PCW

PROGRAMACIÓN DEL CLIENTE WEB

Tema 05 - DOM y BOM


Dept. de Ciència de la Computació i Intel·ligència Artificial
Dpto. de Ciencia de la Computación e Inteligencia Artificial


Universitat d'Alacant
Universidad de Alicante

Document Object Model (DOM)

DOM (Document Object Model)

- Se trata de un API para acceder a documentos, especialmente HTML y XML, y poder manipularlos.
- Representa la estructura lógica del documento y su contenido en forma de árbol de nodos.
- Algunos nodos pueden tener hijos, mientras que otros son siempre hojas.
- El árbol de nodos se puede manipular mediante una serie de **interfaces**: EventTarget, Node, Document, Element, ...
- Los tipos de nodo más utilizados son:
  - **Document**: Nodo raíz del documento.
  - **Element**: Representa un elemento HTML. Es el único nodo que puede tener nodos hijo y nodos de atributo.

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Event**

Permite a los objetos avisar de que ha ocurrido algo.

```
interface Event { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  readonly attribute DOMString type; // nombre del evento disparado
  readonly attribute EventTarget? target; // elemento que lo genera
  readonly attribute EventTarget? currentTarget; // elemento actual

  // distintas fases por las que pasa el evento:
  const unsigned short NONE = 0;
  const unsigned short CAPTURING_PHASE = 1;
  const unsigned short AT_TARGET = 2;
  const unsigned short BUBBLING_PHASE = 3;
  readonly attribute unsigned short eventPhase; // fase del evento

  void stopPropagation(); // detiene la propagación del evento
  void stopImmediatePropagation(); // evita activar otros listeners
};
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Event**

```
interface Event { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // permite saber si el evento es propagable
  readonly attribute boolean bubbles;
  // permite saber si el evento es cancelable
  readonly attribute boolean cancelable;
  // detiene la acción por defecto asociada al evento
  void preventDefault();
  // permite saber si el evento ha sido cancelado?
  readonly attribute boolean defaultPrevented;
  // true si el evento lo genera una acción del usuario
  readonly attribute boolean isTrusted;
  // permite saber cuándo se disparó?
  readonly attribute DOMTimeStamp timeStamp;
  // permite inicializar un evento
  void initEvent(DOMString type, boolean bubbles, boolean cancelable);
};
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **EventTarget**

Representa el objeto sobre el que produce el evento.

```
interface EventTarget { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // añade un listener (manejador) para un evento del elemento
  void addEventListener(String nombreEvento, EventListener callback,
                        optional boolean faseCaptura = false);
  // elimina un listener para un evento del elemento
  void removeEventListener(String nombreEvento,EventListener callback,
                           optional boolean faseCaptura = false);
  // dispara el evento del elemento que se indique
  Boolean dispatchEvent(Event evento);
};

callback interface EventListener {
  void manejadorEvento(Event evento);
};
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **EventTarget**

Ejemplo:

HTML

```
<div id="t">
  <p id="t1">uno</p>
  <p id="t2">dos</p>
</div>
```

JavaScript

```
function modificarTexto(evt) {
   var t2 = document.getElementById('t2');
   t2.textContent = 'tres'; // cambia el texto de t2
}
// añadir listener al evento click del elemento t
var t = document.getElementById('t');
t.addEventListener('click', modificarTexto, false);
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **ParentNode**

```
interface ParentNode { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve la colección de nodos (elementos HTML) hijo
  readonly attribute HTMLCollection children;
  // devuelve el primer nodo hijo
  readonly attribute Element? firstElementChild;
  // devuelve el último nodo hijo
  readonly attribute Element? lastElementChild;
  // devuelve la cantidad de nodos hijo de tipo Element
  readonly attribute unsigned long childElementCount;
  // permite seleccionar nodos mediante selectores CSS
  Element? querySelector(DOMString selectors);
  NodeList querySelectorAll(DOMString selectors);
}
```

Nota: La interfaz **ParentNode** la implementan la interfaz **Document**, la interfaz **DocumentFragment** y la interfaz **Element**.

DOM (Document Object Model)

Tipos de nodos

- Interfaces **ChildNode**, **NodeList** y **HTMLCollection**

```
interface ChildNode { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // se borrar a sí mismo del padre
  void remove();
}

interface NodeList {// PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve el elemento de la posición indicada de la lista
  getter Node? item(unsigned long index);
  // devuelve el número de elementos en la lista
  readonly attribute unsigned long length;
}

interface HTMLCollection {// PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve el número de elementos en la colección
  readonly attribute unsigned long length;
  // devuelve el elemento de la posición indicada de la colección
  getter Element? item(unsigned long index);
  // devuelve el elemento con el id o name indicado
  getter Element? namedItem(DOMString name);
}
```

La interfaz **ChildNode** la implementan la interfaz **DocumentType**, la interfaz **Element** y la interfaz **CharacterData**.

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Node**

La implementan todos los tipos de nodo.

```
interface Node { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve el tipo de nodo: 1 - Element, 3 - Text, ...
  readonly attribute unsigned short nodeType;
  // devuelve el nombre del nodo (etiqueta HTML)
  readonly attribute DOMString nodeName;
  // devuelve URL base del documento html
  readonly attribute DOMString? baseURI;
  // devuelve el nodo <html> del documento al que pertenece
  readonly attribute Document? ownerDocument;
  // devuelve el nodo padre
  readonly attribute Node? parentNode;
  // devuelve el elemento padre (suele coincidir con parentNode)
  readonly attribute Element? parentElement;
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Node**

La implementan todos los tipos de nodo.

```
interface Node { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve true si tiene nodos hijo
  boolean hasChildNodes();
  // devuelve la lista de nodos hijo
  readonly attribute NodeList childNodes;
  // devuelve el primer nodo hijo
  readonly attribute Node? firstChild;
  // devuelve el último nodo hijo
  readonly attribute Node? lastChild;
  // devuelve el inmediatamente anterior nodo hermano
  readonly attribute Node? previousSibling;
  // devuelve el inmediatamente siguiente nodo hermano
  readonly attribute Node? nextSibling;
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Node**

```
interface Node { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve/establece el valor del nodo (Attr, Text, Comment)
  attribute DOMString? nodeValue;
  // devuelve/establece el contenido de texto del nodo y descendientes
  attribute DOMString? textContent;

  // compara la posición de los dos nodos en el documento:
  //  1 - DOCUMENT_POSITION_DISCONNECTED
  //  2 - DOCUMENT_POSITION_PRECEDING
  //  4 - DOCUMENT_POSITION_FOLLOWING
  //  8 - DOCUMENT_POSITION_CONTAINS
  // 16 - DOCUMENT_POSITION_CONTAINED_BY
  // 32 - DOCUMENT_POSITION_IMPLEMENTATION_SPECIFIC
  unsigned short compareDocumentPosition(Node other);
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Node**

```
interface Node { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // devuelve una copia del nodo. Si recibe true, el subárbol completo
  Node cloneNode(optional boolean deep = false);
  // devuelve true si el nodo que se pasa es igual
  boolean isEqualNode(Node? node);
  // devuelve true si el nodo que se le pasa es descendiente
  boolean contains(Node? other);
  // inserta el nodo node delante del nodo hijo child
  Node insertBefore(Node node, Node? child);
  // añade el nodo node como último hijo a la lista de nodos hijos
  Node appendChild(Node node);
  // sustituye el nodo hijo child por el nuevo nodo node
  Node replaceChild(Node node, Node child);
  // elimina el nodo hijo child
  Node removeChild(Node child);
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Document**

Es un documento XML hasta que se abre en el navegador y se convierte en un documento HTML.

```
interface Document : Node { // PROPIEDADES Y MÉTODOS MÁS UTILIZADOS
  // Devuelve la URL del documento
  readonly attribute DOMString URL;
  // Devuelve la URL del documento
  readonly attribute DOMString documentURI;
  // Devuelve la codificación del documento
  readonly attribute DOMString characterSet;
  // Devuelve el elemento raíz (<html>) del documento
  readonly attribute DOMString documentElement;
  // Crea el elemento HTML nombre
  Element createElement(DOMString nombre);
  // Crea un nodo de tipo texto cuyo contenido es datos
  Element createTextNode(DOMString datos);
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Document**

```
interface Document : Node { // MÉTODOS MÁS IMPORTANTES
  // crea una copia de un nodo desde un documento externo.
  // deep permite importar todo el árbol de descendientes.
  Node importNode(Node nodo, optional boolean deep = false);
  // Adopta un nodo y su árbol de descendientes eliminándolo del
  // document en el que estuviera, si fuera el caso
  Node adoptNode(Node nodo);
  // crea un evento del tipo especificado
  Event createEvent(DOMString tipoEvento);
  // Devuelve la lista de elementos HTML nombre
  HTMLCollection getElementsByTagName(DOMString nombre);
  // Devuelve lista de elementos con valor nomClase en atributo class
  HTMLCollection getElementsByClassName(DOMString nomClase);
  // Devuelve el elemento con el valor identificador en el atributo id
  Element getElementById(DOMString identificador);
  // Devuelve el primer elemento del subárbol seleccionado por selectores
  Element querySelector(DOMString selectores);
  // Devuelve una lista con todos los elemento del subárbol seleccionados por selectores
  NodeList querySelectorAll(DOMString selectores);
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Element**

Interfaz que implementan los elementos HTML. Hereda la clase *Node*.

```
interface Element : Node { // ATRIBUTOS MÁS IMPORTANTES
  // Devuelve el valor del atributo id
  readonly attribute DOMString id;
  // Devuelve el valor del atributo class
  readonly attribute DOMString className;
  // Devuelve la lista de valores del atributo class
  readonly attribute DOMTokenList classList;
  // Devuelve la lista de atributos del elemento
  readonly attribute NamedNodeMap attributes;
  // Devuelve/establece el fragmento HTML que representa el contenido
           attribute DOMString innerHTML;
  // Devuelve/establece el fragmento HTML que representa el propio
  // elemento y su contenido
           attribute DOMString outerHTML;
}
```

DOM (Document Object Model)

Tipos de nodos

- Interfaz **Element**

```
interface Element : Node { // MÉTODOS MÁS IMPORTANTES
  // Devuelve el valor del atributo nombre
  DOMString getAttribute(DOMString nombre);
  // Asigna el atributo nombre con el valor valor al elemento
  void setAttribute(DOMString nombre, DOMString valor);
  // Elimina el atributo nombre del elemento
  void removeAttribute(DOMString nombre);
  // Devuelve true si el elemento tiene el atributo nombre
  boolean hasAttribute(DOMString nombre); // DOM2
  // Devuelve la lista de elementos de tipo nombre descendientes
  NodeList getElementsByTagName(DOMString nombre);
  // Devuelve la lista de elementos de tipo nombre descendientes
  NodeList getElementsByClassName(DOMString nomClase);
}
```

Ejemplo de manipulación del atributo *class*

DOM (Document Object Model)

Ejemplo de manipulación del atributo *class*

HTML5 proporciona una nueva API llamada **`classList`**, gracias a la cual la manipulación del atributo *class* de los elementos del DOM se simplifica enormemente.

La nueva API proporciona a los elementos una nueva propiedad llamada **`classList`**. El valor de esta propiedad es un objeto que guarda todas las clases CSS asignadas al elemento y admite los siguientes métodos y propiedades para su manipulación:

- **`add`**`(clase)`. Añade la *clase* al elemento.
- **`remove`**`(clase)`. Elimina la *clase* del elemento.
- **`contains`**`(clase)`. Devuelve **`true`** o **`false`** en función de si el elemento tiene asignada la *clase* o no, respectivamente.

DOM (Document Object Model)

Ejemplo de manipulación del atributo *class*

(*Métodos y propiedades para su manipulación*)

- `toggle(clase)`. Añade la *clase* al elemento si no la tiene, o la quita si la tiene.
- `item(pos)`. Devuelve la clase que ocupa la posición *pos* en *classList*. Empieza en 0.
- `toString()`. Devuelve una cadena de texto con todas las clases del elemento separadas por espacios.
- `length`. Propiedad que devuelve el número de clases que tiene asignadas el elemento.

Al ser un objeto de JavaScript permite añadir nuevos métodos y propiedades personalizadas.

DOM (Document Object Model)

Ejemplo de manipulación del atributo *class*

Ejemplo:

```
<div id="dv01" class="aaa ccc ddd">
   …
</div>
```

```
var dv = document.getElementById("dv01");

console.log(dv.classList.length); // 3
dv.classList.add("bbb"); // class="aaa ccc ddd bbb"
console.log(dv.classList.length); // 4
dv.classList.remove("ddd"); // class="aaa ccc bbb"
dv.classList.toggle("hhh"); // class="aaa ccc bbb hhh"
dv.classList.toggle("hhh"); // class="aaa ccc bbb"
dv.classList.item(2); // "bbb"
dv.classList.contains("hhh"); // false
dv.classList.toString(); // "aaa ccc bbb“co
```

Ejemplo de selección de elementos mediante el atributo *class*

DOM (Document Object Model)

Ejemplo de selección mediante el atributo *class*

Mediante las interfaces proporcionadas por DOM, en HTML5 se pueden utilizar los selectores CSS para seleccionar elementos del documento y poder realizar acciones con ellos.

- **getElementsByClassName(***lista_de_clases***)**

  Aplicado a un elemento html, devuelve una lista de elementos cuyo atributo *class* contiene al menos los especificados en *lista_de_clases*. En *lista_de_clases* los distintos nombres de clases van separados por espacios en blanco.

HTML

```
<div id="ejemplo">
  <p id="p1" class="aaa bbb"/>
  <p id="p2" class="aaa ccc"/>
  <p id="p3" class="bbb ccc"/>
</div>
```

JavaScript

```
var n = document.getElementById('ejemplo');

var lista = n.document.getElementsByClassName('aaa'); // elementos #p1 y #p2
lista = n.document.getElementsByClassName('ccc bbb'); // elemento #p3
lista = n.document.getElementsByClassName('bbb ccc'); // elemento #p3
```

Ejemplo de selección de elementos mediante selectores CSS

DOM (Document Object Model)

Ejemplo de selección mediante selectores CSS

- **querySelector(*expresión*).** Aplicado a un elemento, devuelve el primer nodo dentro de su árbol que coincida con *expresión*. Si no hay ninguno que coincida, devuelve null.
- **querySelectorAll(*expresión*).** Aplicado a un elemento, devuelve una lista de nodos de su árbol que coincidan con *expresión*. La lista de nodos la devuelve como un objeto NodeList. Si no encuentra ningún nodo que coincida con la expresión, devuelve la lista vacía.

expresión es una cadena de texto que contiene uno o más selectores CSS separados por comas.

Ambos métodos se pueden aplicar al elemento **document**, a cualquier otro elemento del documento, y a cualquier fragmento de código html con el que se esté trabajando.

DOM (Document Object Model)

Ejemplo de selección mediante selectores CSS

- Ejemplos:

HTML

```
<div id="ejemplo">
  <p id="p1" class="aaa bbb"/>
  <p id="p2" class="aaa ccc"/>
  <p id="p3" class="bbb ccc"/>
</div>
<p id="p4" class="aaa bbb ccc">Pincha un botón</p>
```

```
var n = document.querySelector('#ejemplo'); // selecciona div#ejemplo
var e, lista;

lista = n.querySelectorAll('.aaa');         // elementos #p1 y #p2
lista = n.querySelectorAll('.ccc,.bbb');    // elementos #p1, #p2 y #p3
lista = n.querySelectorAll('*:not(.ccc)')   // elemento #p1
lista = document.querySelectorAll('.aaa'); // elementos #p1, #p2, #p3 y
#p4

e = n.querySelector('p');                 // elemento #p1
e = n.querySelector('p:nth-child(2)'); // elemento #p2
e = n.querySelector('p:last-child');    // elemento #p3
```

JavaScript

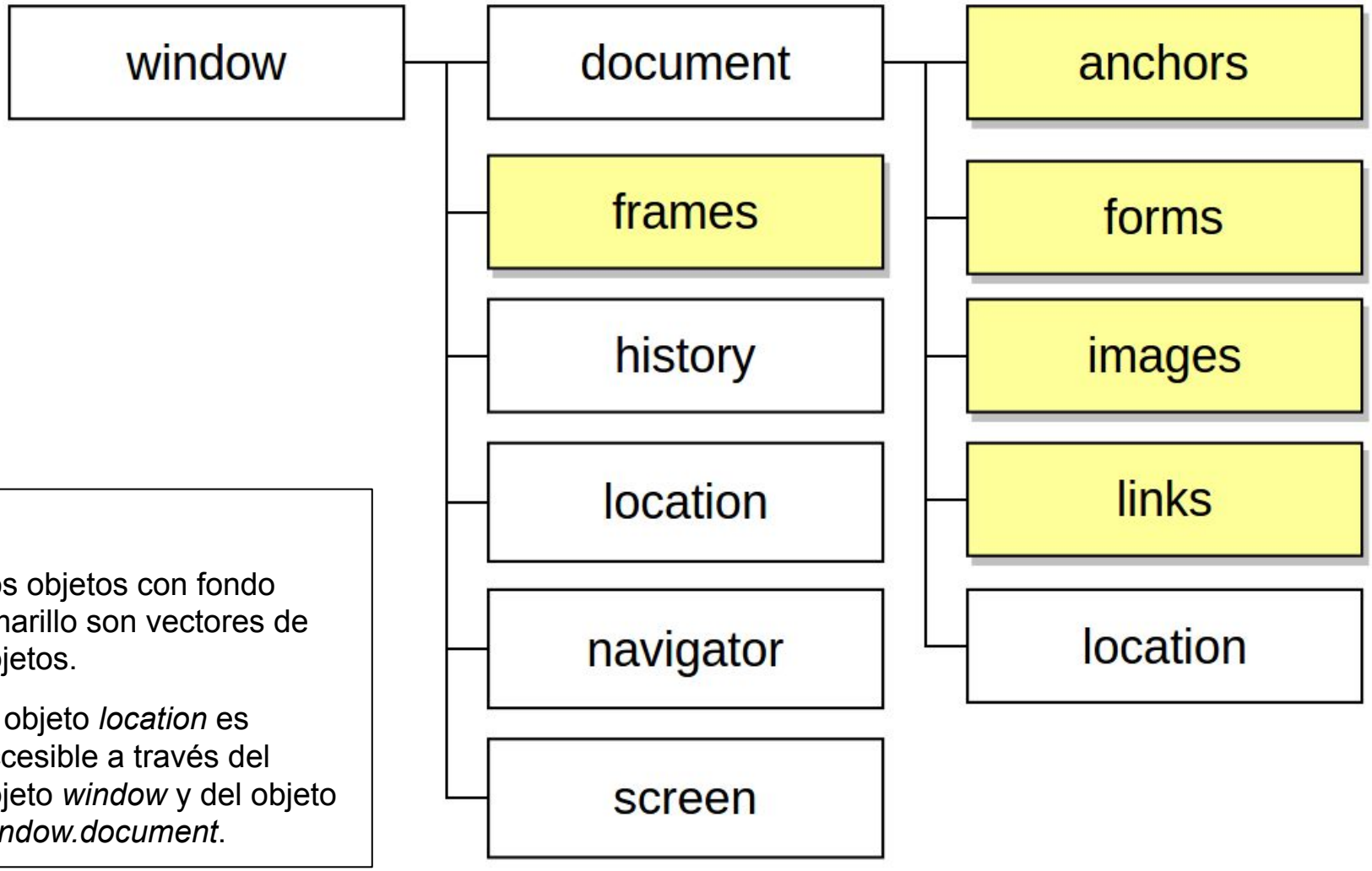
Browser Object Model (BOM)

BOM (Browser Object Model)

- Concepto introducido por las versiones 3.0 de los navegadores Internet Explorer y Netscape Navigator.
- Permite acceder y modificar las propiedades de las ventanas del propio navegador.
- No existe un estándar, por lo que cada navegador puede tener sus propios objetos.

BOM (Browser Object Model)

- Está compuesto por varios objetos relacionados entre sí:



Notas:

- Los objetos con fondo amarillo son vectores de objetos.
- El objeto *location* es accesible a través del objeto *window* y del objeto *window.document*.

BOM (Browser Object Model)

- Objeto **window**

Representa la ventana del navegador, permitiendo moverla, redimensionarla, cerrarla y abrir nuevas ventanas.

Objetos del objeto window:

- **document**. Devuelve el objeto *Document* (DOM) de la ventana.
- **history**. Devuelve el historial de la ventana.
  - Propiedades:
    - length. Devuelve el número de URLs almacenadas en el historial.
  - Métodos:
    - back(). Carga la URL previa en la lista del historial.
    - forward(). Carga la URL siguiente en la lista del historial.
    - go(*número*). A la posición de la URL actual en el historial le suma el *número* de posiciones indicado (positivo o negativo) y carga la URL correspondiente.

BOM (Browser Object Model)

- Objeto **window**

Objetos del objeto window:

- **location**. Devuelve el objeto *location* que proporciona información sobre la URL actual mediante las siguientes propiedades:

propiedades:
  - hash. Devuelve o establece la parte del **#** de la URL (*anchor part*).
  - search. Devuelve o establece la parte del **?** de la URL (*querystring part*).
  - host. Devuelve el nombre del *host* y el *puerto* de la URL.
  - hostname. Devuelve el nombre del *host* de la URL.
  - href. Devuelve o establece la URL entera de la ventana.
  - origin. Devuelve el protocolo, nombre del *host* y *puerto* de la URL
  - pathname. Devuelve o establece lo que va a continuación del *host* de una URL (*path name*).

métodos:
  - assign(*URL*). Carga el nuevo documento indicado por la URL.
  - reload(). Recarga el documento.
  - replace(*URL*). Carga el nuevo documento indicado por la URL. A diferencia de *assign()*, *replace()* elimina la URL del documento actual del objeto *history*.

BOM (Browser Object Model)

- Objeto **window**

Objetos del objeto window:

- **navigator**. Devuelve el objeto *navigator* que proporciona información sobre el navegador.
  - Propiedades:
    - appCodeName. Devuelve la cadena "Mozilla".
    - appName. Devuelve la cadena "Netscape" o el nombre completo del navegador.
    - appVersion. Devuelve la cadena "4.0" o una cadena que representa la versión del navegador en detalle.
    - platform. Devuelve una cadena vacía o una cadena representando la plataforma en la que está corriendo el navegador: "Linux x86_64", "MacIntel", "Win32", "FreeBSD i386", "WebTV OS", …
    - userAgent. Devuelve el valor de la cabecera *User-Agent* en la petición HTTP, o cadena vacía si no se envía dicha cabecera.

BOM (Browser Object Model)

- Objeto **window**

Objetos del objeto window:

- **screen**. Devuelve el objeto *screen* que permite acceder a propiedades de la pantalla (monitor).
  - Propiedades:
    - height/width. Devuelve el alto/ancho de la pantalla en píxeles.
    - availHeight/availWidth. Devuelve en píxeles el alto/ancho disponible de la pantalla, excluyendo las barras de tareas del sistema operativo.
    - colorDepth. Devuelve la profundidad de color de la pantalla. Su valor es siempre 24 por razones de compatibilidad.
    - pixelDepth. Devuelve la profundidad de bit de la pantalla. Su valor es siempre 24 por razones de compatibilidad.

BOM (Browser Object Model)

- Objeto **window**

Algunas propiedades del objeto window:

- **screenX**, **screenY**. Devuelven la posición de la esquina superior de la ventana con respecto a la esquina superior derecha de la pantalla.
- **pageXOffset**/**pageYOffset**. Devuelve la cantidad en píxeles del scroll horizontal/vertical aplicado al documento.
- **innerWidth**/**innerHeight**. Devuelve o establece el ancho/alto del área de contenido de la ventana.
- **outerWidth**/**outerHeight**. Devuelve o establece el ancho/alto exterior de la ventana, incluyendo las barras de herramientas y de scroll.
- **scrollX**/**scrollY**. Devuelve el mismo valor que pageX/YOffset.
- **devicePixelRatio**. Devuelve el resultado de dividir el tamaño de un píxel CSS por el tamaño de píxel del monitor (o dispositivo de salida).

BOM (Browser Object Model)

- Objeto **window**

Algunos métodos del objeto window:

- **open(***URL, target, parámetros***)**. Abre una nueva ventana del navegador. Todos los parámetros son opcionales.
  - *URL*. Especifica la *URL* a abrir en la nueva ventana.
  - *target*. Especifica el valor del atributo target o el nombre del objeto window que se crea (no es el título de la ventana). Valores posibles: _blank, _parent, _self.
  - *parámetros*: Permite especificar parámetros como *height* (valor mínimo 100), *width* (valor mínimo 100), top, left.
- **close()**. Cierra la ventana.
- **focus()**. Pone el foco en la ventana, la pone como activa.
- **blur()**. Elimina el foco de la ventana.
- **stop()**. Detiene la carga del documento.

BOM (Browser Object Model)

- Objeto **window**

Algunos métodos del objeto window:

- **alert(** *mensaje* **)**. Muestra una ventana de alerta con un mensaje y un botón de aceptar.
- **confirm(** *mensaje* **)**. Muestra una ventana de diálogo con un mensaje y botones de aceptar y cancelar.
- **prompt(** *mensaje*, *respuesta_por_defecto* **)**. Muestra una ventana que pide información al usuario.
- **btoa(** *texto* **)**. Codifica *texto* en base-64.
- **atob(** *textoCodificadoBase64* **)**. Decodifica *textoCodificadoBase64*.
- **scroll(** *x*, *y* **), scrollTo(** *x*, *y* **)**. Sitúa la posición de *scroll* del documento en el valor horizontal y vertical indicado por *x* e *y*.
- **scrollBy(** *x*, *y* **)**. Suma a la posición de *scroll* actual del documento el valor horizontal y vertical indicado por *x* e *y*.

BOM (Browser Object Model)

- Objeto **window**

Algunos métodos del objeto window:

- **moveBy**(*x*, *y*). Mueve la ventana ***x*** píxeles a la derecha e ***y*** píxeles a la izquierda.
- **moveTo**(*x*, *y*). Mueve la ventana a la posición (*x*, *y*).
- **resizeBy**(*x*, *y*). Redimensiona la ventana sumando ***x*** píxeles al ancho e ***y*** píxeles al alto.
- **resizeTo**(*x*, *y*). Redimensiona la ventana a ***x*** píxeles de ancho e ***y*** píxeles de alto.

Nota: Por temas de seguridad, estos cuatro métodos sólo funcionarán con nuevas ventanas abiertas en modo *pop-up*.

BOM (Browser Object Model)

- Objeto **window**

Algunos manejadores de eventos del objeto window, comunes a interfaces como `Window`, `HTMLElement` o `Document`:

- `onblur`. Se dispara tras perder la ventana el foco.
- `onfocus`. Se dispara cuando la ventana recibe el foco.
- `onclick`. Se dispara al pulsar y soltar cualquier botón del ratón.
- `ondblclick`. Se dispara al hacer doble click con cualquier botón del ratón.
- `onclose`. Se dispara tras cerrar la ventana (objeto *window*).
- `onload`. Se dispara tras cargar completamente todos los recursos y el DOM. No se dispara si la carga viene de la caché.
- `onkeypress`. Se dispara al pulsar una tecla (excepto Shift, Fn y CapsLock).
- `onkeyup`. Se dispara al soltar cualquier tecla pulsada.

BOM (Browser Object Model)

- Objeto **window**

Algunos manejadores de eventos del objeto window, comunes a interfaces como `Window`, `HTMLElement` o `Document`:

- `onmousedown`. Se dispara cuando se pulsa cualquier botón del ratón.
- `onmouseup`. Se dispara cuando se suelta cualquier botón del ratón que estuviera pulsado.
- `onmousemove`. Se dispara cuando el ratón se mueve dentro de la ventana.
- `onmouseout`. Se dispara cuando el ratón deja la ventana.
- `onmouseenter`. Se dispara cuando el ratón entra en la ventana procedente del exterior.
- `onmouseover`. Se dispara cuando el ratón entra en la ventana, ya sea procedente del exterior o de otro elemento interior.

BOM (Browser Object Model)

- Objeto **window**

Algunos manejadores de eventos del objeto window, comunes a interfaces como `Window`, `HTMLElement` o `Document`:

- `oncontextmenu`. Se dispara cuando se pulsa el botón derecho del ratón.
- `onreset`. Se dispara cuando se hace *reset* en un formulario.
- `onsubmit`. Se dispara cuando se hace *submit* en un formulario.
- `onresize`. Se dispara cuando se cambia el tamaño de la ventana.
- `onscroll`. Se dispara cuando se hace *scroll* en la ventana.
- `onwheel`. Se dispara cuando se mueve la rueda del ratón.
- `onstorage`. Se dispara cuando se produce un cambio en sessionStorage o localStorage.

La lista completa de manejadores de eventos se puede encontrar en:
https://developer.mozilla.org/es/docs/Web/API/GlobalEventHandlers