UD6 - Resumen

1. Introducción

BOM (Browser Object Model)

- **Objeto que representa el navegador** completo, no solo el documento
- Permite acceder a: historial de navegación, dimensiones de ventana, información del navegador, etc.
- No existe un estándar oficial, pero los navegadores modernos implementan métodos y propiedades similares

DOM (Document Object Model)

- **Interfaz para acceder y modificar** dinámicamente el contenido, estructura y estilo del documento
- Representa el documento HTML como un árbol de nodos

Relación BOM - DOM

```
BOM (Navegador completo)
  ↓
DOM (Documento actual)
```

---

2. Modelo de Objetos del Navegador (BOM)

1) OBJETO WINDOW

Representa la ventana del navegador. Todas las variables, funciones y objetos globales son miembros de window.

```
// Obtener dimensiones de la ventana
let ancho = window.innerWidth;
let alto = window.innerHeight;
console.log(`Ancho x Alto: ${ancho} x ${alto}`);
```

Métodos principales de window:

Método	Utilidad
`alert()`	Ventana de alerta con botón Aceptar
`confirm()`	Diálogo con Aceptar/Cancelar (devuelve boolean)
`prompt()`	Diálogo para entrada de datos (devuelve string)
`open()`	Abre nueva pestaña/ventana
`close()`	Cierra la ventana actual
`setTimeout()`	Ejecuta código después de un tiempo
`setInterval()`	Ejecuta código repetidamente
`scrollTo()`	Desplaza el contenido a posición específica

Ejemplo práctico:

```
// Abrir ventana con parámetros
let ventana = open('', '', 'status=yes,width=400,height=250,menubar=yes');

// Confirmación
let respuesta = confirm("¿Estás seguro?");
if (respuesta) {
    alert("Aceptaste");
} else {
    alert("Cancelaste");
}

// Prompt
let nombre = prompt("Introduce tu nombre:");
alert(`Hola ${nombre}`);
```

2) OBJETO NAVIGATOR

Contiene información sobre el navegador del usuario.

```
// Propiedades útiles
console.log("Nombre:", navigator.appName);
console.log("Versión:", navigator.appVersion);
console.log("Cookies habilitadas:", navigator.cookieEnabled);
```

```
console.log("Idioma:", navigator.language);
console.log("Online:", navigator.onLine);
console.log("Plugins:", navigator.plugins.length);
```

Ejemplo completo:

```
document.write('VALORES DEL OBJETO NAVIGATOR:<br>');
document.write('Nombre oficial: ' + navigator.appName + '<br>');
document.write('Versión: ' + navigator.appVersion + '<br>');
document.write('Cookies activas: ' + navigator.cookieEnabled + '<br>');
document.write('Plugins: ' + navigator.plugins.length + '<br>');
```

3) OBJETO SCREEN

Información sobre la pantalla del usuario.

```
// Propiedades principales
console.log("Altura disponible:", screen.availHeight);
console.log("Anchura disponible:", screen.availWidth);
console.log("Altura total:", screen.height);
console.log("Anchura total:", screen.width);
console.log("Profundidad color:", screen.colorDepth);
```

4) OBJETO LOCATION

Representa la URL actual y permite redireccionar.

```
// Propiedades útiles
console.log("URL completa:", location.href);
console.log("Protocolo:", location.protocol);
console.log("Host:", location.host);
console.log("Ruta:", location.pathname);
console.log("Parámetros:", location.search);
console.log("Hash:", location.hash);

// Redireccionar
location.href = "<https://www.google.com>";
```

Ejemplo: Redirección con confirmación

```
function verificarEntrada() {
    if (confirm('¿Desea ir a YouTube?')) {
        location.href = '<https://www.youtube.com/>';
    } else {
        alert('No hay problema');
    }
}
```

5) OBJETO HISTORY

Gestiona el historial de navegación.

```
// Métodos principales
history.back();      // Página anterior (←)
history.forward();   // Página siguiente (→)
history.go(-1);      // Equivale a back()
history.go(1);       // Equivale a forward()
history.go(-2);      // Retrocede 2 páginas
```

Ejemplo práctico:

```
function avanzar() {
    if (history.length > 1) {
        history.go(1);
    } else {
        alert('No hay páginas en caché hacia adelante');
    }
}

function retroceder() {
    history.go(-1);
}
```

6) TEMPORIZADORES

setTimeout() - Ejecuta una vez

```
function saludar(nombre) {
    console.log(`Hola ${nombre}`);
}

// Ejecutar después de 3 segundos
setTimeout(saludar, 3000, "Juan");

// Con clearTimeout
const timeoutID = setTimeout(() => console.log("Este mensaje no aparecer
á"), 2000);
clearTimeout(timeoutID);  // Cancela la ejecución
```

setInterval() - Ejecuta repetidamente

```
let contador = 5;
const intervaloID = setInterval(() => {
    console.log(contador);
    contador--;
    if (contador < 0) {
        console.log("¡Tiempo!");
        clearInterval(intervaloID);  // Detiene el intervalo
    }
}, 1000);
```

---

3. Modelo de Objetos del Documento (DOM)

Estructura del DOM

El DOM representa el documento HTML como un árbol de nodos:

```
Document (nodo raíz)
 ↳ html
    ↳ head
    ↳ body
       ↳ h1, p, div, etc.
```

Tipos de Nodos (nodeType)

Valor	Tipo de Nodo	Constante
1	Elemento	ELEMENT_NODE
2	Atributo*	ATTRIBUTE_NODE
3	Texto	TEXT_NODE
8	Comentario	COMMENT_NODE
9	Documento	DOCUMENT_NODE
10	Tipo documento	DOCUMENT_TYPE_NODE

- Obsoletos en nuevos desarrollos

Selección de Elementos

a) Por ID

```
let elemento = document.getElementById("miId");
elemento.style.color = "red";
```

b) Por Etiqueta

```
let parrafos = document.getElementsByTagName("p");
console.log(parrafos.length);  // Cantidad de párrafos
console.log(parrafos[0]);      // Primer párrafo
```

c) Por Clase

```
let elementos = document.getElementsByClassName("miClase");
```

d) Por Selectores CSS

```
// querySelector - primer elemento que coincide
let primerElemento = document.querySelector(".miClase");

// querySelectorAll - todos los elementos que coinciden
```

```
let todosElementos = document.querySelectorAll("div > p");
let enlaces = document.querySelectorAll("span > a");
```

Navegación por el DOM

```
let elemento = document.querySelector("li");

// Propiedades de navegación
elemento.parentNode;          // Nodo padre
elemento.childNodes;          // Lista de hijos
elemento.firstChild;          // Primer hijo
elemento.lastChild;           // Último hijo
elemento.previousSibling;     // Hermano anterior
elemento.nextSibling;         // Hermano siguiente

// Versiones solo para elementos
elemento.parentElement;
elemento.children;
elemento.firstElementChild;
elemento.lastElementChild;
elemento.previousElementSibling;
elemento.nextElementSibling;
```

Ejemplo: Navegación entre hermanos

```
let mediano = document.getElementsByTagName("li")[1];
console.log('Anterior:', mediano.previousElementSibling);
console.log('Siguiente:', mediano.nextElementSibling);
```

Manipulación de Contenido

textContent vs innerHTML

```
let seccion = document.getElementsByTagName("section")[0];

// textContent - solo texto
console.log(seccion.textContent);
```

```
// "Manipular elementos parece sencillo pero debe puede complicarse bast
ante."

// innerHTML - incluye etiquetas
console.log(seccion.innerHTML);
// "Manipular elementos <i>parece</i> sencillo<br />pero <strike>debe</s
trike> puede complicarse bastante."
```

Creación e Inserción de Elementos

Pasos para crear un elemento:

1. **Crear elemento**
2. **Crear nodo de texto**
3. **Vincular texto al elemento**
4. **Insertar en el documento**

```
// Paso 1: Crear elemento
var parrafo = document.createElement("p");

// Paso 2: Crear contenido
var contenido = document.createTextNode("Síntesis del DOM");

// Paso 3: Vincular contenido al elemento
parrafo.appendChild(contenido);

// Paso 4: Insertar en el documento
document.body.appendChild(parrafo);
```

Métodos de inserción:

```
let nuevoElemento = document.createElement("li");
nuevoElemento.textContent = "Nuevo elemento";

// appendChild - al final
lista.appendChild(nuevoElemento);
```

```
// insertBefore - en posición específica
lista.insertBefore(nuevoElemento, referencia);

// replaceChild - reemplazar
lista.replaceChild(nuevoElemento, viejoElemento);
```

Eliminación de Elementos

```
let elemento = document.getElementById("eliminar");
elemento.parentNode.removeChild(elemento);

// Método moderno (ES6+)
elemento.remove();
```

Manipulación de Atributos

```
let imagen = document.getElementsByTagName("img")[0];

// Comprobar atributo
imagen.hasAttribute("src");       // true/false

// Obtener atributo
imagen.getAttribute("alt");

// Establecer atributo
imagen.setAttribute("alt", "Nueva descripción");

// Eliminar atributo
imagen.removeAttribute("title");

// Alternar atributo
input.toggleAttribute("disabled");

// Lista de atributos
for (let attr of imagen.attributes) {
    console.log(`${attr.name}: ${attr.value}`);
}
```

Manipulación de Estilos

Modificación directa con style

```
let elemento = document.getElementById("miElemento");

elemento.style.color = "blue";
elemento.style.backgroundColor = "#f0f0f0";
elemento.style.fontSize = "16px";
elemento.style.border = "1px solid red";
```

Ejemplo completo:

```
const miCaja = document.getElementById("mi_Caja");
miCaja.style.width = "30%";
miCaja.style.height = "250px";
miCaja.style.border = "5px dashed #00FF00";
miCaja.style.paddingLeft = "15px";
miCaja.style.marginBottom = "25px";
```

Trabajar con clases CSS

```
let elemento = document.getElementById("miElemento");

// className (reemplaza todas las clases)
elemento.className = "clase1 clase2";

// classList (manipulación granular)
elemento.classList.add("nuevaClase");
elemento.classList.remove("claseVieja");
elemento.classList.toggle("activo");
elemento.classList.contains("clase");  // true/false
elemento.classList.replace("vieja", "nueva");

// Recorrer clases
for (let clase of elemento.classList) {
```

```
    console.log(clase);
}
```

Atributos Data Personalizados

```
<ul id="precios" data-divisa="euro" data-iva="21">
    <li>14.95</li>
    <li>95.99</li>
</ul>
```

```
let ul = document.getElementById("precios");

// Acceder a datos
console.log(ul.dataset.divisa);  // "euro"
console.log(ul.dataset.iva);     // "21"

// Modificar datos
ul.dataset.iva = 10;

// Recorrer todos los data attributes
for (let key in ul.dataset) {
    console.log(`${key}: ${ul.dataset[key]}`);
}
```

Ejemplo: Modificación de enlaces

```
let enlaces = document.getElementsByTagName("a");
for (let enlace of enlaces) {
    if (enlace.hasAttribute('class')) {
        enlace.setAttribute("class", "nuevoEnlace");
    } else {
        enlace.setAttribute("class", "enlace");
    }
}
```

4. Cookies

¿Qué son las Cookies?

- Pequeños archivos (≤ 4kB) almacenados en el navegador
- Permiten recordar información entre sesiones
- Máximo 20 cookies por dominio

Crear y Leer Cookies

```
// Crear cookies básicas
document.cookie = "idioma=es";
document.cookie = "usuario=Juan";
document.cookie = "tema=oscuro";

// Leer todas las cookies
console.log(document.cookie);
// "idioma=es; usuario=Juan; tema=oscuro"
```

Cookies con Codificación

```
// Codificar valores especiales
var usuario = "Juan Pérez";
document.cookie = "usuario=" + encodeURIComponent(usuario);

// Decodificar
var valor = decodeURIComponent(cookieValue);
```

Cookies Persistentes

```
// Con max-age (segundos)
let unAnio = 60 * 60 * 24 * 365;
document.cookie = `usuario=Juan; max-age=${unAnio}`;

// Con expires (fecha GMT)
document.cookie = "usuario=Juan; expires=Sat, 01 Jan 2025 12:00:00 GM
T";
```

```
// Especificar ruta
document.cookie = "usuario=Juan; path=/";
```

Eliminar Cookies

```
// Establecer fecha pasada o max-age=0
document.cookie = "usuario=; max-age=0";
document.cookie = "usuario=; expires=Thu, 01 Jan 1970 00:00:00 GMT";
```

Ejemplo completo:

```
// Crear cookies
document.cookie = "nombre=Bea";
document.cookie = "comida_preferida=jamón";

// Función para mostrar cookies
function mostrarCookies() {
    alert(document.cookie);
}

// Botón para mostrar
<button onclick="mostrarCookies()">Mostrar cookies</button>
```

Buenas Prácticas DOM:

1. **Cachear elementos** que se usen repetidamente
2. **Usar event delegation** para elementos dinámicos
3. **Minimizar reflows** agrupando modificaciones
4. **Prefiere classList** sobre className para manipulación granular

Seguridad Cookies:

1. **Nunca almacenar datos sensibles** en cookies
2. **Usar HttpOnly** para cookies de sesión
3. **Considerar SameSite** para protección CSRF

Compatibilidad:

1. **Verificar soporte** de métodos modernos
2. **Usar polyfills** cuando sea necesario
3. **Probar en múltiples navegadores**