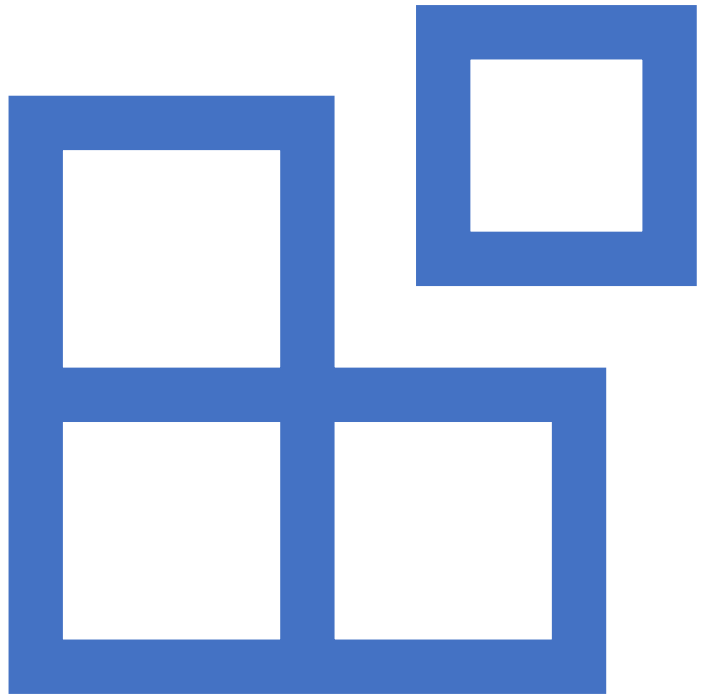
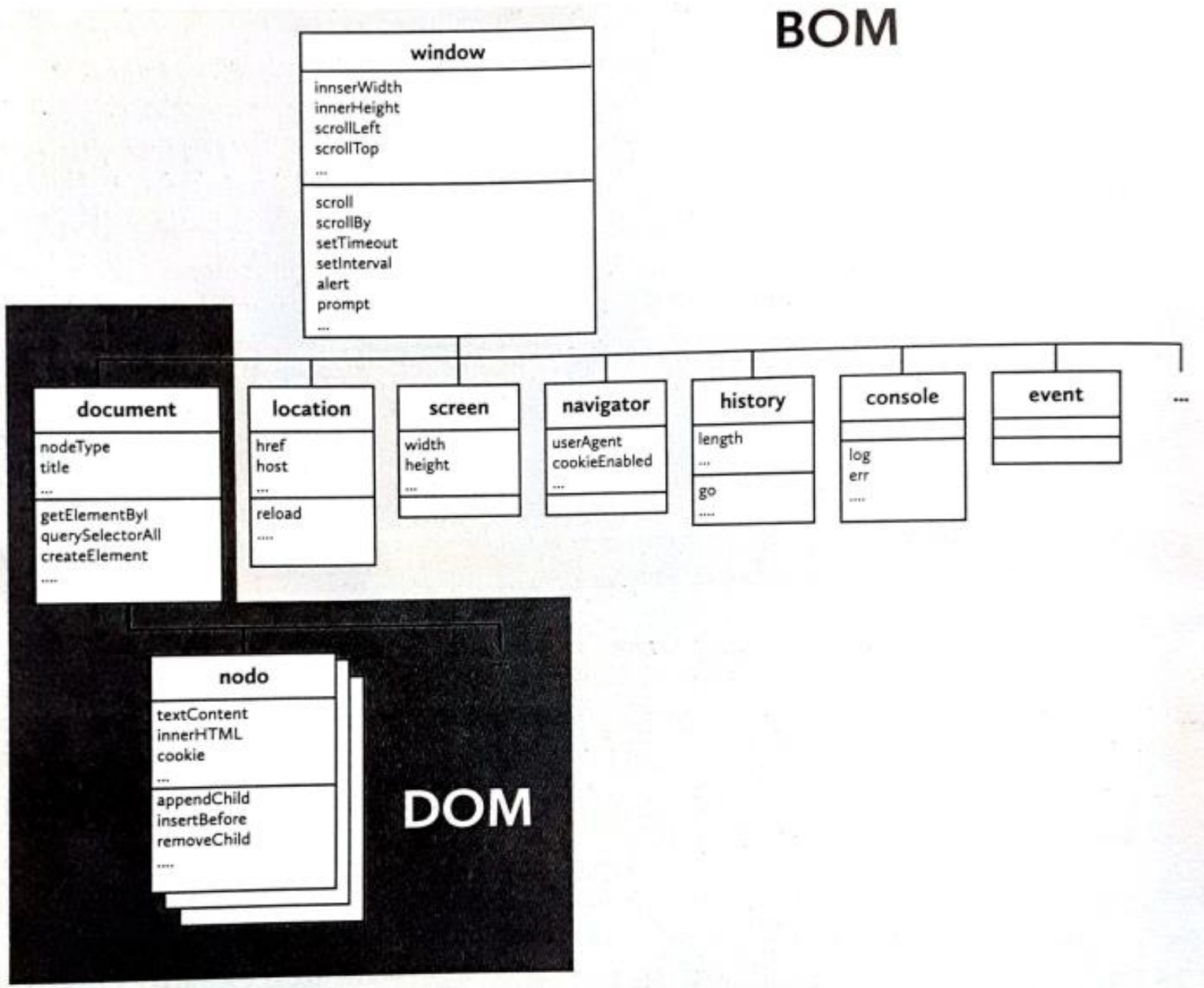


TEMA 7. BOM y DOM. Objeto window


BOM
window
innserWidth
innerHeight
scrollLeft
scrollTop
...
scroll
scrollBy
setTimeout
setInterval
alert
prompt
...
document
nodeType
title
...
getElementByI
querySelectorAll
createElement
....
location
href
host
...
reload
....
screen
width
height
...
navigator
userAgent
cookieEnabled
...
history
length
...
go
....
console
log
err
....
event
...
nodo
textContent
innerHTML
cookie
...
appendChild
insertBefore
removeChild
....
DOM

1. BOM y DOM

- Cuando se habla de JavaScript, quizá una de sus características más importantes es el acceso al **DOM** (Document Objects Model - Modelo de Objetos del Documento).
- Pues bien, realmente el DOM es parte de lo que se conoce como **BOM** (Browser Objects Model -Modelo de Objetos del Navegador) y que aglutina toda la estructura organizativa de los objetos del navegador.
- A partir de ahora, el navegador no es una simple pantalla, sino un conjunto de objetos que podemos consultar y manipular. En él tenemos objetos como el navegador, la barra de búsqueda, la consola, el documento, etc.

1. BOM y DOM

- Aunque el BOM no forma parte del estándar oficial del lenguaje JavaScript, lo cierto es que los distintos navegadores han igualado la forma de trabajar con estos objetos.
- Por lo tanto, podemos hablar de un uso estándar de los objetos del navegador.

2. window

- Hemos trabajado ya con varios métodos del objeto **window**, que resultan tan importantes que incluso se puede obviar su nombre al usar sus métodos y propiedades.
- Es el caso de **alert** y **prompt** donde su forma completa seria:
  - window.alert() → alert()
  - window.prompt() → prompt()

2. window

- Esta es una prueba de lo importante que es el objeto window en la creación de aplicaciones web.
- El objeto window es la propia ventana y es el elemento fundamental del manejo de las aplicaciones web desde JavaScript.
- Es también la raíz de toda la organización de objetos a los que JavaScript puede acceder para manipular todos los aspectos de una aplicación web.

2. Window

- Una característica fundamental del JavaScript, de hoy en día, es que se puede ejecutar en todo tipo de entornos, no solo en los navegadores.
- Pero, por ejemplo, **node.js** no dispone del objeto window porque es un entorno basado en la consola del navegador.
- Cuando usamos este objeto es señal de que estamos escribiendo una aplicación web en el lado del cliente.

2. window

- Como se ha dicho, el objeto window es el más importante. A partir de él se pueden crear nuevos objetos window que serán nuevas ventanas ejecutando el navegador.
- Tales ventanas 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, etc.

2.1 window.open

- El método window.open() recibe tres parámetros, que se colocan dentro de los paréntesis, de este modo:
  - window.open(URL,nombre_de_la_ventana,forma_de_la_ventana)
    - **URL**: Representa el URL que deseamos abrir en la ventana secundaria. Puedes abrir URLs de páginas que estén en tu dominio o en dominios externos.
    - **nombre_de_la_ventana**: es el nombre que se le asigna a esta ventana para dirigir enlaces con el atributo target del HTML
    - **forma_de_la_ventana**: se indica el aspecto que va a tener la ventana secundaria. Por ejemplo se puede definir su altura, anchura, si tiene barras de desplazamiento, etc.

2.1 window.open

- Estos atributos los puedes utilizar en la función window.open() para definir la forma que deseas que tenga tu ventana secundaria.

Atributo	Acción
Width	Ajusta el ancho de la ventana. En pixels
Height	Ajusta el alto de la ventana
Top	Indica la posición de la ventana. En concreto es la distancia en pixels que existe entre el borde superior de la pantalla y el borde superior de la ventana.
Left	Indica la posición de la ventana. En concreto es la distancia en pixels que existe entre el borde izquierdo de la pantalla y el borde izquierdo de la ventana.
Scrollbars	Para definir de forma exacta si salen o no las barras de desplazamiento. scrollbars=NO hace que nunca salgan. Scrollbars=YES hace que salgan
Resizable	Establece si se puede o no modificar el tamaño de la ventana. Con resizable=YES se puede modificar el tamaño y con resizable=NO se consigue un tamaño fijo.

2.1 window.open

- Veamos un ejemplo de sentencia JavaScript completa para abrir una ventana secundaria:

```
window.open("pagina2.html" , "ventana1" , "width=120,height=300,scrollbars=NO")
```

- Esto quiere decir que abrirá la pagina web “pagina2.html” en una ventana secundaria a la que vamos a llamar **ventana1**. Además, la ventana será de 120 pixels de ancho, 300 de alto y no tendrá barras de desplazamiento.
- Una aclaración adicional, si después de abrir esa ventana colocamos otro enlace en la página que abría la ventana cuyo atributo target está dirigido hacia el nombre_de_la_ventana (en este caso ventana1), este enlace se mostrará en la ventana secundaria.

2.1 window.open

- Vamos a realizar un caso práctico para ilustrar el método window.open.
- En una carpeta tenemos dos paginas “.html” pagina1.html y pagina2.html
- Ambas paginas contendrán su estructura general (head y body) y dentro del body vamos a incluir una etiqueta h1 para indicar en que pagina estamos.

2.1 window.open

pagina1.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=de
    <title>Pagina1</title>
</head>
<body>
    <h1>Estoy en la página 1</h1>
</body>
</html>
```

pagina2.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=de
    <title>Pagina2</title>
</head>
<body>
    <h1>Estoy en la página 2</h1>
</body>
</html>
```

2.1 window.open

- Creamos un fichero **prueba.js** donde irá alojado el código JavaScript
- Vamos a utilizar el método window.open dentro de una función dejando la URL como parámetro (así podemos llamarla con varias web distintas).

JS prueba.js ×

```
1   'use strict'
2   function ventanaSecundaria (URL){
3       window.open(URL,"ventana1","width=120,height=300,scrollbars=NO");
4   }
```

2.1 window.open

- Añadimos en enlace con la pagina “.js” en la pagina1.html

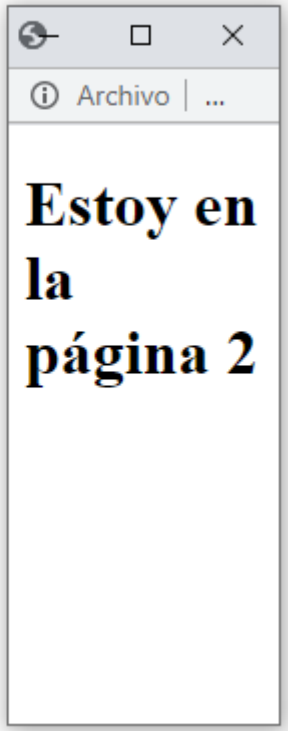
```
<script type="text/javascript" src="prueba.js"></script>
```

- Desde el body vamos a llamar a la función que hemos definido “**ventanasecundaria**” desde un enlace (a href) de nuestra pagina.

```
<body>
    <h1>Estoy en la página 1</h1>
    <a href="javascript:ventanaSecundaria('pagina2.html')">
        Pincha en este enlace para abrir la ventana secundaria</a>
</body>
```

2.1 window.open

- Ahora al hacer clic en el vinculo nos abrirá una nueva ventana con la pagina2.html con los modificadores que hemos indicado.



2.1 window.open

- En otras ocasiones desearemos abrir una ventana secundaria automáticamente, es decir, sin necesidad de que el usuario pulse sobre ningún enlace.
- En este caso, el código de la función **ventanaSecundaria** nos sirve también y habrá que añadir una línea de código Javascript a continuación de la función ventanaSecundaria.
- Esta línea a añadir simplemente será una llamada a la función que se ejecutará según se está cargando la página. Veamos como quedaría este código:

2.1 window.open

- **Prueba.js** nos quedaría de la siguiente forma:

```
'use strict'
function ventanaSecundaria (URL){
   window.open(URL,"ventana1","width=120,height=300,scrollbars=NO");
}

ventanaSecundaria('pagina2.html');
```

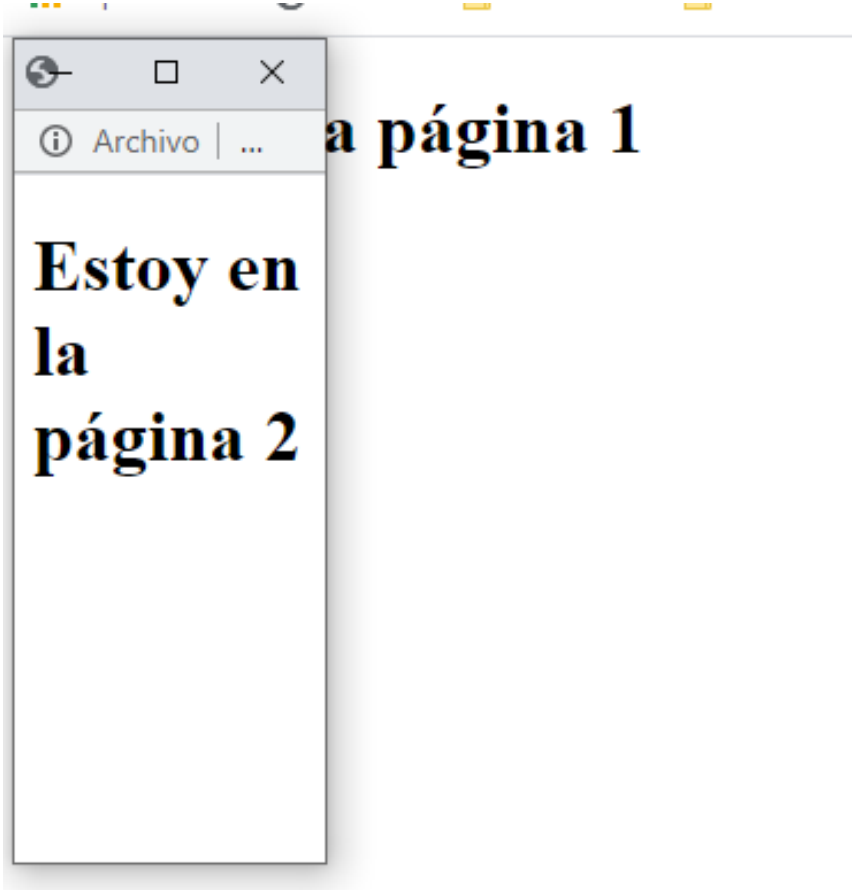
2.1 window.open

- En pagina1.html ahora ya no es necesario en enlace a la pagina2.html ya que se carga automáticamente al cargar la pagina.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Pagina1</title>
    <script type="text/javascript" src="prueba.js"></script>
</head>
<body>
    <h1>Estoy en la página 1</h1>
</body>
</html>
```

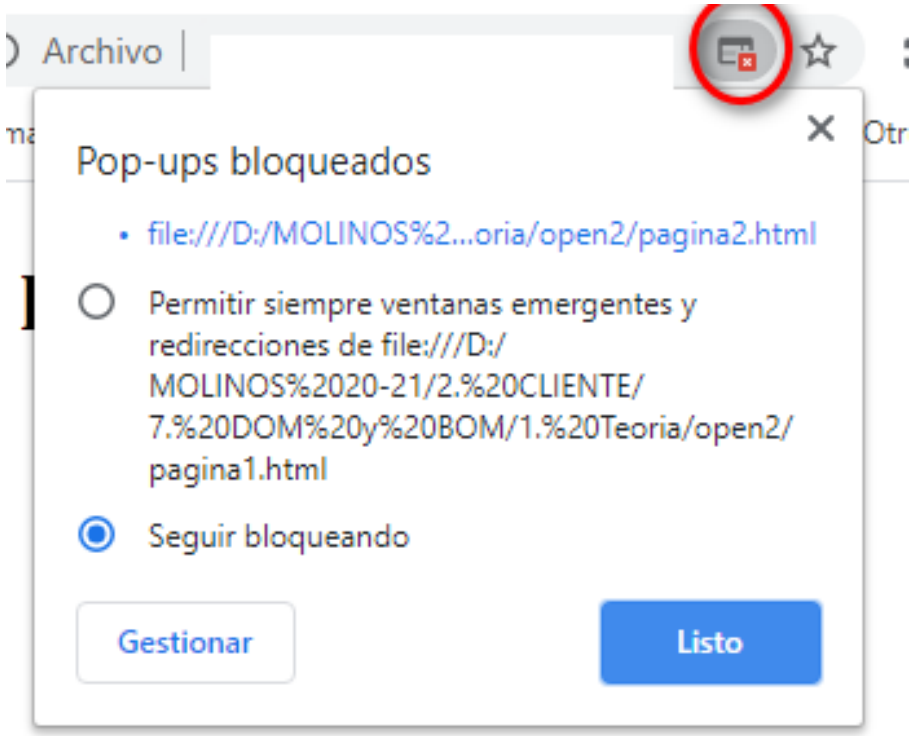
2.1 window.open

- Se nos abrirá automáticamente la otra pagina.



2.1 window.open

- Con este segundo sistema, corremos el riesgo que el navegador lo interprete como un intento de abrir spam que se carga a la vez que mi pagina, y nos bloqueará la ventana emergente.



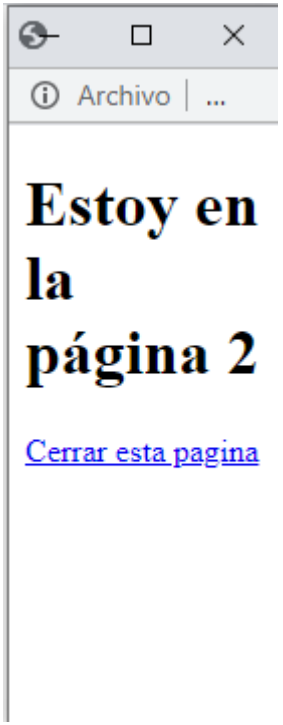
2.2 window.close

- De igual forma que abrimos ventanas nuevas, podemos cerrarlas también
- ***NOTA**: Hay veces que el navegador no permite el cierre de las paginas mediante este método*.
- Partiendo del ejemplo anterior, en pagina2.html vamos a añadir un enlace a window.close
- En este caso se puede incorporar directamente al html sin necesidad de utilizar una función en JavaScript como en el caso anterior.

2.2 window.close

- Contenido de pagina2.html

```
<body>
    <h1>Estoy en la página 2</h1>
    <a href="javascript:window.close()">Cerrar esta pagina</a>
</body>
```



2.3 Otros métodos de window

Método	Acción
alert()	
prompt()	
confirm()	Abre un cuadro emergente en el navegador, como parámetro se pasa el texto que aparecerá en el cuadro. El cuadro tiene dos botones: "aceptar" y "rechazar", en el que se espera una respuesta del usuario. Devuelve un valor booleano según sea la respuesta del usuario.
blur()	Quita el foco de la ventana actual.
focus()	Establece el foco en la ventana actual.
setTimeout()	Abre el temporizador para que una acción retrase su ejecución un cierto tiempo
setInterval()	Abre el temporizador para que una acción se repita cada cierto tiempo.
clearTimeout()	Detiene el temporizador abierto mediante el método setTimeout.
clearInterval()	Detiene el temporizador abierto mediante el método setInterval.
moveBy(x,y)	Mueve la ventana a la posición relativa indicada en los parámetros (coordenadas).

2.3 Otros métodos de window

moveTo(x,y)	Mueve la ventana a la posición absoluta indicada en los parámetros (coordenadas).
resizeBy(ancho,alto)	Aumenta o disminuye el tamaño de la ventana en los píxeles indicados en los parámetros (suma o resta -si es negativo- al tamaño actual).
resizeTo(ancho,alto)	Cambia el tamaño de la ventana a la altura y anchura indicada en los parámetros (medidos en píxeles).
open(URL,"nombre", propiedades)	Abre una nueva ventana emergente.
scrollBy(x,y)	Mueve el contenido de la ventana el número de píxeles indicado en los parámetros. Para que funcione el contenido de la página debe ser más grande que la ventana, de manera que tenga barras de desplazamiento.
scrollTo(x,y)	Mueve el contenido de la ventana a la posición indicada en los parámetros (coordenadas). El contenido debe ser más amplio que la ventana, de manera que tenga barras de desplazamiento.

2.3 Otros métodos de window

Método	Acción
print()	Imprime la página actual (abre el cuadro de diálogo de la impresora).
close()	Cierra la página actual.
moveTo(x,y)	Mueve la ventana a la posición absoluta indicada en los parámetros (coordenadas).

2.4 SetTimeOut y window.close

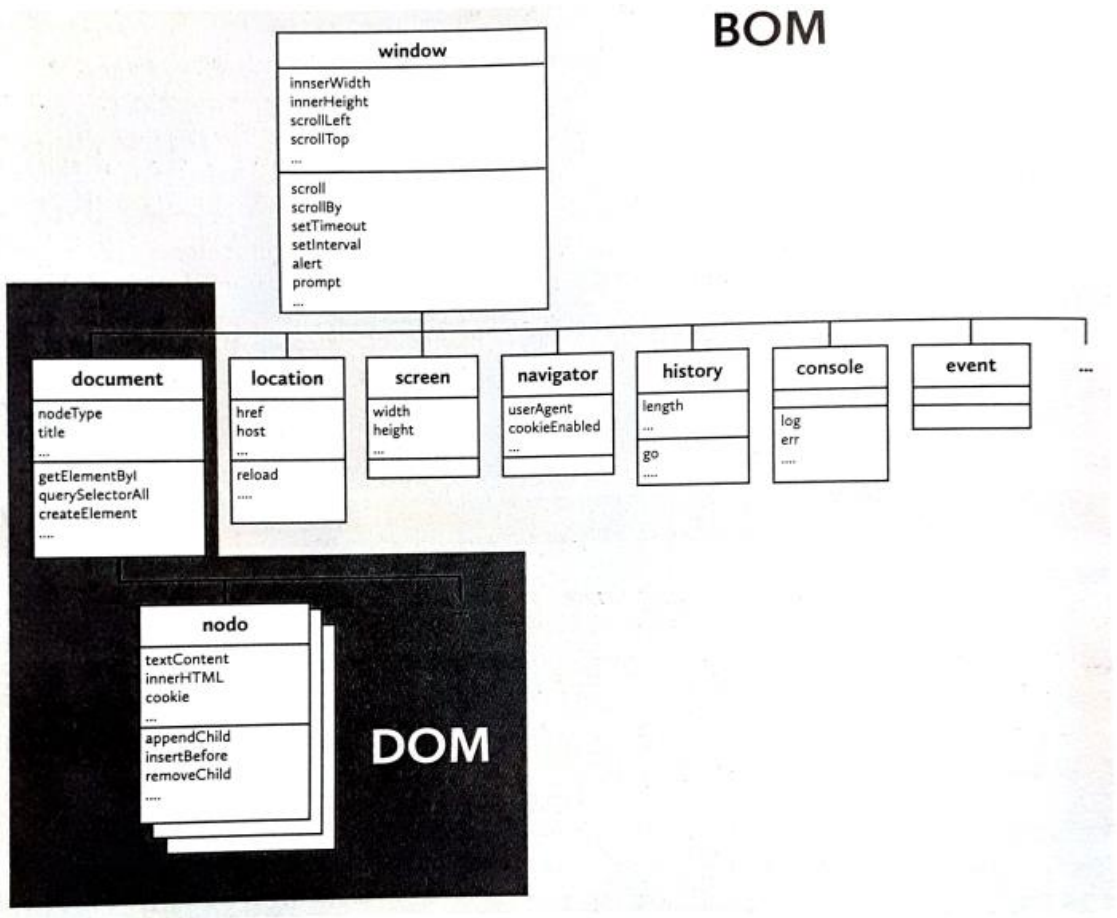
- Un ejemplo combinado de ambos.

```
function cerrarVentana() {
    window.setTimeout('window.close()',3000); // 3 segundos


}
```

- Si hacemos un enlace a esta función esperará 3 segundos (3000) y después cerrará la ventana (window.close()).

2.5 Nota final



- Hemos visto en este apartado algunos atributos y métodos de la clase window.
- Como puede observarse en el diagrama, el resto de clases (incluido el DOM) heredan de esta clase PADRE.
- Cuando trabajemos con ellas, incluso los métodos que hemos visto en este apartado, podemos omitir al padre para su uso.