JavaScript Objetos predefinidos

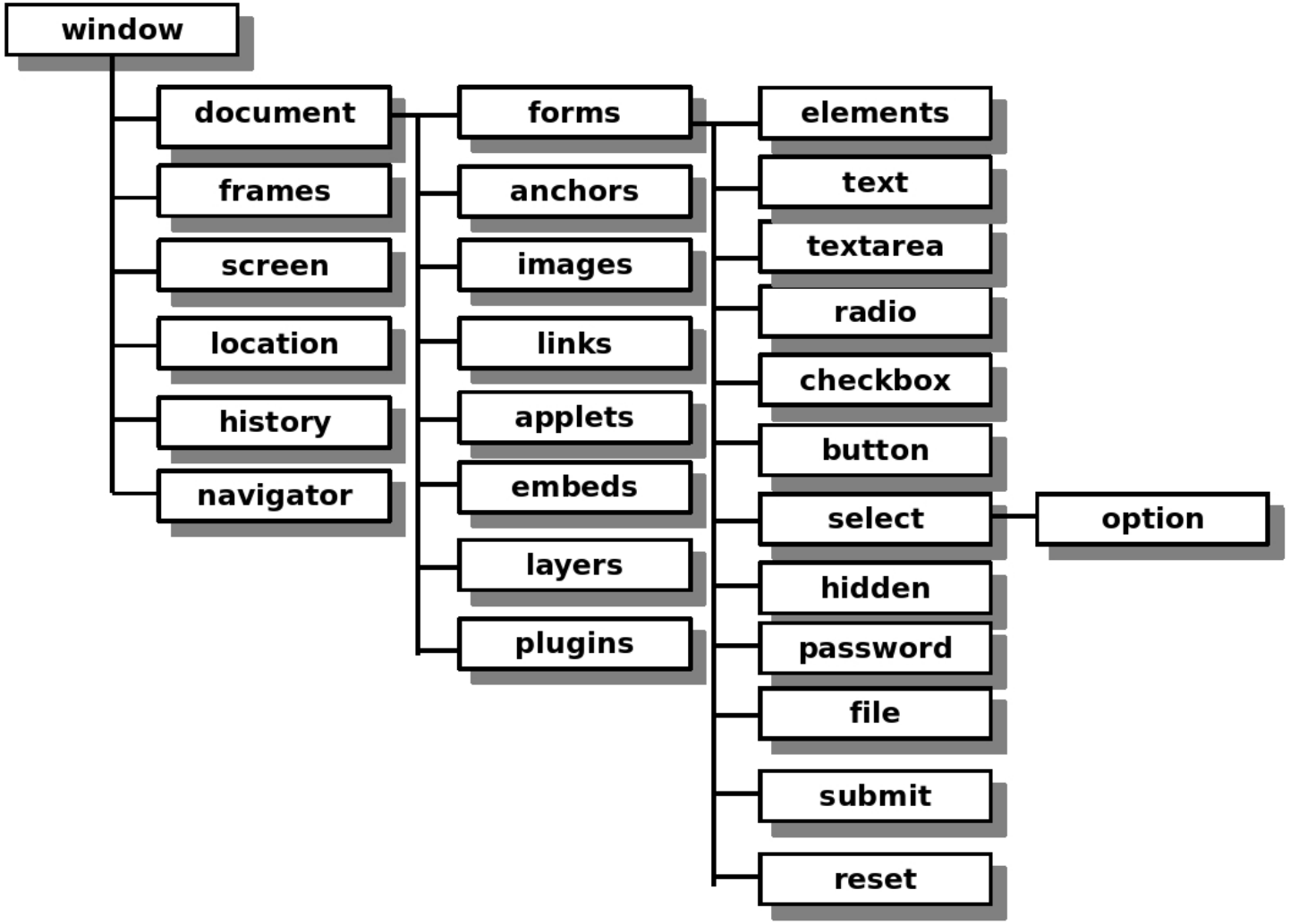
Víctor Custodio

- Cada vez que se carga una página en el navegador, el intérprete de JavaScript crea automáticamente una serie de objetos que pueden ser usados al programar en JavaScript. De entre los distintos objetos generados automáticamente destacan los siguientes:
  - **Window**: es el objeto principal del cual "cuelgan" los demás como si fueran propiedades suyas. Se crea un objeto de este tipo para cada ventana que pueda ser lanzada desde una página Web.
  - **Navigator**: contiene información acerca del navegador, tal como nombre, versión, tipos MIME soportados por el cliente y *plugs-in* instalados.
  - **Location**: con información acerca del URL que estemos visualizando.

- **History**: historial en el que se guardan los URL ya visitados.
- **Document**: es el contenedor de todos los objetos que se hayan insertado en la página web: enlaces, formularios, imágenes o marcos.
- **Frames**: Matriz conteniendo los distintos objetos *frame* que puede contener una ventana. Cada *frame* es a su vez otra ventana.

El objeto Window

- Es el más importante, ya que bajo el cuelgan el resto de elementos



El objeto Window

- 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 frames, puede sacar mensajes (de error u otro tipo), confirmación y entrada de datos, y como se ha dicho, mantiene un array por cada tipo de elemento que puede haber en un documento, formularios, enlaces, imágenes y marcos.

Window: apertura de una ventana

var nuevaVentana = window.open("dirección URL");

En realidad, son varios los parámetros que se pasan a una ventana al crearla, pero solamente la URL es obligatoria:

variable=window.open("dirección URL", "nombre de la Ventana","parámetros de apertura");

- **dirección URL** es la página que se va a cargar en la nueva ventana.
- **nombre de la ventana** es el nombre que se podrá utilizar posteriormente en los *target* de los enlaces.
- **parámetros de apertura** es una cadena que contiene los valores para ciertos atributos de la ventana, que son los siguientes:

  toolbar,location,directories,status,menubar, scrollbars,resizable,width, height.

  - Todos menos los de tamaño, pueden tomar valores de yes o no, o 1 y 0.
- Ejemplo completo:

nuevaVentana=window.open("http://www.as.com", "segundaPag","toolbar=no,location=yes,resizable=yes,height=400");

Cerrar Ventanas

- Para cerrar una ventana se utiliza el método **close()** de las mismas:

  variable_de_ventana.close()

- Si la ventana a cerrar es la propia, entonces tendremos que usar:

window.close() , o simplemente , **close();**

Window.confirm("texto")

- Es una caja de texto, como el alert, pero solicita la respuesta del usuario, que en este caso puede ser positiva o negativa.
- Si la respuesta es positiva nos devuelve true, si es negativa, false.
- Ejemplo

```
function  nueva() {
        if(confirm("¿Seguro que quieres cerrar la  ventana?")){
        close();
        }else{
        alert("Gracias por no despedirte de nosotros aún");
        }
}
```

Ejemplo: Abrir y cerrar ventanas en JS

```
<!DOCTYPE html>
<HTML>
<HEAD>
<TITLE>Abriendo y cerrando ventanas con Javascript</TITLE>
<SCRIPT>
<!--
var nuevaVentana;
function  nueva() {
  nuevaVentana=nuevaVentana=window.open("", "segundaPag",   "width=300,height=300" );
  nuevaVentana.document.write("<HTML><HEAD><TITLE>"+
  "Sin Título</TITLE></HEAD>\n");
  nuevaVentana.document.write("<BODY><form>\n");
  nuevaVentana.document.write("<input type='button' "+
  "value='Cerrar' onClick='window.close();'>\n");
  nuevaVentana.document.write("</form>\n");
  nuevaVentana.document.write("</BODY></HTML>\n");
  nuevaVentana.document.close();
}
</SCRIPT>
<BODY>
<FORM>
  <INPUT TYPE="button" VALUE="Abrir" onClick="nueva();"><br>
</FORM>

</BODY>

</HTML>
```

Mover una ventana

- Podemos mover una ventana con los metodos moveTo() y moveBy(),
- El moveTo(x,y) mueve la ventana a unas coordinadas fijas
- El moveBy(x,y) mueve la ventana de forma relativa a donde se encuentre

Ejemplo mover ventana

```
<!DOCTYPE html>
<html>
<body>
<p>Abre una nueva ventana y la mueve de forma relativa hacia abajo a la derecha</p>
<button onclick="openWin()">Open "myWindow"</button>
<button onclick="moveWin()">Move "myWindow"</button>
<script>
var myWindow;
function openWin() {
   myWindow = window.open("", "myWindow", "width=200, height=100");
   myWindow.document.write("<p>This is 'myWindow'</p>");
}
function moveWin() {
   myWindow.moveBy(250, 250); //probar tambien moveTo(250,250)
   myWindow.focus();
}
</script>

</body>
</html>
```

Window: cambiar de tamaño la ventana

- Al igual que para mover tenemos moveBy y moveTo, para cambiar el tamaño tenemos los metodos resizeTo(ancho,alto) y resizeBy(ancho,alto):

```
<!DOCTYPE html>
<html>
<body>
<p>Abre una nueva ventana y aumenta su tamaño de forma relativa 250 pixeles de ancho y alto cada vez que se pulsa el boton</p>
<button onclick="openWin()">Open "myWindow"</button>
<button onclick="moveWin()">Move "myWindow"</button>
<script>
var myWindow;
function openWin() {
   myWindow = window.open("", "myWindow", "width=200, height=100");
   myWindow.document.write("<p>This is 'myWindow'</p>");
}
function resizeWin() {
   myWindow.resizeBy(250, 250); //probar tambien con resizeTo(250,250)
   myWindow.focus();
}}
</script>

</body>
</html>
```

Temporizadores

- Existen dos formas de utilizar temporizadores en javaScript, ambas son métodos del objeto window.
  - ***setTimeOut(funcionJS, tiempo)***
  - ***setInterval(funcionJS, tiempo)***
- El primer método, ejecutará las instrucciones que le pasemos en el primer parámetro, una vez haya pasado el tiempo que le indicamos en el segundo parámetro.
- El segundo, realizará lo mismo que el primero, sólo que periódicamente, cada vez que pase el tiempo, ejecutará las instrucciones

Temporizadores

- Para borrar los temporizadores tenemos sus metodos archienemigos:
  - clearTimeOut(temporizadorTimeOut)
  - clearInterval(temporizadorInterval);

  - Ambas pararán la ejecución de las instrucciones si son llamadas durante su tiempo de espera.

Ejemplo temporizadores

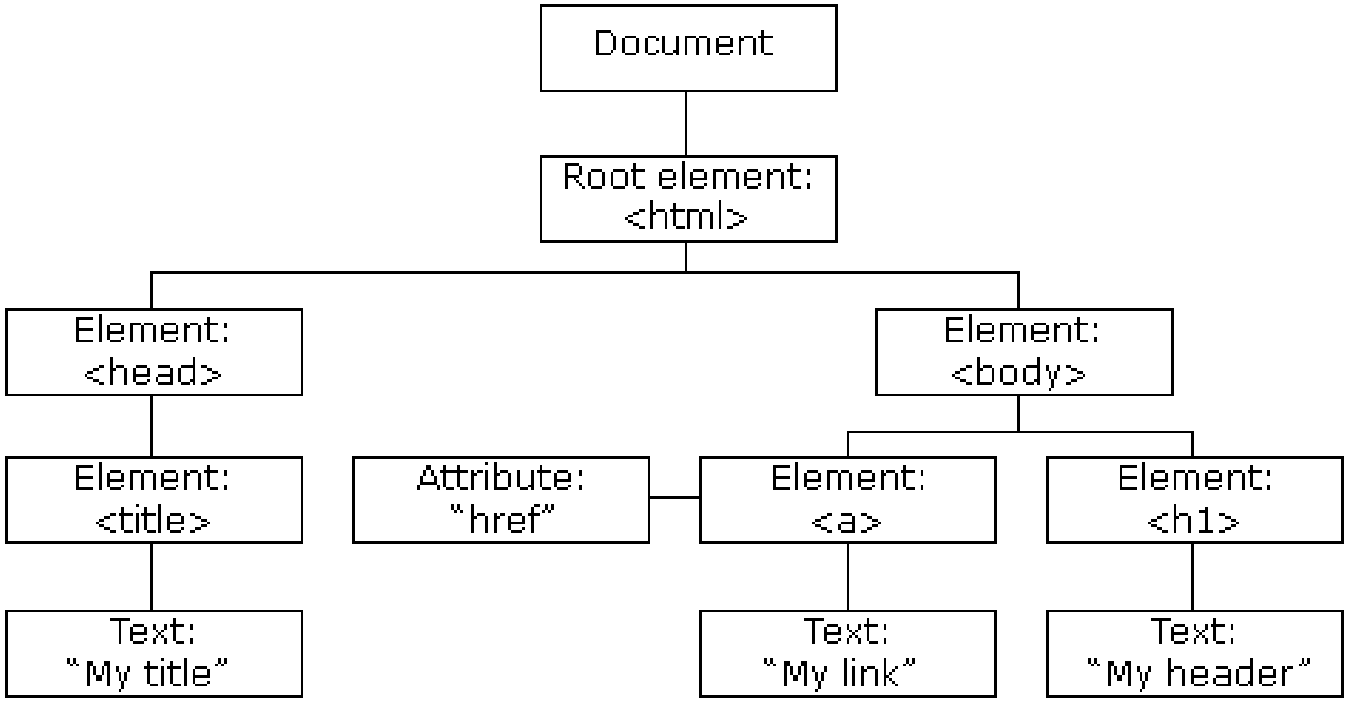
```
<!DOCTYPE html>
<html>
<body>

<p>Con ayuda del temporizador Interval, ejecutamos un script que cambia la hora cada segundo:</p>
<p> tambien tenemos un boton que paraliza el script</p>

<p id="demo"></p>

<button onclick="congeladorDeTiempo()">Parar el tiempo</button>

<script>
var myVar = setInterval(function(){ miReloj() }, 1000);

function miReloj() {
   var d = new Date();
   var t = d.toLocaleTimeString();
   document.getElementById("demo").innerHTML = t;
}

function congeladorDeTiempo() {
   clearInterval(myVar);
}
</script>

</body>
</html>
```

Window

- Existen más propiedades y métodos de Windows,
- Aquí tienes una lista de éstos explicada:
- http://www.webestilo.com/javascript/js17.phtml

DOM

- Cuando se carga una página web, el navegador crea un **D**OCUMENT **O**bject **M**odel de la página.
- El **HTML DOM** modelo está construido como un árbol de **objetos** :



DOM

- Con el modelo de objetos, JavaScript recibe toda la información que necesita para crear HTML dinámico:
  - JavaScript puede cambiar todos los elementos HTML en la página
  - JavaScript puede cambiar todos los atributos HTML en la página
  - JavaScript puede cambiar todos los estilos CSS en la página
  - JavaScript puede eliminar los elementos HTML y atributos existentes
  - JavaScript puede agregar nuevos elementos HTML y atributos
  - JavaScript puede reaccionar a todos los eventos de HTML existentes en la página
  - JavaScript puede crear nuevos eventos HTML en la página

Objeto document

- El objeto **document**, es el objeto que representa el árbol del DOM que vimos en la diapositiva 4.
- Cómo la mayoría de objetos en JavaScript, este objeto tiene métodos y propiedades.

DOM: Métodos y propiedades

- Métodos DOM HTML son **las acciones** que se pueden realizar (en Elementos HTML)
- Propiedades HTML DOM son **los valores** (de los elementos HTML), estas se pueden acceder, establecer o cambiar.
- Ejemplo de método:
  - getElementById(id)
- Ejemplo de propiedad:
  - innerHTML.
- Ejemplo de uso:

```
<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>
```

Objeto Document: Propiedades

- **alinkColor**. Esta propiedad tiene almacenado el color de los enlaces activos
- **anchors**. Se trata de un array con los enlaces internos existentes en el documento
- **bgColor**. Propiedad que almacena el color de fondo del documento
- **cookie**. Es una cadena con los valores de las cookies del documento actual
- **domain**. Guarda el nombre del servidor que ha servido el documento
- **embeds**. Es un array con todos los EMBED del documento
- **fgColor**. En esta propiedad tenemos el color del primer plano

Objeto Document: Propiedades

- **forms**. Se trata de un array con todos los formularios del documento. Los formularios tienen a su vez elementos (cajas de texto, botones, etc) que tienen sus propias propiedades y métodos
- **images**. Array con todas las imágenes del documento
- **lastModified**. Es una cadena con la fecha de la última modificación del documento
- **linkColor**. Propiedad que almacena el color de los enlaces
- **links**. Es un array con los enlaces externos
- **location**. Cadena con la URL del documento actual
- **referrer**. Cadena con la URL del documento que llamó al actual, en caso de usar un enlace.
- **title**. Cadena con el título del documento actual
- **vlinkColor**. Propiedad en la que se guarda el color de los enlaces visitados

Objeto Document: Métodos

- Métodos para encontrar elementos en el DOM

document.getElementById(“unaID”)	Encuentra un elemento por su Id
document.getElementsByTagName(“unaEtiqueta”)	Encuentra todos los elementos con el nombre de la etiqueta pasada por parametros
document.getElementsByClassName(“unClass”)	Encuentra todos los elementos con la clase pasada por parametros

Objeto Document: Métodos

- Métodos para manejar eventos

document.getElementById(*id*).onclick=function(){*codigo*}	Añade una función al evento onclick del element con el id id.

Objeto Document: Métodos para modificar el HTML

element.innerHTML=	Cambia el contenido de la etiqueta(lo que hay entre la etiqueta de apertura y de cierre)
element.nombreAtributo=	Cambia el valor del atributo del elemento
element.setAttribute*(attribute,value)*	Cambia el valor del atributo del elemento
element.style.*nombrePropiedad=*	Cambia el estilo de un elemento HTML.

Propiedades CSS en JavaScript

Objeto Document: Métodos para crear y eliminar elementos

Metodo	Descripción
document.createElement(nombreetiqueta)	Crea un elemento HTML
document.removeChild(elemento)	Elimina un elemento html hijo
document.appendChild(elemento)	Añade un elemento html al documento
document.write(*text*)	Escribe directamente sobre el documento

Estos métodos pueden ejecutarse sobre cualquier elemento que pueda tener hijos, no solamente sobre document. Por ejemplo:

```
var para = document.createElement("p");
para.innerHTML="Esto es nuevo";
var element = document.getElementById("div1");
element.appendChild(para);
```

DOM

- Realiza los ejercicios DOM1,2,3,4 y 5
- Con lo que hemos visto, ya podemos hacer animaciones con JS(ejemplo)
- Existen más formas de trabajar con el DOM de javascript que nos dan casi infinitas posibilidades. Muchas de estas formas no son muy intuitivas de utilizar (aunque van mejorando con las nuevas versiones de JS) y requieren demasiado código para realizar pocas cosas.
- Para ello existen librerías que nos ofrecen otra forma de trabajar con el DOM y otros elementos de JS, entre ellas JQUERY , pero eso ya queda para cursos más específicos de JS.