MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO DE WEB ENTORNO CLIENTE
TÉCNICO EN DESARROLLO DE APLICACIONES WEB

Los objetos de lenguaje

05

ÍNDICE

OBJETIVOS

Conocer los objetos definidos en el lenguaje

Saber utilizar marcos

Conocer los métodos de acceso a la ventana del navegador

Conocer los métodos de acceso a la pantalla del usuario

Conocer los mecanismos de apertura de ventana con parámetros

Saber modificar el contenido de un documento web dinámicamente

Conocer el concepto de cookie

/ 1. Introducción y contextualización práctica

Para poder crear documentos web de forma dinámica necesitamos conocer los métodos JavaScript que nos permiten realizarlo. Desde HTML es necesario realizar ciertas modificaciones en el código para permitir el acceso desde JavaScript. En el código HTML podemos distinguir entre etiquetas y atributos, y estas últimas son fundamentales para el acceso dinámico.

JavaScript es un lenguaje que nos ofrece muchas funcionalidades de manera nativa, por lo que es importante conocer algunas de las funciones más relevante para que nuestro código sea más fácil de desarrollar.

Como veremos a lo largo del tema, existen aspectos de HTML y JavaScript que son estándar para todos los navegadores y no implican ningún problema. Sin embargo, hay otros que no lo son, y son precisamente estos los que dificultan el desarrollo web.

Escucha el siguiente audio que describe el caso práctico que iremos resolviendo a lo largo de la unidad:

Audio Intro. JavaScript: frames adaptativos
https://bit.ly/30q0gzi

/ 2. Frames o marcos

Un frame o marco es un elemento HTML que permite contener y dividir la página en N partes. Cada una de ellas es independiente y puede tener diferentes contenidos. Realmente no existe límite para lo que pueden albergar cada una de las partes, aunque pueden tener propiedades semejantes, como, por ejemplo, el tamaño o resolución de la pantalla.

En ocasiones se confunde un marco con una tabla, puesto que pueden tener una apariencia similar. La principal diferencia entre estos es que, en la tabla, el tamaño de celda es fijo. Además, un marco posee todos los elementos propios del HTML.

Cada una de las zonas en la que se divide la pantalla se define con la etiqueta <frameset>. Cada frameset puede contener un conjunto de marcos que se definen con la etiqueta <frame>. Las páginas que contienen el frameset no pueden tener cuerpo y, en caso de tenerlo, este no se visualizará correctamente. Como la página principal está dividida en diferentes marcos, cada uno de ellos posee una página independiente. Es importante tener en cuenta que no todos los navegadores soportan los marcos. Por este motivo, se incluye un comportamiento alternativo a los marcos mediante el uso de la etiqueta <noframes>. Esta etiqueta garantiza que la página se muestre, aunque con otro aspecto, cuando el navegador del usuario no soporte los marcos. Por ello es importante que también cuidemos esta visualización, pues no es de extrañar que el navegador prohíba los marcos.

Fig. 1. Un documento web puede dividirse en diferentes marcos

Nótese que a partir de HTML5 los marcos no están soportados, pues existen otras técnicas para lograr que la página sea adaptativa, es decir, que ajuste la visualización del contenido al tipo dispositivo que se está utilizando.

2.1. Ejemplo de marcos

A continuación, se muestra un ejemplo en el que se utilizan marcos para una página:

```
<HTML>
<HEAD>
<TITLE>Un documento simple con marcos</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
 <FRAMESET rows="100, 200">
   <FRAME src="frame1.html">
   <FRAME src="frame2.html">
 </FRAMESET>
 <FRAME src="frame3.html">
 <NOFRAMES>
   <P>Este conjunto de marcos contiene:
   <UL>
     <LI><A href="frame1.html">Frame 1</A>
     <LI><A href="frame2.html">Frame 2</A>
     <LI><A href="frame3.html">Frame 3</A>
   </UL>
 </NOFRAMES>
</FRAMESET>
</HTML>
```

Código 1. Marcos para una página

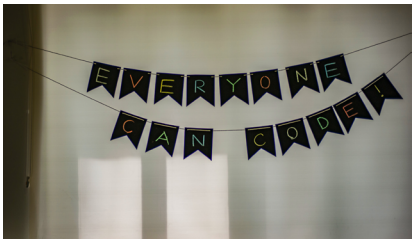

Fig. 2. Cuando los frames no están soportados hay que ofrecer alternativas

Como se puede observar, en el ejemplo se crean tres marcos, cada uno de los cuales tiene su correspondiente documento HTML: frame1.html, frame2.html y frame3.html.

Además, se incluye un comportamiento especial en el caso de que el navegador no pueda visualizar los marcos. En esta solución se ha optado por mostrar una lista desordenada que tiene enlaces a las diferentes páginas.

Audio 1. "Robo de cookies"
https://bit.ly/3dOEpVV

/ 3. Introducción al DOM (Document Object Model)

Una página HTML, cuando se analiza por el navegador, se puede visualizar como un árbol de objetos, donde cada uno de ellos es un nodo del árbol. Esta representación arbórea recibe el nombre de DOM (Documento Object Model).

Gracias a esta representación, y utilizando JavaScript, podemos modificar parte de la página web a través de código JavaScript. Algunas de las funciones que podemos realizar accediendo al DOM son las siguientes:

- Cambiar todos los elementos HTML que se encuentran en la página.
- Cambiar todos los atributos de aquellos elementos que deseemos que estén en la página que se ha cargado.
- Modificar la representación visual, es decir, los estilos de los elementos que están en la página.
- Eliminar elementos HTML que se encuentre en la página.
- Añadir nuevos elementos a dicha página.
- Reaccionar a eventos existentes en la página.
- Crear nuevos eventos en la página.

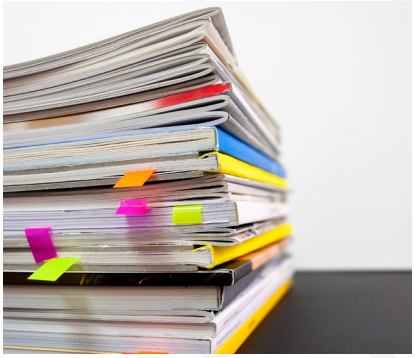

Fig. 3. JavaScript proporciona el modelo de documento para HTML

Como se puede observar, gracias a JavaScript podemos tener acceso a cada una de las partes del documento que hemos cargado. Este acceso nos permite leer o modificar prácticamente cualquier elemento que está en la página cargada.

Video 1. "DOM HTML"
https://bit.ly/2ziBu97

/ 4. Caso práctico 1: “Frames”

Planteamiento. Nuestro jefe nos pide que revisemos qué está pasando en el siguiente código HTML cuando se visualiza en un navegador web:

```
<html>
<head>
</head>
<frameset cols=”20%, 80%”>
 <frameset rows=”100, 200”>
   <frame src=”frame1.html”>
 </frameset>
 <frame src=”frame3.html”>
 <noframes>
   <ul>
     <li><a href=”frame1.html”>Frame 1</a>
   </ul>
 </noframes>
</frameset>
<body>Example</body>
</html>
```

Código 2. Caso práctico visualización de frames

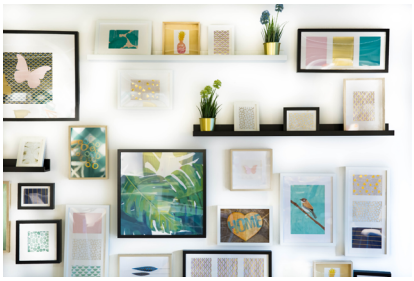

Fig. 4. El abuso de marcos puede ser caótico en el diseño web

Nudo. ¿Ves algo extraño? ¿Se visualizan todos los frames?

Desenlace. Como hemos visto en el apartado sobre los frames, para poder disponer de todos los frames es necesario utilizar un frame para cada web que queramos visualizar. Además, en las páginas que utilicen frames no podemos utilizar el body, pues el cuerpo debe ir en las páginas que definen el contenido. En el caso de no seguir estas pautas, los frames no se visualizarán correctamente como sucede en el código planteado.

/ 5. DOM como estándar

Todos los cambios que realicemos sobre el documento utilizando el DOM se visualizarán o surtirán efecto en el momento, de modo que no es necesario volver a cargar la página. El DOM es un estándar web que define una forma uniforme de acceder a los documentos.

Se encuentra dividido en tres partes:

- Núcleo: es estándar para todos los tipos de documentos.
- XML: es estándar únicamente para documentos XML.
- HTML: estándar únicamente para HTML.

5.1. Generación de código HTML

Como hemos visto anteriormente, el DOM permite insertar o crear nuevo código HTML. En este apartado vamos a aprender cómo generarlo. Para poder crear código HTML es necesario conocer dónde vamos a generarlo, es decir, hay que identificar el elemento sobre el que lo vamos a añadir. Supongamos que tenemos el siguiente fragmento de código HTML

```
<div id="div1">
  <p id="p1">Párrafo 1</p>
  <p id="p2">Párrafo 2</p>
</div>
```
Código 3. Contenedor de 2 párrafos

Fig. 5. JavaScript utiliza los identificadores HTML para seleccionar tags

Como se puede observar en el ejemplo anterior, se muestra un contenedor que alberga dos párrafos. HTML define un conjunto de etiquetas y cada una de ellas puede tener unos atributos asociados. Centrándonos en el ejemplo, todas las etiquetas tienen definido el atributo identificador (id), que posee la peculiaridad de ser el único dentro de un documento HTML. Así, el contenedor tiene identificador div1 y los párrafos p1 y p2 respectivamente. Desde JavaScript, utilizando los identificadores de HTML y el DOM, podemos manejar todo el documento web.

5.2. Ejemplo de generación de código HTML

Supongamos que queremos crear un nuevo párrafo dentro del contenedor div1 del ejemplo anterior (Código 3). El código JavaScript resultante sería:

```
var para = document.createElement("p");
var node = document.createTextNode("Párrafo 3");
para.appendChild(node);
var element = document.getElementById("div1");
element.appendChild(para);
```
Código 4. Generación código HTML

En este código vemos que para generar un nuevo párrafo se utiliza una variable llamada document. Esta variable es proporcionada directamente por el DOM, de modo que no es necesaria su declaración. La forma de manejar el DOM va a ser siempre similar, es decir, accediendo a la variable document.

La sintaxis que se utiliza para manejar el DOM requiere de la utilización de ciertas funcionalidades, en este caso createElement, createTextNode y getElementById. Cada una de estas funcionalidades se denominan funciones y están documentadas en JavaScript. Siguiendo el desarrollo del ejemplo, vemos cómo se crea primero un elemento párrafo; a continuación, se añade un nodo de texto al párrafo y, por último, al contenedor se le añade el párrafo.

5.3. MODIFICACIÓN Y BORRADO DEL HTML

Como hemos visto anteriormente, para modificar el documento HTML utilizamos el DOM de JavaScript y las posibilidades que este nos ofrece. Para la modificación y borrado de elementos del documento HTML sería necesario encontrar la funcionalidad de JavaScript correspondiente. Lo representamos en el siguiente ejemplo:

```
<script>
var para = document.createElement("p");
var node = document.createTextNode("Nuevo párrafo 1");
para.appendChild(node);
var parent = document.getElementById("div1");
var p1 = document.getElementById("p1");
var p2 = document.getElementById("p2");
parent.replaceChild(para, p1); // reemplazo p1
parent.removeChild(p2); //borrado del párrafo 2
</script>
```
Código 5. Modificación y borrado del HTML

/ 6. Utilización de objetos

La orientación a objetos en JavaScript se ha implementado de una manera sencilla. Un objeto se define como una colección de propiedades que tiene una asociación entre una clave y un valor. Una propiedad puede ser una funcionalidad que el objeto puede desarrollar, o bien, un estado que se traduce como un valor. El lenguaje puede ofrecer objetos de manera nativa y también permitir que el usuario cree sus propios objetos.

6.1. Funciones nativas

JavaScript proporciona algunas funciones nativas que permiten llevar acabo determinadas tareas:

Función	Descripción
eval()	Permite evaluar el código JavaScript representado como una cadena
parseFloat()	Permite transformar el contenido a un número flotante
parseInt()	Permite transformar el contenido a un número entero
isNaN()	Permite comprobar si la variable es NaN o no.

Tabla 1. Funciones nativas de JavaScript

6.2. Objetos nativos

En JavaScript casi todo es un objeto exceptuando los tipos primitivos. Algunos objetos nativos son:

Función	Descripción
Array	Permite definir una conexión de elementos ordenados
Map	Permite definir un conjunto de elementos con clave valor
JSON	Permite definir datos estructurados

Tabla 2. Objetos nativos de JavaScript

/ 7. Interacción con el navegador

JavaScript permite manejar el documento web y también la ventana del navegador utilizando el BOM (Browser Object Model). A diferencia de lo que sucede con el DOM, a la hora de manejar la ventana del navegador no existe ningún estándar. Sin embargo, la mayor parte de los navegadores actuales implementan casi la misma funcionalidad.

7.1. Ventana

La ventana del navegador se puede manejar a través del objeto window que ofrece JavaScript. Las variables globales, objetos y funciones provienen del objeto window. Incluso el documento web también proviene de él. A través de este objeto se puede controlar diferentes aspectos como, por ejemplo:

- Ajustar las dimensiones de la ventana del navegador.
- Abrir una ventana del navegador.
- Cerrar una ventana del navegador.
- Cambiar el tamaño de la ventana del navegador.

```
<script>
var w = window.innerWidth
|| document.documentElement.clientWidth
|| document.body.clientWidth;

var h = window.innerHeight
|| document.documentElement.clientHeight
|| document.body.clientHeight;
</script>
```

Código 6. Ancho y alto ventana del navegador

En el ejemplo anterior se muestra cómo se obtienen el ancho y el alto de la ventana del navegador. Como el BOM no es estándar, hay que adaptarse a los diferentes navegadores. En este código se utiliza el operador lógico 'O' para obtener las dimensiones a través de diferentes métodos. Desde que un valor sea válido, no se ejecutará el resto. Para abrir una nueva ventana del navegador:

```
var valor = 'valorParaLaVentana';
window.open('/childwindow.html?value=' + valor);
```

Código 7. Uso de la función open() para abrir una nueva ventana

Video . "JavaScript nueva página"
https://bit.ly/3fbJVSH

7.2. Pantalla

A través del objeto window se puede tener información de la pantalla del usuario. Hay ocasiones en las que esta información es relevante parar adaptar la representación visual del contenido web.

En el siguiente ejemplo se captura el ancho de la pantalla del usuario y se añade a la etiqueta del párrafo.

```
<!DOCTYPE html>
<html>
<body>
<p id=”test”></p>
<script>
document.getElementById(“test”).innerHTML =
“Ancho de la pantalla “ + screen.width;
</script>
</body>
</html>
```

Código 8. Captura del ancho de pantalla del usuario

7.3. Cookies

Las cookies permiten almacenar datos web de los usuarios en la máquina cliente. Consiste en pequeños archivos de texto con cierta información relevante para el servidor. Cuando el servidor envía una página web al cliente, la conexión se cierra, lo cual provoca que el servidor no sea capaz de saber en qué estado está la conexión. Este funcionamiento se debe a que el protocolo HTTP no puede guardar el estado de la conexión. Las cookies permiten dar solución a problemas como los siguientes:

- Es posible almacenar el nombre de usuario cuando visite una web.
- La próxima vez que el usuario visita la web, la cookie permitirá recordar su nombre.

Fig.6. JavaScript ofrece funcionalidades para acceder a la ventana

JavaScript proporciona métodos para manejar todo el ciclo de vida de una cookie: creación, edición y borrado.

/ 8. Caso práctico 2: “DOM”

Planteamiento. Después de que nos asignen un proyecto para el desarrollo JavaScript, nuestro responsable nos pide crear un código JavaScript que permita consultar el valor que toma un elemento en el documento HTML.

Nudo. ¿Es posible realizar esto? ¿Qué debería cumplir el documento HTML?

Desenlace. Como hemos visto en el apartado sobre el DOM, es posible consultar, borrar, modificar o agregar elementos a un documento HTML. En este sentido, JavaScript ofrece mecanismos para realizar esta tarea. Sin embargo, es necesario que el documento HTML incluya identificadores o algún campo que permita identificar el campo que queremos consultar.

Veamos una posible solución del caso planteado:

```
<html>
<head>
        <title>Ejemplo de resolución</title>
</head>
        <input type="text" id="value" placeholder="Introduce un valor">
        <button onclick="showValue()">Muestra valor</button>

        <script>
                function showValue(){
                        var value = document.getElementById("value").value;
                        alert("El valor es: "+value);
                }
        </script>

</html>
```

Código 9. Desenlace caso práctico 2

/ 9. Resumen y resolución del caso práctico de la unidad

A lo largo de este tema hemos presentado:

- Las principales características de un documento HTML cuando se interpreta por el navegador, es decir, el DOM de HTML. Como hemos visto, utilizando JavaScript podemos acceder al DOM y realizar las tareas necesarias.

- Cómo a través de los marcos del documento web podemos gestionar diferentes páginas que se visualizan con una única representación. Sin embargo, en la actualidad la utilización de marcos está en desuso.

- La manera en que con el DOM de HTML y JavaScript podemos realizar modificaciones en un documento web dinámicamente, por ejemplo, pudiendo añadir elementos o borrar otros. Como se ha expuesto, esto solo es posible si utilizamos las funcionalidades de JavaScript para acceder a los elementos del HTML que queramos modificar. Para ello, sería necesario definir correctamente los atributos de los elementos relevantes en el documento HTML.

Resolución del caso práctico de la unidad
Como hemos visto en este tema, la utilización de frames o marcos representa un problema real cuando se utilizan diferentes tipos de dispositivos. A partir de HTML5 los frames dejan de estar soportados.
Para ofrecer una mejor experiencia de usuario, deberíamos proponer al cliente crear una versión móvil para lograr una visualización correcta y favorecer la interacción con la web. Por ello, ofreceríamos la posibilidad de crear una versión para dispositivos no móviles con frames y una versión móvil para eliminar los frames.

/ 10. Bibliografía

Flanagan, D. (2020). JavaScript: The Definitive Guide. (7th edition). Sebastopol: O'Reilly Media, Inc.

Haverbeke, M. (2019). Eloquent JavaScript: a modern introduction to programming. San Francisco: No Starch Press.