseup para cancelar el temporizador.

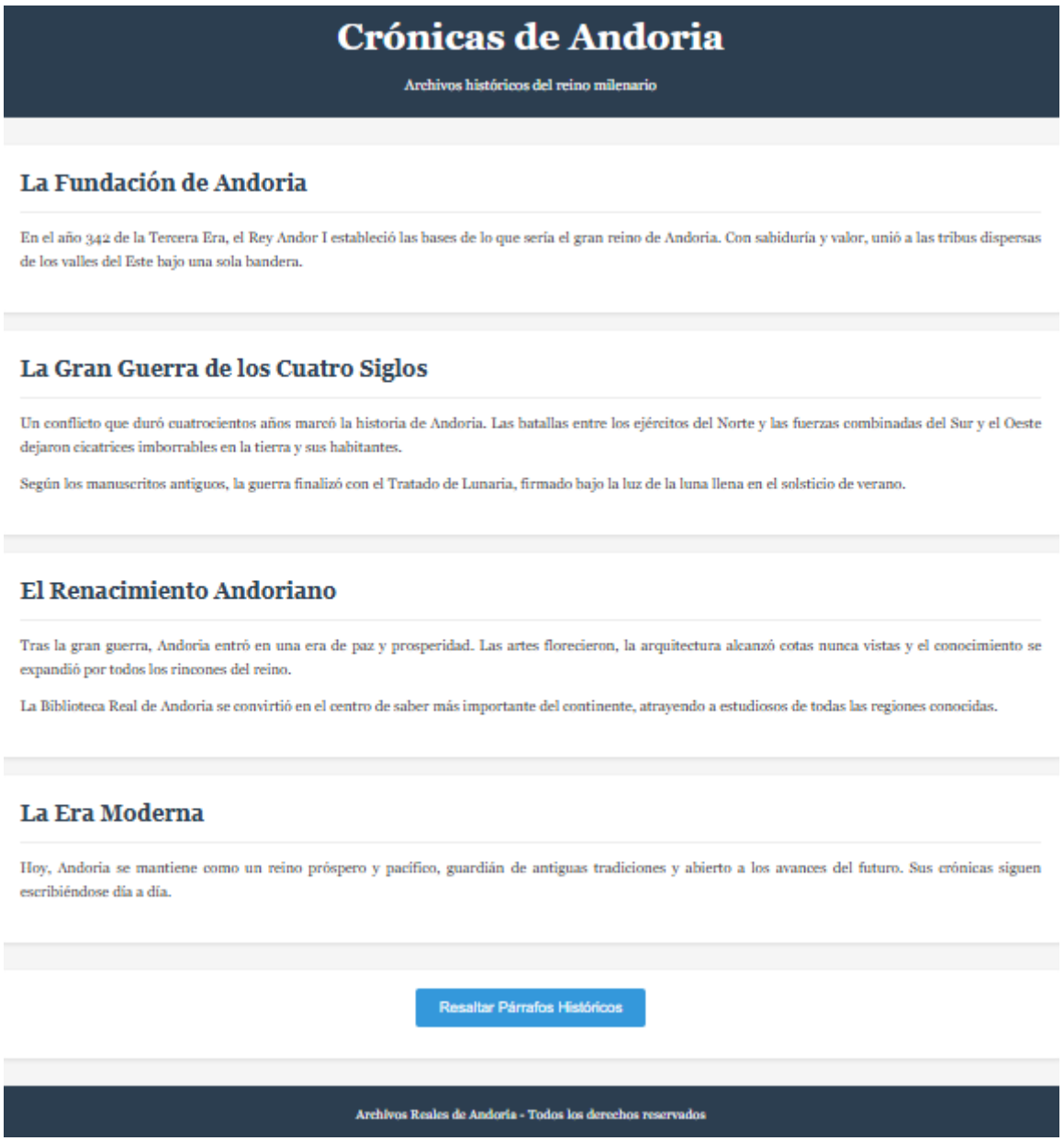
Sesión 3.

4.3. Ejemplo guiado

4.3.1 Registros del Archivo Galáctico

Ejemplo guiado: Registros del Archivo Galáctico

Los cronistas de Andoria quieren poder ver los registros de su historia. Crea una página con varios textos (títulos y párrafos). Al pulsar un botón, recorre todos los <p> del documento y cámbiales el estilo para resaltarlos (negrita, subrayado, color azul). Muestra cuántos párrafos han sido modificados.



1.- Vamos a crear el fichero con el nombre que aparece en el HTML e iniciamos con:

```
document.addEventListener('DOMContentLoaded', function() { …… }
```

2.- Obtenemos las referencias del botón y contador

```
// Obtener referencia al botón y al contador
const resaltarBtn = document.getElementById('resaltarBtn');
const contador = document.getElementById('contador');
```

3.- Añadimos Listener al evento click al botón:

```
resaltarBtn.addEventListener('click', () => { …. });
```

4.- para modificar los párrafo necesitamos obtenerlos y luego recorrerlos para modificar su estilo.

```
// Añadir evento click al botón
resaltarBtn.addEventListener('click', () => {
    // Obtener todos los párrafos
    const parrafos = document.querySelectorAll('p');
    let contadorResaltados = 0;

    // Recorrer cada párrafo
    parrafos.forEach((parrafo) => {
        // Verificar si el párrafo no está en el footer
        if (!parrafo.closest('footer')) {
            // Alternar la clase 'resaltado'
            parrafo.classList.toggle('resaltado');

            // Si el párrafo tiene la clase resaltado, incrementar el contador
            if (parrafo.classList.contains('resaltado')) {
                contadorResaltados++;
            }
        }
    });
});
```

Nota que excluimos el párrafo del footer. Y que alternamos con .toggle la clase css ‘resaltado’.

4.- Ahora solo queda contar los párrafos, dentro del forEach:

```
  // Añadir evento click al botón
  resaltarBtn.addEventListener('click', () => {
      // Obtener todos los párrafos
      const parrafos = document.querySelectorAll('p');
      let contadorResaltados = 0;

      // Recorrer cada párrafo
      parrafos.forEach((parrafo) => {
          // Verificar si el párrafo no está en el footer
          if (!parrafo.closest('footer')) {
              // Alternar la clase 'resaltado'
              parrafo.classList.toggle('resaltado');

              // Si el párrafo tiene la clase resaltado, incrementar el contador
              if (parrafo.classList.contains('resaltado')) {
                  contadorResaltados++;
              }
          }
      });

      // Mostrar el contador de párrafos modificados
      if (contadorResaltados > 0) {
          contador.textContent = `Se han resaltado ${contadorResaltados} párrafos históricos.`;
      } else {
          contador.textContent = "Se ha quitado el resaltado de todos los párrafos.";
      }
  });
});
```

4.3.2 Ejercicio propuesto

Ejemplo guiado: Registros del Archivo Galáctico

Crea una página con texto en “andoriano” (ficticio). Al pulsar un botón, cambia todos los textos por su versión en idioma galáctico común (español). Usa innerText o textContent para hacerlo.

ANEXO VIII. BOM y DOM

Conceptos previos

Como indicamos en la introducción JavaScript nos permite hacer que las páginas web sean dinámicas e interactivas. El lenguaje JavaScript permite modificar el contenido de una página, responder a eventos del usuario y comunicarse con servidores, entre muchas otras cosas.

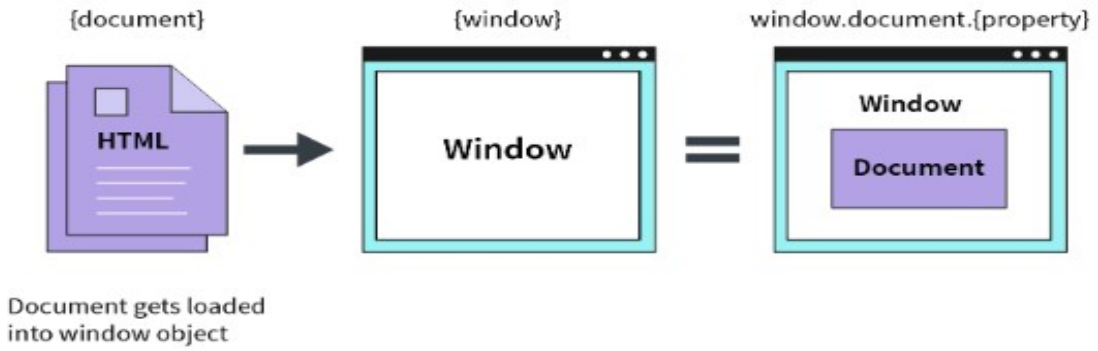
JavaScript hace uso de de unos objetos del navegador para acceder a sus elementos, este es el BOM (Browser Object Model) y proporciona acceso a elementos fuera del contenido de la página, como la barra de direcciones, historial de navegación, y tamaño de la ventana del navegador además de interactuar con el navegador y realizar acciones como abrir nuevas pestañas o manipular la URL.

Por otra parte JavaScript también es capaz de manipular el documento HTML a través del objeto DOM (Document Object Model) que representa, de forma estructurada, el contenido de una página web

window es el objeto global en el navegador que contiene tanto el BOM como el DOM. Representa la ventana del navegador y ofrece propiedades y métodos para controlar la ventana, como su tamaño y eventos de carga.

document es un objeto dentro de window que representa la página web actual cargada. Es parte del DOM y permite acceder y modificar los elementos HTML de la página mediante JavaScript.

En resumen, JavaScript usa el objeto window para acceder al BOM (datos del navegador) y al DOM (estructura de la página web), mientras que document es el punto de entrada al DOM dentro de la ventana.



Modelo de Objetos del Navegador (BOM)

El concepto de BOM es más amplio, ya que incluye muchas más cosas que el propio documento HTML que se está visualizando. Por ejemplo en un navegador, y en el BOM, tenemos también la posibilidad de controlar el historial de páginas que el usuario ha visitado, las direcciones de las páginas que está visitando, las cookies y muchas más cosas.

Podemos acceder a las dimensiones de la ventana del navegador con:

```
let w = window.innerWidth;
let h = window.innerHeight;
```

Oros métodos útiles son:

Método	Descripción
window.open()	Abre una nueva ventana.
window.close()	Cierra una la ventana
window.moveTo()	Mueve la ventana a una dirección
window.resizeTo()	Redimensiona la ventama

A parte de estos métodos existen otros objetos que nos permiten acceder a otros elementos:

Objeto	Descripción
window.screen o screen solo	contiene información sobre la pantalla del usuario • screen.width • screen.height • screen.availWidth • screen.availHeight • screen.colorDepth • screen.pixelDepth
window.location o location solo	se puede utilizar para obtener la dirección de la página actual (URL) y redirigir el navegador a una nueva página. • href → devuelve el href (URL) de la página actual • hostname → devuelve el nombre de dominio del host web • pathname → devuelve la ruta y el nombre del archivo de la página actual • protocol → devuelve el protocolo web utilizado (http: o https:) • assign() → carga un nuevo documento
window.history o history solo	contiene el historial de los navegadores.

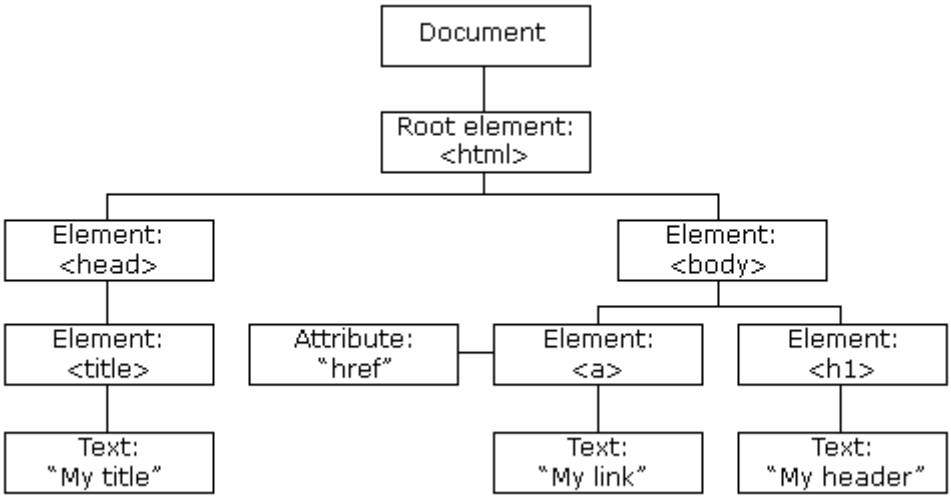
	• Back() → lo mismo que hacer clic atrás en el navegador • forward() → lo mismo que hacer clic hacia adelante en el navegador
window.navigator o navigator solo	contiene información sobre el navegador del visitante. • cookieEnabled → propiedad devuelve verdadero si las cookies están habilitadas • language → propiedad devuelve el idioma del navegador • onLine → propiedad devuelve verdadero si el navegador está en línea
window.alert(“mensaje”) o alert solo	cuadro de alerta, el usuario deberá hacer clic en "Aceptar" para continuar.

window.confirm o confirm solo	Cuando aparezca un cuadro de confirmación, el usuario deberá hacer clic en "Aceptar" o "Cancelar" para continuar.
	`if (confirm("Press a button!")) {`
	`  txt = "You pressed OK!";`
	`} else {`
	`  txt = "You pressed Cancel!";`
	`}`

window.prompt o prompt solo	Cuando aparece un cuadro de aviso, el usuario deberá hacer clic en "Aceptar" o "Cancelar" para continuar después de ingresar un valor de entrada.
window.setTimeout() o setTimeout() solo	Véase anexo ANEXO X. Funciones setTimeout y setInterval
window.setInterval() o setInterval() solo	
window.document o document solo	Véase Modelo de Objetos de Documento (DOM)

Modelo de Objetos de Documento (DOM)

El árbol de objetos HTML DOM



El Modelo de Objetos de Documento (DOM) del W3C es una plataforma y una interfaz neutral en cuanto al idioma que permite a los programas y scripts acceder y actualizar dinámicamente el contenido, la estructura y el estilo de un documento. En definitiva, HTML DOM es un estándar sobre cómo obtener, cambiar, agregar o eliminar elementos HTML. O dicho de una manera más simplificada: es una estructura jerárquica que representa todos los elementos HTML como objetos en JavaScript. Nos permite:

- Acceder a elementos
- Cambiar su contenido o estilos
- Crear o eliminar nodos
- Escuchar y responder a eventos

El estándar W3C DOM se divide en 3 partes diferentes:

Core DOM: modelo estándar para todo tipo de documentos
XML DOM: modelo estándar para documentos XML
HTML DOM: modelo estándar para documentos HTML

La interfaz de programación son las propiedades y métodos de cada objeto.

Una propiedad es un valor que puede obtener o establecer (como cambiar el contenido de un elemento HTML).

Un método es una acción que puedes realizar (como agregar o eliminar un elemento HTML).

El siguiente ejemplo cambia el contenido (el innerHTML) del <p>elemento con id="demo":

```
<html>
  <body>
    <p id="demo"></p>
      <script>
        document.getElementById("demo").innerHTML = "Hello World!";
      </script>
  </body>
</html>
```

En el ejemplo anterior, getElementByIdes un **método** , mientras que innerHTML es una **propiedad**.

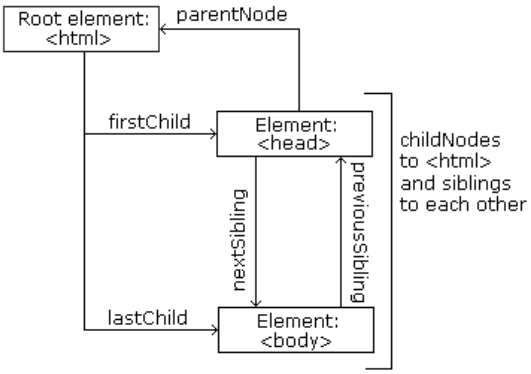
Volvamos a la navegabilidad. Tenemos el documento:

```
<html>

  <head>
    <title>DOM Tutorial</title>
  </head>

  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>

</html>
```



En el HTML anterior puedes leer:

<html> es el nodo raíz
<html> no tiene padres
<html> es el padre de <head> y <body>
<head> es el primer hijo de <html>
<body> es el último hijo de <html>

y:

<head> tiene un hijo: <title>
<title> tiene un hijo (un nodo de texto): "DOM Tutorial"
<body> tiene dos hijos: <h1> y <p>
<h1> tiene un hijo: "DOM Lesson one"
<p> tiene un hijo: "¡Hello world!"

<h1> y <p> son hermanos

Puede utilizar las siguientes propiedades de nodo para navegar entre nodos con JavaScript:

- parentNode
- childNodes[nodenumber]
- firstChild
- lastChild
- nextSibling
- previousSibling

Por ejemplo el título puede ser accedido de esta manera:

```
myTitle = document.getElementById("demo").firstChild.nodeValue;
```

o de esta;

```
myTitle = document.getElementById("demo").innerHTML;
```

o esta:

```
myTitle = document.getElementById("demo").childNodes[0].nodeValue;
```

Buscar elementos

Las formas de acceder a un elemento son:

- document.**getElementById(id)** → Busca elemento por id
- document.getElementsByTagName(name) → Busca elemento por tag
- document.getElementsByClassName(name) → Busca elemento por clase

Ejemplo: dado el párrafo:

Hola

- document.getElementById(main_section)
- document.getElementsByTagName(p)
- document.getElementsByClassName(main_paragraph)

Otras formas de acceder

Modificar elementos

Propiedades

element.innerHTML = nuevo valor → cambiar el html interno de un elemento.
element.attribute = nuevo valor → cambia el valor del atributo
element.style.property = nuevo estilo → cambia el estilo
element.textContent = nuevo valor → cambia el contenido

Método

element.setAttribute(attribute, value) Cambia el valor del atributo de un elemento HTML

Crear, borrar, agregar, reemplaza y escribe elementos

document.createElement(element) → **Crea** un elemento en un documento HTML
document.appendChild(element) → **Agrega** un elemento en un documento HTML

Un ejemplo práctico:

```
const p = document.createElement("p");
p.textContent = "Este es un nuevo párrafo";
document.body.appendChild(p);
```

document.removeChild(element) → **Borra** un elemento en un documento HTML
document.replaceChild(new, old) → **Reemplaza** un elemento en un doc. HTML

document.write(text) → **Escribir** en el flujo de salida HTML

Eventos

Un evento es cualquier suceso que le ocurra a un elemento HtML Javascript puede darse cuenta de ese evento y reaccionar a este ejecutando el código programado.

Algunos eventos que pueden ocurrir podrían ser:

- Pulsar un botón
- Modificar un campo de texto
- Pulsar una tecla
- La página ha terminado de cargarse
- Pasar sobre un elemento HTML

La forma de programar en el elemento HTML el evento sería la siguiente:

```
<elemento_HTML evento=’código_Javascript’>
```

También podemos crear el evento desde código javascript con el método addEventListener():

```
elemento.addEventListener("evento", () => {
  alert("Has hecho clic");
});
```

elemento es cualquiera del DOM.

Eventos comunes:

- click
- input
- submit
- mouseover, mouseout
- keydown, keyup

Para conocer todos los eventos visita w3School.

ANEXO IX. Fechas

Las fechas en JavaScript se pueden visualizar como número o como string. Veamos un ejemplo:

```
<p id=”fecha”></p>
<script>
document.getElementById(“fecha”).innerHTML = Date();
</script>
```

Esto devuelve algo como:

Mon Jun 21 2025 10:20:11 GMT+0001 (CEST)

Creación de un objeto tipo fecha

Existen cuatro formas de crear un objeto tipo fecha:

new Date();
new Date(milisegundos);
new Date(fechaString);
new Date(año, mes día, hora, minuto, segundo, milisegundos);

Algunos ejemplos:

```
var v = new Date();
var v = new Date(92500000);
var v = new Date(“08/16/2020”);
var v = new Date(2025, 11, 5, 15, 20, 12, 01);
```

Formatos de tipo fecha

Existen cuatro tipos.

Tipo	Ejemplo
Formato corto	var v = new Date(“01/10/2025”);
Formato largo	var v = new Date(“Ago 16 2025”);
Formato completo	var v = new Date(“Fri Ago 16 2019 12:20:15 GMT+0100 (W. Europe Standard Time)”);
Formato ISO	var v = new Date(“2025-08-16”);

Métodos de fechas

Los métodos para manejar las fechas son los siguientes:

Método	Descripción
getDate()	Obtiene el día del mes.
getDay()	Obtiene el día de la semana en forma numérica (0-6)
getFullYear()	Obtiene el año (ejemplo 2025)
getHour()	
getMinutes()	
getSeconds()	
getMonth()	
getTime()	Obtiene la hora en milisegundos desde 1 enero de 1970.

ANEXO X. Funciones setTimeout y setInterval

Hay unos métodos útiles para tratar el tiempo con Javascript. Son:

Método	Descripción	Ejemplo
setTimeout(funcion_a_llamar, milisegundos)	Ejecutará la funcion_a_llamar transcurido el tiempo indicado en milisegundos	var temp1 = setTimeout(laFuncion, 5000);
setInterval(funcion_a_llamar, milisegundos)	En este caso se ejecuta funcion_a_llamar de forma periódica según milisegundos.	var temp2 = setInterval(laFuncion, 5000);
clearTimeout()	Para la ejecución iniciada con setTimeout()	clearTimeout(temp1);
clearInterval()	Para la ejecución iniciada con setInterval()	clearInterval(temp2);

ANEXO XI. Almacenamiento

Tenemos dos maneras de almacenar datos con JavaScript, usando Cookies, localStorage y sessionSotrage().

Cookies

Estas sirven para recordar información del usuario. Son pequeños ficheros que se almacenarán en una ruta del equipo y contendrá pares clave, valor, por ejemplo: usuario = David;

- La utilidad de las cookies son:
- Monitorizar actividad de usuarios
- Mantener opciones de configuraciones
- Almacenar variables del lado del clientes
- Identificación o autenticación

Las cookies se eliminan al cerrar el navegador. Si queremos que duren el en tiempo tendremos que darle una fecha de caducidad.

Creación de Cookies

Es tan fácil como:

```
document.cookie=”usuario=David”;
```

Puesto que la cookie se envía en la cabecera HTTP debe estar bien codificada, por lo que es recomendable hacer:

```
var usuarioCookie=”David”;
document.cookie=”usuario=” + encodeURIComponent(usuarioCookie);
```

Tanto encodeURIComponent como decodeURIComponen codifican y decodifican en HTML los caracteres especiales.

Si queremos que la cookie dure en el tiempo le daremos una fecha de expiración:

```
document.cookie=”usuario=David;expire=Sat, 6 Aug 2025 12:01:01 GMT”;
```

Leer Cookies

En document.cookie se almacenas un string con todas las cookies. Quedaría por nuestra parte tratar ese string.

Borrar Cookies

Es tan simple como volver a crear la cookie que queremos borrar con una fecha pasada. Pero no te olvides de indicar la ruta donde se encuentra guardada.

```
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;";
```

localStorage

Este objeto permite guardar y recuperar información sin importar la sesión. Se utiliza casi como las cookies y se puede usar con los métodos:

- ***localStorage.setItem(clave, valor)***: Se usa para almacenar la información en la variable.
- ***localStorage.getItem(clave)***: Para recuperar la información, se indica la clave.
- ***localStorage.removeItem(clave)***: Elimina la información de una clave.

Por ejemplo:

```
<html>
  <body>
    <p id="demo"></p>
      <script>
        localStorage.setItem(“usuario”, “David”);
        alert( localStorage.getItem(“usuario));
      </script>
  </body>
</html>
```

sessionStorage

Este objeto se usa para almacenar datos solo en la sesión actual. Una vez cerrada la sesión los datos se pierden.

Los métodos usados son los mismos que los de localStorage().