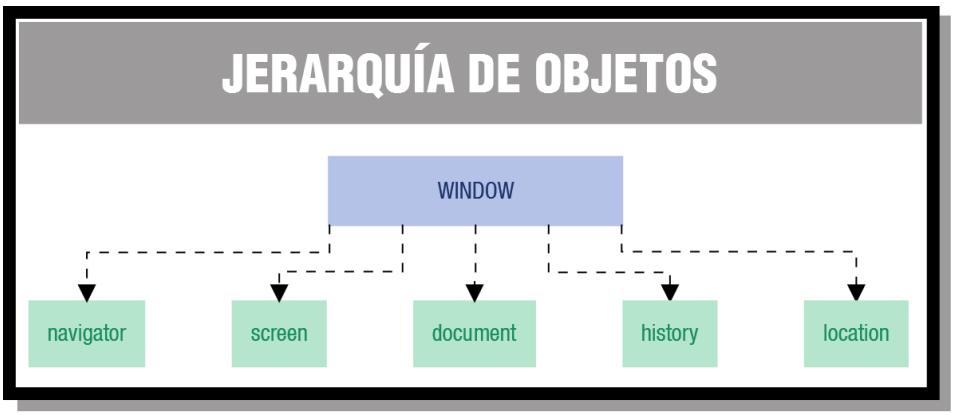
Tema 2. Utilización de objetos predefinidos en JavaScript.

Boletín de actividades

Contenido

2. El modelo de objetos del navegador.

El BOM (Browser Object Model o Modelo de Objetos del Navegador) es un conjunto de objetos creados por el navegador y que se pueden consultar y modificar desde JavaScript. Este esquema muestra los objetos que forman parte del BOM:



Hay que tener en cuenta que no existe una especificación común para el BOM y su implementación depende del navegador que se esté utilizando. Puede haber por tanto, métodos, propiedades o comportamientos que varíen de un navegador a otro.

Referencias:

- Artículo The Browser Object Model en W3Schools.

Objeto `Window`

El objeto `Window` representa la ventana del navegador. Cada pestaña y cada marco también tendrán asociados un objeto `Window`.

Actividad 1. Propiedades de las ventanas.

Implementa una aplicación que consulte los siguientes valores de la ventana actual del navegador:

- Coordenadas superior e izquierda de la ventana del navegador con respecto a la pantalla del equipo. Propiedades: `screenX/screenLeft` y `screenY/screenTop`.
- La altura y anchura de la página. Propiedades: `innerHeight/clientHeight` e `innerWidth/clientWidth`.
- La altura y anchura de la ventana del navegador. Propiedades: `outerHeight` y `outerWidth`.

Referencias:

- Artículo The Window Object en W3Schools.

Actividad 2. Abrir y cerrar ventanas.

Implementa una aplicación que contenga 2 botones, uno para abrir una nueva ventana (método `open`) y otro para cerrarla (método `close`). Para llamar a una función al hacer click en el botón utiliza el atributo HTML `onclick` (artículo HTML onclick Event Attribute de W3Schools). Un desarrollo más profundo de la gestión de eventos será estudiado en la próxima unidad.

La ventana debe abrirse con las siguientes propiedades:

- 600px de ancho y 450px de alto (Propiedades: `widht` y `height`)
- 50px de distancia al borde superior y 50px al borde izquierdo (Propiedades: `top` y `left`)
- El menú del navegador sea visible (propiedad `menubar`)
- Se pueda redimensionar (propiedad `resizable`)
- Se muestre la barra de direcciones (propiedad `location`)
- Aparezcan las barras de desplazamiento (propiedad `scrollbars`)
- Sea visible la barra de estado (propiedad `status`)
- Se muestre la barra de herramientas (propiedad `toolbar`)

Referencias:

- Artículo Window open() Method de W3Schools.
- Artículo Window close() Method de W3Schools.

Actividad 3. Redimensionar y mover ventanas.

Implementa una aplicación con 5 botones que realicen las siguientes tareas:

1. Abrir una nueva ventana de 100 x 100 píxeles.
2. Aumentar el ancho y alto de la nueva ventana en 250px. Método `resizeBy`.
3. Fijar el ancho y alto de la nueva ventana a 250 x 250 píxeles. Método `resizeTo`.
4. Mover la nueva ventana 250 píxeles a la derecha y 250 píxeles abajo. Método `moveBy`.
5. Mover la nueva ventana 500 píxeles a la derecha de la pantalla y 200 píxeles abajo. Método `moveTo`.

Nota: En algunos navegadores estos métodos sólo funcionan cuando se aplican sobre una ventana abierta con JavaScript.

Referencias:

- Artículo The Window Object en W3Schools.

Actividad 4. Ventanas emergentes del navegador.

El objeto `Window` también tiene métodos para crear ventanas emergentes del navegador. Se pueden crear 3 tipos de ventanas con: `alert`, `confirm` y `prompt`.

4.1. Implementa una aplicación que muestre al usuario este mensaje informativo: “A partir de ahora la navegación será privada” (método `alert`).

4.2. Implementa una aplicación que pregunte al usuario si acepta los términos y condiciones de la página. Si lo acepta se mostrará por consola el mensaje “Términos y condiciones aceptados”, si no “No se han aceptado los términos y condiciones” (método `confirm`).

4.3. Implementa una aplicación que pida al usuario su nombre siendo “Harry Potter” el valor por defecto. Si el usuario no introduce nada o cancela la ventana se mostrará por consola “El usuario ha cancelado la ventana” en caso contrario le saludará con “¡Hola nombre introducido!” (método `prompt`)

Referencias:

- Artículo JavaScript Popup Boxes de W3Schools.

Actividad 5. Ejecución a intervalos.

El objeto `Window`, además de permitir la gestión de ventanas, incluye métodos para ejecutar código JavaScript en intervalos de tiempo. Estos métodos son importantes para la programación de juegos, animaciones y otras aplicaciones. Los dos métodos principales son:

- `setTimeout` que ejecuta una función después de un número de milisegundos.
- `SetInterval` que repite la ejecución de una función cada cierto número de milisegundos.

Implementa estas dos aplicaciones:

5.1. Implementa una aplicación que abra una nueva ventana y 5 segundos después de abrirla la cierre (método `setTimeout`). Añade un botón a la aplicación que pare el cierre de la ventana si se pulsa antes de los 5 segundos (método `clearTimeout`).

5.2. Implementa una aplicación que ponga en marcha una cuenta atrás de 10 segundos. Los segundos deben ir mostrándose por consola. (método `setInterval`). Añade un botón a la aplicación que pare la cuenta atrás cuando se pulse. (método `clearInterval`).

Referencias:

- Artículo JavaScript Timing Events de W3Schools.

Objeto Screen.

El objeto `Screen` nos proporciona información sobre la pantalla del usuario.

Actividad 1. Propiedades de la pantalla.

Implementa una aplicación que muestre la siguiente información sobre la pantalla del usuario:

- El ancho y alto de la pantalla. Propiedades `width` y `height`.
- El ancho y alto de la pantalla disponibles teniendo en cuenta los elementos de la interfaz como la barra de tareas del sistema operativo. Propiedades `availWidth` y `availHeight`.
- La profundidad de color de la pantalla del usuario. Propiedad `colorDepth/pixelDepth`.

Referencias:

- Artículo JavaScript Window Screen de W3Schools.

Objeto Location

El objeto `Location` permite obtener la dirección URL de la página e incluye métodos para redireccionar a una nueva.

Actividad 1. Propiedades de la URL.

Implementa una aplicación que muestre la siguiente información sobre la URL de la página:

- La URL completa de la página. Propiedad `href`.
- El nombre de dominio. Propiedad `hostname`.
- La ruta y el nombre del fichero de la página. Propiedad `pathname`.
- El protocolo de comunicación utilizado. Propiedad `protocol`.

Referencias:

- Artículo JavaScript Window Location en W3Schools.

Actividad 2. Redirección y recarga de la página.

2.1. Implementa una aplicación que redireccione la página actual a la página de Google guardando la página actual en el historial de navegación (método `assign`). A continuación realiza la misma tarea pero sin guardar la página actual en el historial de navegación (método `replace`).

2.2. Implementa una aplicación que incluya un botón que permita recargar la página actual realizando la misma función que el botón del navegador. (método `reload`)

Objeto **`History`**

El objeto `History` permite navegar por el historial de navegación del navegador.

Hay tres métodos principales para moverse por el historial:

- `back` con el mismo resultado que el botón atrás del navegador
- `forward` con el mismo resultado que el botón hacia delante del navegador.
- `go` permite ir a una url concreta que no esté inmediatamente antes o después.

Referencias:

- Artículo JavaScript Window History de W3Schools.

Objeto Navigator

El objeto `Navigator` contiene información sobre el navegador del usuario.

Actividad 1. Propiedades del navegador.

Implementa una aplicación que muestre la siguiente información sobre el navegador del usuario:

- El valor de la cabecera HTTP user-agent enviada por el navegador al servidor. Propiedad `userAgent`.
- Si las cookies están habilitadas o no. Propiedad `cookieEnabled`.
- El sistema operativo sobre el que corre el navegador. Propiedad `platform`.
- El lenguaje del navegador. Propiedad `language`.

Referencias:

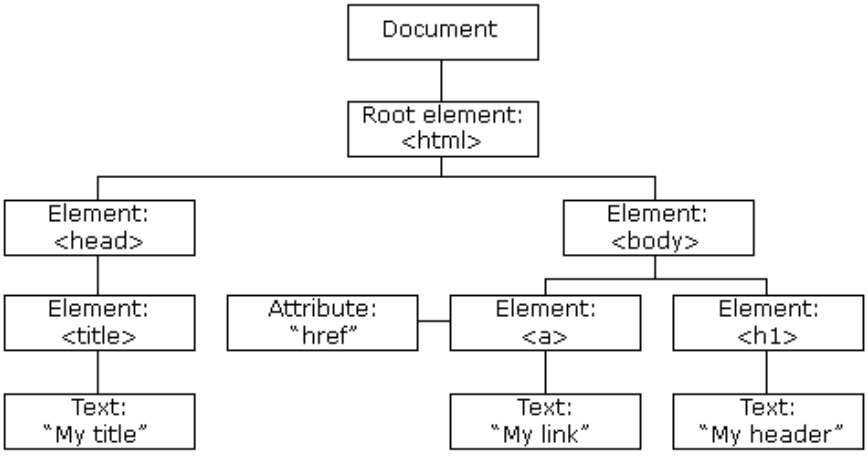
- Artículo JavaScript Window Navigator de W3Schools.

Objeto Document

El objeto `Document` representa la página HTML cargada en el navegador.

Se conoce como DOM al conjunto de objetos que representan cada uno de los elementos de una página HTML. Estos objetos se pueden consultar, añadir o eliminar desde JavaScript para hacer modificaciones en los elementos HTML de una página.

Este es un ejemplo del conjunto de objetos que proporciona el navegador para una página HTML que contiene únicamente un título, un enlace y una cabecera de primer nivel:



A continuación haremos una introducción a los principales métodos aunque por su importancia, el DOM se estudiará en detalle en la siguiente unidad.

Actividad 1. Recuperar elementos HTML.

Los principales métodos para recuperar elementos de un documento HTML son: `getElementById`, `getElementsByTagName`, `getElementsByClassName`, `querySelector y querySelectorAll`.

Crea un nuevo documento HTML con 4 párrafos (elementos `p`). Recupera el segundo párrafo utilizando los 5 métodos.

Actividad 2. Cambiar contenido.

Las principales propiedades para cambiar el contenido de elementos HTML son: `innerHTML` y `textContent`.

Cambia el contenido del segundo párrafo por "Éste es el nuevo contenido del segundo párrafo" y el contenido del tercero por "Éste es el nuevo contenido del tercer párrafo."

Actividad 3. Añadir y eliminar elementos.

Los principales métodos para añadir y eliminar elementos HTML son: `createElement` y `remove`.

Elimina el cuarto párrafo del documento. Crea un nuevo párrafo y añádelo en último lugar. Crea un nuevo párrafo y añádelo en tercer lugar.

Actividad 4. Otras propiedades y métodos.

a) Cambia el contenido de un párrafo por el contenido del título del documento.
b) Añade una imagen a un documento y cámbiala desde JavaScript.
c) Incluye un campo de texto de un formulario en un documento y cambia su valor por "Modificado desde JavaScript".
d) Utiliza el método `write` para escribir el contenido "Nuevo contenido escrito con write". ¿qué ocurre con el contenido previo de la página?
e) Realiza las siguientes operaciones sobre un párrafo:
   - Cambia a rojo el color del texto.
   - Añádele un fondo amarillo.
   - Cambia el tamaño del texto a 40 píxeles.
   - Ocúltalo.

3. Marcos

Para crear marcos en HTML5 se utiliza la etiqueta `iframe`. La etiqueta `frame` no está soportada por HTML5. Puedes ver un ejemplo de uso en el artículo HTML <frame> tag de W3Schools.

Actividad 1. Uso de marcos.

Actualmente se desaconseja el uso de marcos en una página web, ¿por qué?

Actividad 2. Creación de página con marcos.

Crea una página web en HTML5 con dos marcos para mostrar dos páginas web externas que también deben crearse. Ambas páginas deben contener un título y un párrafo que identifique a la página.

Actividad 3. Comunicación entre marcos.

- Añade un botón en la página del primer marco que cambie el contenido del párrafo de la página del segundo marco.

- Añade un botón en la página del segundo marco que cambie el contenido del título de la página del primer marco.

4. Objetos nativos de JavaScript

Objeto `String`

El objeto `String` es utilizado para almacenar y manipular textos. Por mantenibilidad siempre deberíamos usar *template Strings* para su creación.

Actividad 1. Propiedades y métodos.

A partir del fichero proporcionado en esta carpeta compartida. Implementa una aplicación que realice las siguientes tareas sobre la cadena:

```
I have become comfortably brilliant
```

```
La cadena cuenta con dos espacios en blanco delante y dos detrás.
```

1. Indica el número total de caracteres incluyendo los espacios en blanco (`length`)
2. Obtén el carácter que ocupa la octava posición (`charAt`)
3. Obtén el código Unicode del primer carácter (`charCodeAt`)
4. Concatena la cadena con la cadena " and exciting". ¿Se modifica la primera cadena? (`concat`)
5. Comprueba si la cadena termina con los caracteres "brillian" (`endsWith`)
6. Convierte el valor Unicode 65 a su carácter equivalente (`fromCharCode`)
7. Comprueba si la cadena contiene los caracteres "comfortably" (`includes`)
8. Indica la posición que ocupa el primer carácter "a" de la cadena. (`indexOf`)
9. Indica la posición que ocupa el último carácter "a" de la cadena (`lastIndexOf`)
10. Compara la cadena con la cadena "You `have become comfortably brilliant`". ¿Cuál iría en primer lugar? (`localeCompare`)
11. Obtén todas las coincidencias de la cadena con la expresión regular "/com/g" (`match`)
12. Obtén una nueva cadena con 3 repeticiones de la cadena actual (`repeat`)
13. Reemplaza los caracteres "brilliant" por "exciting" (`replace`)
14. Busca los caracteres "brit" en la cadena (`search`)
15. Obtén de la cadena los caracteres del primero al quinto (`slice`)
16. Obtén un array con todas las palabras de la cadena (`split`)
17. Comprueba si la cadena comienza con los caracteres "I have" (`startsWith`)
18. Obtén siete caracteres de la cadena a partir del segundo carácter (`substr`)
19. Obtén todos los caracteres de la cadena a partir del cuarto carácter (`substring`)
20. Convierte todos los caracteres de la cadena a mayúsculas (`toUpperCase`)
21. Convierte todos los caracteres de la cadena a minúsculas (`toLowerCase`)
22. Quita los espacios en blanco de la cadena por delante y detrás (`trim`)

Referencias:

- Artículo JavaScript String Reference en W3Schools.

Actividad 2. Cifrado.

Implementa una aplicación que pida al usuario un mensaje y una clave numérica y a continuación use el alfabeto Unicode para codificar el mensaje desplazando cada carácter un número de veces igual al número que corresponda en la clave. Mira el siguiente ejemplo:

```
Mensaje:            Este es el mensaje
Clave:              12345
Mensaje cifrado:    Fuwi%fu#iq!ohrxblh
```

El mensaje cifrado se forma de esta manera:

- E + 1 = F
- s + 2 = u
- t + 3 = w
- e + 4 = i
- (espacio en blanco) + 5 = %
- e + 1 = f
- s + 2 = u
- etc.

Nota: Utiliza las siguientes propiedades y funciones del objeto `String`: `charCodeAt`, `charAt`, `fromCharCode`, `length`.

Objeto `Math`

Actividad 1. Propiedades y métodos.

Utiliza las propiedades y métodos del objeto `Math` para realizar las siguientes tareas. Sigue el índice de W3Schools:

1. Obtén el resultado de multiplicar el número PI por el número E.
2. Obtén el valor absoluto de -9,87.
3. Redondea 1,2 al entero más alto.
4. Redondea 1,8 al entero más bajo.
5. Redondea 2.51 al entero más próximo.
6. Obtén el mayor número de los siguientes 6, 7, 4, 10, 10.1
7. Obtén el menor número de los siguientes 6, 7, 4, 3.9, 10
8. Obtén un valor aleatorio entre 100 y 200 (Puedes consultar MDN)
9. Obtén el valor de 2 elevado a 16.
10. Obtén el valor de la raíz cuadrada de 16.

Actividad 2. Círculo.

Implementa una aplicación que calcule la circunferencia y el área de un círculo a partir de su radio.

Objeto `Number`

Actividad 1. Propiedades y métodos.

Utiliza las propiedades y métodos del objeto `Number` para realizar las siguientes tareas sobre el número 123,123. Sigue el índice de W3Schools:

1. Comprueba si es un número.
2. Comprueba si el número es un entero.
3. Convierte el número a una cadena.
4. Formatea el número para que solo tenga un número decimal.

Objeto `Date`.

Actividad 1. Propiedades y métodos.

A partir del fichero proporcionado en esta carpeta compartida, realiza las siguientes actividades sobre el objeto Date:

1. El día del mes (getDate)
2. El día de la semana (getDay)
3. El año (getFullYear)
4. La hora (getHours)
5. Los milisegundos (getMilliseconds)
6. Los minutos (getMinutes)
7. El mes del año (getMonth)
8. Los segundos (getSeconds)
9. Una cadena legible de la fecha (toDateString)
10. Una cadena legible de la fecha en formato local (toLocaleDateString)
11. Una cadena legible de la hora en formato local (toLocaleTimeString)
12. Una cadena legible de la fecha y la hora en formato local (toLocaleString)

Referencias:

- Artículo JavaScript Date Reference de W3Schools.
- Artículo JavaScript Date Objects de W3Schools.

Actividad 2. Cálculo de tiempo hasta el mundial.

Calcula cuánto tiempo queda hasta el próximo mundial de fútbol. Indica años, meses y días.

5. Cookies

Actividad 1. Creación de funciones para facilitar la lectura y escritura de *cookies*.

En muchas ocasiones, cuando se crea una *cookie*, ésta contiene datos introducidos por el usuario desde un formulario. Es muy posible que, si se hacen peticiones GET al servidor, que algunos de los datos almacenados en las *cookies* se envíen en la URL como parte de una consulta (*?parametro1=valor1&parametro2=valor2&…*).

Ya sabemos que hay caracteres que no pueden formar parte de una URL. Por esta razón es bastante común utilizar métodos que codifiquen y decodifiquen los valores de las *cookies*. Concretamente se utilizan los métodos globales *encodeURIComponent* y *decodeURIComponent*.

Otra razón para la codificación y decodificación de las cadenas es que las *cookies* no aceptan punto y coma (;), comas (,) ni espacios en blanco ( ).

Implementa las siguientes funciones para facilitar el trabajo con las *cookies*:

- *setAndEncodeCookie(name, value, daysToLive=null)* – Función que crea una nueva *cookie*, codificando su valor.
  - *name* – nombre de la *cookie*.
  - *value* – valor de la *cookie*.
  - *daysToLive* – número de días que va tener de vida la *cookie* (servirá para establecer el valor del atributo *max-age* de la *cookie*).
- *getAndDecodeCookie(name)* – Función que obtiene una *cookie* dada y la decodifica, devolviendo su valor. Si la *cookie* no existe, devuelve *null*.
  - *name* – nombre de la *cookie*.
- *checkCookie(name)* – Función que comprueba si una *cookie* existe, en cuyo caso devuelve *true*. En caso contrario devuelve *false*.
  - *name* – nombre de la *cookie*.
- *getAndDecodeCookies()* – Función que devuelve un mapa (*Map*) con todas las *cookies* en su interior como pares clave-valor. Investiga cómo se usa el objeto *Map* y realiza la implementación de la función.

6. Almacenamiento local

Con *localStorage* y *sessionStorage* no tenemos el problema que teníamos con las *cookies*, se puede almacenar cualquier carácter sin tener que codificarlo.

Sin embargo, es normal almacenar mayores cantidades de datos en el almacenamiento local que en una *cookie*.

Si queremos almacenar *objetos literales* en lugar de elementos primitivos (*string*, *number*, etcétera), si lo hacemos en una *cookie*, es posible que nos quedemos sin espacio si los objetos que queremos almacenar son complejos.

Para almacenar objetos en *localStorage*, *sessionStorage* y como *cookies* se utilizan los métodos del Objeto *JSON* como hemos visto en los apuntes de clase. En el caso de usar *cookies*, seguramente debamos utilizar los métodos vistos en la actividad anterior para codificar y decodificar las cadenas.

En cualquier caso, hoy en día se recomienda usar *localStorage* y *sessionStorage* en lugar de *cookies* para almacenar los datos locales.

Actividad 1. Creación de funciones para facilitar la lectura y escritura de *cookies*.

Descarga el proyecto *Amazonia* de la carpeta compartida que representa a una página de compras con una lista de productos que puedes añadir o eliminar del carrito de compras.

Monta el proyecto en tu *IDE* favorito e implementa el código que se pide. Para tener claro qué debes implementar puedes guiarte por los comentarios que aparecen en el archivo. Tú debes implementar el código que está entre los comentarios:

```
//////////////////////COMIENZA EL CÓDIGO DEL ALUMNADO//////////////////////////

//////////////////////TERMINA EL CÓDIGO DEL ALUMNADO//////////////////////////
```

Notas importantes:

- Debes comentar el código de forma que sea más mantenible.
- Utiliza el depurador para entender el código que se suministra y facilitar el seguimiento de la tarea.
- En resumen, tienes que implementar:
    - Un trozo del código de la función *getProduct*.
    - La función completa *addProduct*.
    - La función completa *removeProduct*.
    - La función completa *printShoppingCart*.
    - La función completa *isProductStored*.