UT3. Modelo de Objetos predefinidos en JavaScript

1) Objetos de más alto nivel en JavaScript
    1) Objeto window
    2) Objeto location
    3) Objeto navigator
    4) Objeto document

2) Objetos Nativos en JavaScript
    1) Objeto String.
    2) Objeto Math.
    3) Objeto Number.
    4) Objeto Boolean.
    5) Objeto Date.

1. Objetos de más alto nivel en JavaScript .DOM

- El **Modelo de Objetos del Documento (DOM)**, permite ver el mismo documento de otra manera, describiendo el contenido del documento como un **conjunto de objetos,** sobre los que un programa de Javascript puede interactuar.

- Según el W3C, el Modelo de Objetos del Documento es una **interfaz de programación de aplicaciones (API)**, para documentos válidos HTML.

- Define la **estructura lógica** de los documentos, y el modo en el que se acceden y se manipulan.

1. Objetos de más alto nivel en JavaScript (II).DOM

Algunas de las operaciones más comunes para las cuales se diseñó JavaScript son:

- Abrir una nueva ventana en el navegador
- Escribir un texto en un document
- Redirigir un navegador a otra ubicación
- Validar los datos de un formulario
- Etc

1. Objetos de más alto nivel en JavaScript (IV)

OBJETO: una entidad con una serie de **propiedades** que definen su estado, y unos **métodos (funciones)**, que actúan sobre esas propiedades.

- Para acceder a una **propiedad** de un objeto

```
nombreobjeto.propiedad
```

- La forma de acceder a un **método** de un objeto:

```
nombreobjeto.metodo([parámetros opcionales] )
```

También podemos referenciar a una propiedad de un objeto, por su **índice** en la creación. Los índices comienzan por 0.

1. Objetos de más alto nivel en JavaScript (V)

- Objetos de alto nivel que vamos a ver:

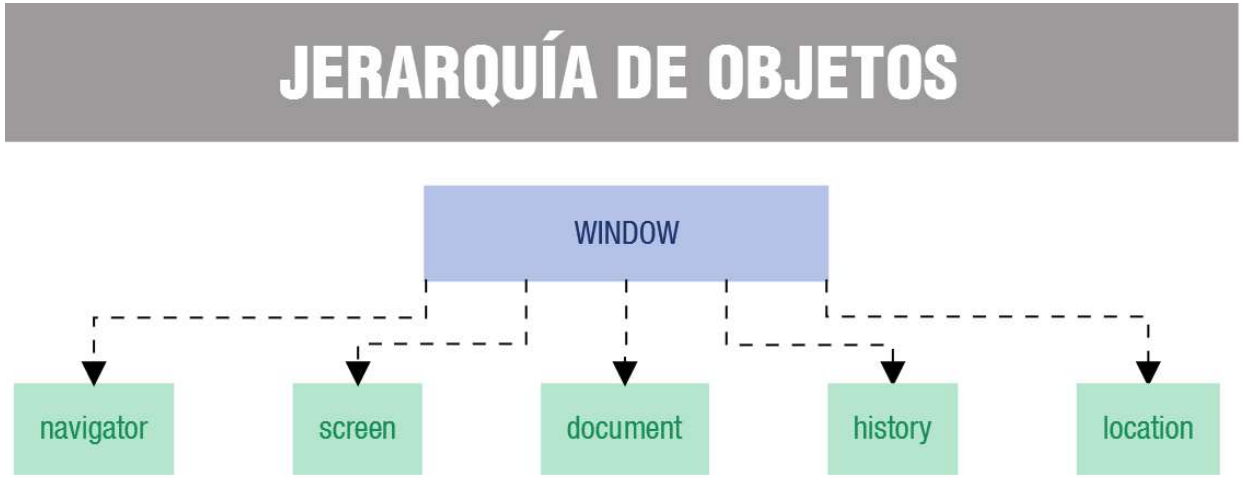
`window , location , navigator` y **`document`**

Se encuentran frecuentemente en las aplicaciones de JavaScript.

La mayoría de los navegadores web modernos han implementado la mayoría los mismos métodos y propiedades para la interacción JavaScript

1. Objetos de más alto nivel en JavaScript (VI)

- Gráfico del modelo de objetos de alto nivel, para todos los navegadores que permitan usar JavaScript:



1.1 Objeto window

- Situado en la parte superior de la jerarquía de objetos.
- Es el contenedor principal de todo el contenido que se visualiza en el navegador.
- El objeto **`window`** hace referencia a la ventana actual.
- Tan pronto como se abre una ventana (window) en el navegador, incluso aunque no se cargue ningún documento en ella, este objeto window ya estará definido en memoria.

1.1 Objeto window

Campo de **influencia de Objeto window**:

- Contenido de dicho objeto (donde se cargarán los documentos)
- Dimensiones de la ventana
- Todo lo que rodea al área de contenido: las barras de desplazamiento, barra de herramientas, barra de estado, etc.

1.1 Objeto window

- El objeto **`document`** será el que contendrá toda la jerarquía de objetos que tengamos dentro de nuestra página HTML.

1.1 Objeto window

En los navegadores más modernos, los usuarios tienen la posibilidad de abrir las páginas tanto en nuevas pestañas dentro de un navegador, como en nuevas ventanas de navegador.

- Para JavaScript tanto las **ventanas de navegador**, como las **pestañas**, son ambos **objetos window** .
- En los navegadores con pestañas, cada una contiene su propio objeto window.
- Esto significa que **el objeto window no se comparte entre diferentes pestañas de la misma ventana** del navegador

1.1 Objeto window

Acceso a propiedades y métodos

Hay varias formas de acceder, la forma que se elija dependiendo más de nuestro estilo, que de requerimientos sintácticos.

```
1.window.nombrePropiedad
 window.nombreMétodo( [parámetros] )
```

Los parámetros van entre corchetes, indicando que son opcionales y que dependerán del método al que estemos llamando.

1.1 Objeto window

Acceso a propiedades y métodos

2. **`self.nombrePropiedad`**
   `self.nombreMétodo( [parámetros]`

Se puede un objeto window desde el propio documento contenido en esa ventana.

Mejor dejar la palabra reservada **`self`** para scripts más complejos en los que tengamos múltiples ventanas.

1.1 Objeto window.

Acceso a propiedades y métodos

3. nombrePropiedad
nombreMétodo([parámetros]

Debido a que el objeto **window** siempre estará presente cuando ejecutemos nuestros scripts, podremos omitirlo, en referencias a los objetos dentro de esa ventana

También funcionaría sin ningún problema, porque se asume que esas propiedades o métodos, son del objeto de mayor jerarquía (el objeto **window**) en el cual nos encontramos.

1.1 Objeto window. Gestión de ventanas

- Un script no creará nunca la **ventana principal** de un navegador. Es el usuario, quien realiza esa tarea abriendo una URL en el navegador o un archivo desde el menú de abrir.

- Sin embargo, un script que esté ejecutándose en una de las ventanas principales del navegador, si que podrá crear o abrir nuevas **sub-ventanas**.

1.1 Objeto window
Gestión de ventanas

Métodos para abrir y cerrar ventanas

`window.open()`

- método que genera una nueva ventana
- Hasta 3 parámetros, que definen las características de la nueva ventana:
    - la URL del documento a abrir
    - el nombre de esa ventana
    - su apariencia física (tamaño, color,etc.).

`window.close()`

1.1 Objeto window. Gestión de ventanas

- Para manejar **sub-ventanas**: hacer asignación a una **variable**.

- Con esta asignación podremos a lo largo de nuestro código, referenciar a la nueva ventana desde el script original de la ventana principal

1.1 Objeto window. Gestión de ventanas

Ejemplo abrir y cerrar sub-ventanas:

```
var subVentana =
window.open("nueva.html","nueva","height=800,
width=600");

subVentana.close()
```

si escribiéramos `window.close()` , `self.close()` o `close()` estaríamos intentando cerrar nuestra propia ventana (previa confirmación), pero no la subVentana que creamos en los pasos anteriores

1.5.Comunicación entre múltiples ventanas

Objeto `window`: Método `open` y propiedad `opener`

Propiedad opener del objeto `window`: contiene la referencia a la ventana, que ha abierto ese objeto `window` empleando el método `open()` .
Para la ventana principal el valor de opener será `null`

- Debido a que `opener` es una referencia válida a la ventana padre que abrió las otras, podemos emplearlo para iniciar la referencia a objetos de la ventana original (padre) desde la ventana hija. Es semejante a lo que vimos con frames , pero en este caso es entre ventanas independientes del navegador.

1.5 Comunicación entre ventanas

Código HTML de padre.html:

** **

Código HTML de hijo1.html:

** **

Código HTML de hijo2.html:

** **

Ventana **padre.html** crea dos subventanas:

```
//utilizar var, let restringe la visibilidad del objeto al padre
var v1 = window.open("hijo1.html","hijo1");
var v2 = window.open("hijo2.html","hijo2");
```

1.5. Comunicación entre ventanas

Ventana **padre.html modifica un elemento de hijo1.html**:

v1.parHijo1.innerHTML=”Esto viene del padre”

Ventana **hijo1.html modifica un elemento de padre.html**:

opener.parPadre.innerHTML=”Esto viene del hijo1”

Ventana **hijo1.html modifica un elemento de hijo2.html**:

opener.v2.parHijo2.innerHTML=”Esto viene del hijo1

Para que este código funcione es necesario ejecutarlo desde un mismo servidor web, sino dará el error de directiva de seguridad cross_origin, ya que ejecutándolas con file:// desde el navegador considera que el origen de las 3 páginas es diferente, por lo que salta esta directiva de seguridad que no permite que una página modifique el elemento de la otra página.